

Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Theory:

Static application security testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabilities that make your organization's applications susceptible to attack. SAST scans an application before the code is compiled. It's also known as white box testing.

What problems does SAST solve?

SAST takes place very early in the software development life cycle (SDLC) as it does not require a working application and can take place without code being executed. It helps developers identify vulnerabilities in the initial stages of development and quickly resolve issues without breaking builds or passing on vulnerabilities to the final release of the application.

SAST tools give developers real-time feedback as they code, helping them fix issues before they pass the code to the next phase of the SDLC. This prevents security-related issues from being considered an afterthought. SAST tools also provide graphical representations of the issues found, from source to sink. These help you navigate the code easier. Some tools point out the exact location of vulnerabilities and highlight the risky code. Tools can also provide in-depth guidance on how to fix issues and the best place in the code to fix them, without requiring deep security domain expertise. It's important to note that SAST tools must be run on the application on a regular basis, such as during daily/monthly builds, every time code is checked in, or during a code release.

Why is SAST important?

Developers dramatically outnumber security staff. It can be challenging for an organization to find the resources to perform code reviews on even a fraction of its applications. A key strength of SAST tools is the ability to analyze 100% of the codebase. Additionally, they are much faster than manual secure code reviews performed by humans. These tools can scan millions of lines of code in a matter of minutes. SAST tools automatically identify critical vulnerabilities—such as buffer overflows, SQL injection, cross-site scripting, and others—with high confidence.

Thus, integrating static analysis into the SDLC can yield dramatic results in the overall quality of the code developed.

What are the key steps to run SAST effectively?

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.

Integrating Jenkins with SonarQube: Windows installation

Step 1 Install JDK 1.8

Step 2 download and install
jenkins

installing-the-default-jre-

jdk Step 1 Install JDK 1.8

sudo apt-get install openjdk-
8-jre sudo apt install default-
jre /

how-to-install-jenkins-on-ubuntu-20-04

Open SSH

Prerequisites:

- Jenkins installed
- Docker Installed (for SonarQube)

(sudo apt-get install docker-ce=5:20.10.15~3-0~ubuntu-jammy docker-ce-
cli=5:20.10.15~3-0~ubuntu-jammy containerd.io docker-compose-plugin)

- SonarQube Docker Image

Prajyot Shinde 57/D15A

```
C:\Users\91952>docker run -d --name sonarqube -p 9000:9000 sonarqube
0392bc54d6d9ecd3e651c04433976aae2ab159b63bc07776b81a03ebf6e754b5
```

```
C:\Users\91952>docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
postgres	latest	026d0ab72b34	7 weeks ago	611MB
sonarqube	latest	72e9feec7124	2 months ago	1.92GB

