NAME: MULLA AFRAH AKKAS ALI

ROLL NO.: 612038

BRANCH: T.E. – I.T.

SEMESTER: ODD SEMESTER 5

COURSE: Advance DevOPs (ITL504)

DATE: 24-10-2022

EXPERIMENT 14

1. What is SonarQube? Why use SonarQube?

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications.

SonarQube can record metrics history and provides evolution graphs. SonarQube provides fully automated analysis and integration with Maven, Ant, Gradle, MSBuild and continuous integration tools (Atlassian Bamboo, Jenkins, Hudson, etc.).

SonarQube includes support for the programming languages Java (including Android), C#, C, C++, JavaScript, TypeScript, Python, Go, Swift, COBOL, Apex, PHP, Kotlin, Ruby, Scala, HTML, CSS, ABAP, Flex, Objective-C, PL/I, PL/SQL, RPG, T-SQL, VB.NET, VB6, and XML. As of December 2021, analyzing C, C++, Obj-C, Swift, ABAP, T-SQL and PL/SQL is only available via a commercial license.

In order to deliver the required functionalities on time, achieve and maintain good quality code and continuous code integration and deployment, SonarQube is used.

2.What is software quality measurement?

Software quality measurement is a quantitative process summing up weighted attribute values, which in part describe specific software characteristics. For each characteristic, a set of such measurable attributes is defined. Software quality measurement helps you understand the level of quality of each release, even each build, produced by your development team.

The ISO/IEC 25010 standard provides a useful model of 8 software quality dimensions. They are –

- Maintainability
- Portability
- Functionality
- Performance
- Compatibility
- Usability
- Reliability
- Security

3.What is Static and Dynamic Code Analysis?

Static Code Analysis

Static code analysis is a process for analyzing an application's code for potential errors. It is “static” because it analyses applications without running them, which means an application can be tested exhaustively without constructing a runtime environment or posing risk to production systems. This makes static code analysis very well suited to testing applications for security flaws, a process called Static Application Security Testing (SAST).

SAST tools work by “modeling” an application to map control and data flows based upon analysis of the application’s source code. The analysis compares the code to a predefined set of rules to identify potential security issues.

Static analysis tools are best at identifying vulnerabilities that are easily detectable within an application’s source code. This includes common vulnerabilities such as:

- Injection (SQL, LDAP, etc.)
- Cross-Site Scripting
- Buffer Overflows

Additionally, SAST tools are relatively easy to integrate into a development workflow. Since they are only applied to application source code – and don’t require a realistic execution environment – they can be incorporated into DevOps automated continuous integration/continuous deployment (CI/CD) workflows and applied automatically. This reduces the workload on developers and enables them to focus on the task at hand.

Dynamic Code Analysis

Dynamic code analysis – also called Dynamic Application Security Testing (DAST) – is designed to test a running application for potentially exploitable vulnerabilities. DAST tools to identify both compile time and runtime vulnerabilities, such as configuration errors that only appear within a realistic execution environment.

A DAST tool uses a dictionary of known vulnerabilities and malicious inputs to “fuzz” an application. Examples of these potentially malicious inputs include:

- SQL queries (to identify SQL injection vulnerabilities)
- Long input strings (to exploit buffer overflow vulnerabilities)
- Negative and large positive numbers (to detect integer overflow and underflow vulnerabilities)
- Unexpected input data (to exploit invalid assumptions by developers)

As the application runs, it is bombarded by these potentially malicious inputs, and the DAST tool analyzes the responses from the application. If the application has a negative response to an input (such as crashing, returning an invalid response, etc.), then the DAST tool records the identified vulnerability.

Since DAST tools are executed on a running application they can detect a wide range of potential vulnerabilities. This includes vulnerabilities that are difficult or impossible to detect in source code, such as memory allocation issues.

4.What are the benefits of using SonarQube?

Sustainability - Reduces complexity, possible vulnerabilities, and code duplications, optimising the life of applications.

Increase productivity - Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code

Quality code - Code quality control is an inseparable part of the process of software development.

Detect Errors - Detects errors in the code and alerts developers to fix them automatically before submitting them for output.

Increase consistency - Determines where the code criteria are breached and enhances the quality

Business scaling - No restriction on the number of projects to be evaluated

Enhance developer skills - Regular feedback on quality problems helps developers to improve their coding skills