**SAD Lab**

**Exp-4**

**Name: Shashwat Tripathi**

**Division: D20A**

**Roll no: 64**

**Aim:** Study of SAST(Static Application Security Testing) Tools

**Theory:**

**What is SAST?**

SAST (Static Application Security Testing) is a software security testing technique that analyzes an application's source code or compiled code without executing it. It examines the code for potential security vulnerabilities, such as SQL injection, cross-site scripting (XSS), buffer overflows, and insecure coding practices.

There are six simple steps needed to perform SAST efficiently in organizations that have a very large number of applications built with different languages, frameworks, and platforms.

1. **Finalize the tool.** Select a static analysis tool that can perform code reviews of applications written in the programming languages you use. The tool should also be able to comprehend the underlying framework used by your software.
2. **Create the scanning infrastructure, and deploy the tool.** This step involves handling the licensing requirements, setting up access control and authorization, and procuring the resources required (e.g., servers and databases) to deploy the tool.
3. **Customize the tool.** Fine-tune the tool to suit the needs of the organization. For example, you might configure it to reduce false positives or find additional security vulnerabilities by writing new rules or updating existing ones. Integrate the tool into the build environment, create dashboards for tracking scan results, and build custom reports.
4. **Prioritize and onboard applications.** Once the tool is ready, onboard your applications. If you have a large number of applications, prioritize the high-risk applications to scan first. Eventually, all your applications should be onboarded and scanned regularly, with application scans synced with release cycles, daily or monthly builds, or code check-ins.
5. **Analyze scan results.** This step involves triaging the results of the scan to remove false positives. Once the set of issues is finalized, they should be tracked and provided to the deployment teams for proper and timely remediation.
6. **Provide governance and training.** Proper governance ensures that your development teams are employing the scanning tools properly. The software security touchpoints should be present within the SDLC. SAST should be incorporated as part of your application development and deployment process.

**Importance**

1. **Early Vulnerability Detection:**

SAST tools can identify vulnerabilities as early as the development phase, enabling developers to address issues before they become critical. This significantly reduces the cost and time associated with fixing defects later in the development lifecycle.

1. **Improved Code Quality:**

By enforcing secure coding standards and best practices, SAST helps in producing higher quality code with fewer vulnerabilities. This leads to more reliable and resilient software systems.

1. **Risk Mitigation:**

Proactive identification of security weaknesses helps organizations to prioritize remediation efforts and mitigate potential risks. SAST can be used to assess the overall security posture of an application and identify areas requiring improvement.

1. **Compliance Adherence:**

Many industries and regulatory bodies have strict security requirements. SAST can help organizations demonstrate compliance with these standards by identifying and addressing vulnerabilities.

**Limitations**

1. **False Positives:**

SAST tools may sometimes generate false positives, indicating vulnerabilities that do not actually exist. This can lead to wasted time and effort in investigating and resolving non-existent issues.

1. **Limited Scope:**

SAST primarily focuses on code-level vulnerabilities and may not detect runtime or configuration-based issues. It cannot identify vulnerabilities that arise due to interactions between different components or systems.

1. **Complexity and Performance Overhead:**

Analyzing large and complex codebases can be computationally expensive and time-consuming. SAST tools may impact developer productivity if not integrated efficiently into the development workflow.

1. **Tool Limitations:**

Not all SAST tools are created equal. Some tools may have limitations in terms of supported programming languages, frameworks, and vulnerability detection capabilities.

1. **Remediation Guidance:**

Effective remediation guidance helps developers fix vulnerabilities quickly and accurately.

1. **Integration with Development Workflow:**

Seamless integration into the development process is essential for efficient use of SAST.

1. **Cost and Licensing:**

Consider the cost of the tool and the licensing model that best suits your organization's needs.

1. **Support and Training:**

Adequate support and training resources are important for successful tool adoption.

**SAST Tools in detail(Any 5)**

**Checkmarx:**

Checkmarx is a comprehensive application security platform that offers SAST, DAST, IAST, and SCA capabilities. It is designed to provide a holistic view of an application's security posture.

**Key Features:**

* Advanced Code Analysis: Employs sophisticated techniques like semantic analysis and machine learning to accurately identify vulnerabilities.
* Wide Language Support: Covers a broad spectrum of programming languages, including Java, C++, C#, Python, JavaScript, and more.
* Deep Code Analysis: Goes beyond syntax checking to analyze code logic and data flow, uncovering hidden vulnerabilities.
* Remediation Guidance: Provides actionable recommendations to fix identified vulnerabilities, including code snippets and best practices.
* False Positive Reduction: Uses advanced algorithms to minimize false positives, improving efficiency.

**SonarQube:**

While primarily known for code quality, SonarQube also offers robust SAST capabilities. It focuses on improving code maintainability, reliability, and security.

**Key Features:**

* Code Quality Metrics: Provides detailed metrics on code complexity, duplication, test coverage, and security hotspots.
* Continuous Inspection: Offers real-time analysis of code changes to prevent vulnerabilities from being introduced.
* Multiple Language Support: Supports a wide range of programming languages, including Java, C#, JavaScript, Python, and more.
* Security Rules: Contains a built-in set of security rules to identify common vulnerabilities like SQL injection, XSS, and buffer overflows.
* Visualization: Provides visual representations of code quality and security issues, making it easier to understand and prioritize remediation efforts.
* Open Source: Offers a free community edition for smaller projects.

**Fortify Static Code Analyzer:**

Fortify is a dedicated SAST tool focused on vulnerability detection and prevention. It provides in-depth analysis of code to identify potential security risks.

**Key Features:**

* Deep Vulnerability Detection: Uses advanced algorithms to uncover complex vulnerabilities that other tools might miss.
* Prioritization: Ranks vulnerabilities based on severity and potential impact, helping to focus remediation efforts.
* Remediation Guidance: Offers clear and actionable recommendations for fixing identified vulnerabilities.
* Integration: Integrates with various development environments and CI/CD pipelines.
* Compliance Reporting: Provides reports to demonstrate compliance with security standards and regulations.
* Customizable Rules: Allows users to create custom rules to address specific security requirements.

**Veracode Static Analysis:**

Veracode is a cloud-based SAST solution that offers fast scan times and easy integration. It provides comprehensive vulnerability assessment and remediation guidance.

**Key Features:**

* Cloud-Based Delivery: Eliminates the need for on-premise infrastructure, simplifying deployment and management.
* Rapid Scan Times: Provides quick analysis results, allowing for faster development cycles.
* Vulnerability Prioritization: Ranks vulnerabilities based on severity and exploitability.
* Remediation Guidance: Offers detailed guidance on how to fix identified vulnerabilities.
* Compliance Reporting: Generates reports to demonstrate compliance with industry standards and regulations.
* Integration: Integrates with various development tools and CI/CD pipelines.

**Coverity:**

Coverity is a comprehensive SAST tool with advanced code analysis capabilities. It provides deep code coverage and accurate vulnerability detection.

**Key Features:**

* Deep Code Analysis: Analyzes code at multiple levels, including data flow, control flow, and semantic analysis.
* Accurate Vulnerability Detection: Identifies a wide range of vulnerabilities with high precision.
* Code Coverage: Measures the extent to which code is exercised by tests, ensuring thorough testing.
* Defect Prevention: Helps prevent defects from being introduced into the codebase by identifying potential issues early in the development process.
* Integration: Integrates with various development tools and CI/CD pipelines.
* Scalability: Handles large codebases efficiently.

**Conclusion:** Thus, we have studied and understood the SAST(Static Application Security Testing) Tools