```
C:\Users\91952>
```

The screenshot shows a web browser window with the address bar displaying `localhost:9000/maintenance?return_to=%2F`. The page header includes the SonarQube logo and a navigation menu with links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The main content area displays a 'Create a local project' form with the following fields:

- Project display name ***: `sonarqube-test` (with a green checkmark)
- Project key ***: `sonarqube-test` (with a green checkmark)
- Main branch name ***: `main`

Below the form, there is a note: 'The name of your project's default branch [Learn More](#)'. At the bottom of the form are 'Cancel' and 'Next' buttons. A yellow warning banner at the bottom of the page states: 'Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.'

Ok so go to Jenkins dashboard and then go to Manage Jenkins then plugins , install plugins and then install “sonarqube scanner”

Dashboard > Manage Jenkins > Plugins

Plugins

Updates29

Available plugins

Installed plugins

Advanced settings

Search installed plugins

Script Security134.1va_2619b_4146b6

Allows Jenkins administrators to control what in-process scripts can be run by less-privileged users.

Report an issue with this plugin

✓

✕

SnakeYAML API2.2-111.vc6598e30cc65

This plugin provides SnakeYAML for other plugins.

Report an issue with this plugin

✓

✕

SonarQube Scanner for Jenkins2.17.2

This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.

Report an issue with this plugin

✓

✕

SSH Build Agents plugin2.973.v0fa_8c0dea_f9f

Allows to launch agents over SSH, using a Java implementation of the SSH protocol.

Report an issue with this plugin

✓

✕

SSH Credentials Plugin343.v884f71d78167

Allows storage of SSH credentials in Jenkins

Report an issue with this plugin

✓

✕

Jenkins

Search (CTRL+K)

🔔🛡️🔴

Prajyot Shinde

log out

Dashboard > Manage Jenkins > Plugins

Plugins

Updates35

Available plugins

Installed plugins

Advanced settings

sona

Name

Enabled

SonarQube Scanner for Jenkins2.17.2

This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.

Report an issue with this plugin

✓

✕

REST API

Jenkins 2.462.2

Dashboard > Manage Jenkins > System >

SonarQube installations

List of SonarQube installations

Name

sonarqube

Server URL

Default is http://localhost:9000

http://localhost:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

- none -

+ Add

Save

Apply

Jenkins

Search (CTRL+K)

log out

Dashboard > All > New Item

New Item

Enter an item name

Sonarube

Select an item type

Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

OK

Dashboard > sonarQube > Configuration

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Source Code Management

None

Git

Repositories

Repository URL

https://github.com/shazforiot/MSBuild_firstproject.git

Credentials

- none -

+ Add

Save

Apply

Dashboard > sonarQube2 > Configuration

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Path to project properties

Analysis properties

Additional arguments

JVM Options

Save

Apply

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministrationMore

Administration

ConfigurationSecurityProjectsSystemMarketplace

	Administer System	Administer	Execute Analysis	Create
<div>sonar-administrators</div> <div>System administrators</div>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Quality Gates <input checked="" type="checkbox"/> Quality Profiles	<input type="checkbox"/>	<input checked="" type="checkbox"/> Projects
<div>sonar-users</div> <div>Every authenticated user automatically belongs to this group</div>	<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Projects
<div>Anyone</div> <div>DEPRECATED</div> <div>Anybody who browses the application belongs to this group. If authentication is not enforced, assigned permissions also apply to non-authenticated users.</div>	<input type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input type="checkbox"/>	<input type="checkbox"/> Projects
<div>Administrator</div> <div>admin</div>	<input checked="" type="checkbox"/>	<input type="checkbox"/> Quality Gates <input type="checkbox"/> Quality Profiles	<input checked="" type="checkbox"/>	<input type="checkbox"/> Projects

After 3 failures finally there was 4th success which is shown down below in image

The image shows the Jenkins Dashboard for the SonarQube project. The top navigation bar includes the Jenkins logo, a search bar, and user information (Prajyot Shinde). The main content area is divided into two sections. On the left, a sidebar contains links to various project actions: Status, Changes, Workspace, Build Now, Configure, Delete Project, SonarQube, and Rename. The main section displays the SonarQube status as 'Success' with a green checkmark. Below this, there are 'Permalinks' for various builds: Last build (#4), Last stable build (#4), Last successful build (#4), Last failed build (#3), Last unsuccessful build (#3), and Last completed build (#4). A 'Build History' table is also visible, showing the status of builds #4, #3, and #2. Build #4 is marked as successful, while #3 and #2 are marked as failed.

The image shows the Jenkins Console Output for build #4. The top navigation bar is the same as the previous image. The main content area is titled 'Console Output' with a green checkmark. Below the title, there are buttons for 'Download', 'Copy', and 'View as plain text'. The console output text shows the build process starting with 'Started by user Prajyot Shinde' and 'Running as SYSTEM'. It details the build configuration, including the workspace path, the recommended git tool, and the git repository URL. The output shows the successful execution of 'git.exe rev-parse', 'git.exe config', 'git.exe fetch', 'git.exe checkout', and 'git.exe rev-list'. It also shows the successful execution of 'sonar-scanner' with the command 'sonar-scanner -Dsonar.projectKey=sonarqube-test -Dsonar.testdata=...'. The final output shows the build status as 'Success'.

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministrationMore

sonarqube-testmain

OverviewIssuesSecurity HotspotsMeasuresCodeActivity

Project SettingsProject Information

main

Version not providedSet as homepage

Quality Gate

Passed

Last analysis 3 minutes ago

The last analysis has warnings. See details

New CodeOverall Code

Security

0 Open issues

0 H0 M0 L

Reliability

0 Open issues

0 H0 M0 L

Maintainability

0 Open issues

0 H0 M0 L

Accepted issues

0

Coverage

0.0%

Duplications

0.0%