# ***SAST – Static Application Security Testing***

Static analysis, also known as static application security testing (SAST), examines the source code of apps to find specific flaws that could pose a major risk to your company.

## **How Does SAST Work?**

Static analysis tools are made to examine and find errors in code, from minor readability and style problems to major vulnerabilities that may be caused by the usage of improper programming constructs or be revealed by environmental changes.

Similar to how a security guard's job is to keep anyone with evil intentions out of the building, a static code analyzer scans source code for lines of code that could be used by an anonymous user to insert indicators of harmful activity onto a website or application.

## **Benefits of SAST:**

* Source code is scanned using static application security testing (SAST) to check for anomalies that can point to a security flaw.
* As a result of moving security "Left," SAST tools can be used early in the SDLC (Software Development Life Cycle) to find vulnerabilities before your code is even compiled. This enables for vulnerability detection during the build step.
* Real-time issues are reported through static application security testing (SAST).
* A development team's existing toolkit can be simply expanded with SAST tools. They will be able to do scalable testing on their codebase as a result, allowing developers the freedom to decide how and when to test their applications without placing unjustified constraints on them or their projects.

## **Drawbacks of SAST:**

* It is crucial to carry out additional testing in dynamic environments whenever possible because it doesn't give any information about how apps or their components operate in dynamic situations.
* Assessments of static application security have a significant likelihood of reporting false positives, which might exaggerate the perception of a project's vulnerabilities.
* It is crucial to be able to execute a fresh scan every few hours in order to keep track of the most recent revisions to reports because static application security testing (SAST) is only as good as its most recent scan.

# ***LINTER***

A linter is a tool that may be used to make your code better. But how does it accomplish that? The solution is to examine your source code for errors.

The word "linter" is derived from a C source code analysis tool initially known as "lint." This tool was created in 1978 by computer scientist Stephen C. Johnson while he was employed by Bell Labs.

The original lint tool and earlier tools with a similar purpose both aimed to analyze source code and produce compiler optimizations. As time went on, lint-like tools began to incorporate numerous additional tests and verifications.

Linters aren't only used with compiled languages, as we indicated in the beginning. Contrarily, we may argue that since interpreted languages lack a compiler that can catch faults while they are being developed, linters are significantly more important.

## Advantages of Linting:

* **Fewer production faults:** Linters are used to identify and correct technical problems in the code, such as code smells. As a result, fewer flaws enter the production process.
* **Code is easier to read, maintain, and more consistent:** By enforcing its standards, Linters can assist teams in developing a more understandable and consistent style.
* **Less debate during code reviews regarding the aesthetics of the code:** Discussions of code reviews shouldn't be rife with pointless debates over aesthetic preferences. Linters can move the subjects to the side.
* **The measurement of code quality objectively:** Code quality discussions frequently stray towards subjectivity. Linters offer a quantitative, impartial evaluation of the quality of the code.
* **Code becomes more efficient and secure:** Though not all linters examine source code for efficiency and safety, some do.
* **More developers are exposed to education regarding code quality:** Linters can aid developers in learning about good code, especially the most novice ones.

## Drawbacks of Linting:

* Code Verifications Are Limited To A Single Programming Language.
* Linters check code against the predefined rules for stylistic and programming faults. While excellent for spotting mistakes when using accepted guidelines, this could provide issues when utilizing more experimental coding techniques.
* Linters are frequently criticized for producing a lot of false positives. In other words, they mark issues in source code that are not actually issues.
* Linting tools are useful for simple metrics, but they might not be the ideal option for reporting on more intricate software quality indicators. This contains cyclomatic complexity, a measure of a program's complexity.
* Linters do not guarantee that best practices are taught, even when they do identify them. Linting tools can help developers make their code better, but they might not be able to duplicate best practices.
* Although open-source technologies appear to be "free," there are hidden charges associated with many of its limitations. For instance, maintaining and monitoring each tool is necessary when integrating many tools into the development workflow. Changes will need to be made when new upgrades are made.