**Experiment 4**

**Aim:** Study of SAST Tools

**Theory:**

The study of **Static Application Security Testing (SAST) tools** focuses on understanding how these tools analyze source code, bytecode, or binaries to detect security vulnerabilities without executing the code. SAST tools are typically used during the software development process to find flaws early in the development cycle.

### **Key aspects of studying SAST tools:**

1. **How SAST tools work**:
   * SAST tools scan the application's source code or compiled code to identify vulnerabilities such as SQL injection, cross-site scripting (XSS), and buffer overflows.
   * These tools don’t require the application to be running; they analyze the code statically (hence "static").
   * By examining the flow and logic of the code, they can detect security risks based on patterns and known vulnerabilities.
2. **Common vulnerabilities SAST tools detect**:
   * **Input validation issues**: Weaknesses that could allow attackers to inject harmful input into the system.
   * **Authentication and authorization flaws**: Problems in how users are verified and granted access.
   * **Buffer overflows**: When the application writes more data to a buffer than it can hold, potentially leading to system crashes or exploits.
   * **Code injection vulnerabilities**: Issues like SQL injection or command injection, where untrusted data is processed by an interpreter.
   * **Insecure API usage**: When developers misuse APIs in ways that could lead to vulnerabilities.
3. **Advantages of SAST**:
   * **Early detection**: Finds vulnerabilities early in the development phase, reducing the cost and effort required to fix them later.
   * **Wide language support**: Many SAST tools support multiple programming languages, making them versatile for different projects.
   * **No need for running code**: SAST tools can work on non-executable code, meaning they can be applied to incomplete projects.
4. **Challenges with SAST tools**:
   * **False positives**: These tools may flag potential vulnerabilities that are not actual security threats, leading to unnecessary work.
   * **Limited context**: SAST tools might not fully understand the application's runtime environment, which can limit their ability to detect certain types of vulnerabilities.
   * **Performance overhead**: Analyzing large codebases can be slow and resource-intensive.

Here’s a more detailed look at some popular **Static Application Security Testing (SAST)** tools, highlighting their features, strengths, and use cases:

### **1. SonarQube**

* **Overview**: SonarQube is an open-source platform that continuously inspects the quality of code and highlights vulnerabilities, code smells, and security issues. It supports multiple languages and integrates with many popular development environments.
* **Key Features**:
  + **Multi-language support**: SonarQube supports over 25 programming languages, including Java, C/C++, Python, JavaScript, and more.
  + **Code Quality & Security**: Besides security vulnerabilities, SonarQube also detects issues related to code maintainability, reliability, and duplication.
  + **Integration with DevOps**: It integrates seamlessly with popular CI/CD pipelines like Jenkins, GitLab, and Azure DevOps, allowing automated scanning during code commits or builds.
  + **OWASP Top 10 & SANS Top 25**: SonarQube includes predefined rules that check for vulnerabilities related to these widely recognized security standards.
* **Strengths**:
  + Strong community support and frequent updates due to its open-source nature.
  + **Easy to set up and use**: Offers a web-based interface to review security issues, making it user-friendly for development teams.
  + **Customizable rules**: Developers can add or modify rules to suit their project needs.
* **Use Cases**:
  + Ideal for development teams looking for a free, easy-to-use solution to monitor both code quality and security.
  + Commonly used in agile environments and DevOps pipelines to ensure continuous code health checks.

### **2. Checkmarx**

* **Overview**: Checkmarx is a commercial, comprehensive SAST tool designed to detect security vulnerabilities in the early stages of the software development lifecycle. It’s widely used by enterprises to ensure application security at scale.
* **Key Features**:
  + **Deep code scanning**: Checkmarx uses static code analysis to detect a wide array of vulnerabilities, including SQL injection, XSS, insecure configurations, and more.
  + **Scans across languages and frameworks**: Supports more than 30 programming languages, including legacy languages, making it versatile for organizations with diverse tech stacks.
  + **Security-focused**: Prioritizes identifying security risks based on standards such as OWASP, PCI-DSS, and GDPR compliance.
  + **Integration & Automation**: Works with popular DevOps tools like Jenkins, Bamboo, GitHub, and Bitbucket. It integrates well with IDEs (e.g., Visual Studio, IntelliJ) to help developers identify issues as they code.
* **Strengths**:
  + **Scalability**: Designed for large enterprises with vast and complex codebases.
  + **Detailed reporting**: Offers comprehensive vulnerability reports that detail the source of the vulnerability, the risk level, and suggestions for remediation.
  + **Custom queries**: Checkmarx allows users to write their own security rules and queries, providing greater flexibility in scanning tailored to specific applications.
* **Use Cases**:
  + Enterprise-level applications where security is a priority from the beginning of the development cycle.
  + Ideal for organizations requiring robust governance and compliance with strict security standards.

### 

### **3. Fortify Static Code Analyzer (Fortify SCA)**

* **Overview**: Fortify SCA, by OpenText (formerly Micro Focus), is a well-established SAST tool that helps organizations detect and fix security vulnerabilities in code before it is deployed. It is designed to support a broad range of languages and platforms, making it suitable for large and complex applications.
* **Key Features**:
  + **Wide language support**: Fortify SCA supports 27+ programming languages and numerous frameworks, including Java, C/C++, .NET, Python, and PHP.
  + **Automated scanning**: Provides an automated process for security vulnerability detection throughout the development lifecycle, ensuring secure software delivery.
  + **Security rule packs**: Comes with over 1000 built-in security rules, based on standards like OWASP Top 10, PCI-DSS, and NIST, that target common vulnerabilities such as SQL injection, XSS, and buffer overflows.
  + **Custom rule creation**: Fortify allows organizations to extend its detection capabilities by creating custom rules to address specific application vulnerabilities.
* **Strengths**:
  + **Enterprise-grade**: Known for its scalability and performance, making it suitable for large organizations with complex and multi-layered applications.
  + **Comprehensive reporting**: Fortify SCA provides detailed security analysis reports that highlight vulnerabilities, their severity, and suggestions for remediation.
  + **Secure coding guidance**: It provides developers with step-by-step instructions on how to fix identified security issues, reducing the learning curve.
* **Use Cases**:
  + Large enterprises that require an extensive security scanning solution to comply with regulations and standards like PCI-DSS, HIPAA, or GDPR.
  + Organizations that need a deeply integrated SAST solution to work with their existing development environments and processes.

### 

### **4. Veracode Static Analysis**

* **Overview**: Veracode is a cloud-based SAST tool known for its ease of use and focus on identifying security vulnerabilities in code. It helps development teams quickly scan code for security risks while maintaining high development speed.
* **Key Features**:
  + **Cloud-based solution**: Veracode operates entirely in the cloud, reducing the need for on-premise hardware and allowing easy scaling. It’s easy to integrate into cloud-based CI/CD pipelines.
  + **Policy-driven scanning**: Veracode allows organizations to enforce security policies automatically, ensuring that code meets specific security standards before it is deployed.
  + **Language support**: Supports over 30 languages, including Java, JavaScript, Python, C#, and Ruby.
  + **Developer-centric feedback**: Veracode emphasizes providing actionable feedback to developers, helping them understand the security issues and how to fix them.
  + **Third-party component scanning**: It includes the ability to scan open-source and third-party libraries for vulnerabilities, helping organizations manage their dependencies.
* **Strengths**:
  + **Ease of integration**: Veracode integrates well with many tools, including Jira, Jenkins, GitHub, and GitLab, enabling seamless security scanning during the development process.
  + **Developer education**: Offers educational resources and remediation tips to help developers better understand security issues and improve their secure coding practices.
  + **Speed and scalability**: As a cloud-based solution, Veracode is optimized for performance and is capable of handling frequent scans without impacting infrastructure.
* **Use Cases**:
  + Ideal for agile teams working with cloud-based CI/CD pipelines that want to ensure security without slowing down development.
  + Organizations that need fast, reliable scans with minimal on-premise setup.

**Conclusion**: Successfully learned what are SAST tools, it’s limitations and some of the tools available in the market.