DevSecOps

DevSecOps is a cultural approach where every team and person working on an application considers security throughout it's lifecycle. It ensures that security is implemented at every stage of the application software development lifecycle (SDLC) by incorporating required security checks embedded into CI/CD automation using appropriate tools.

Stages :

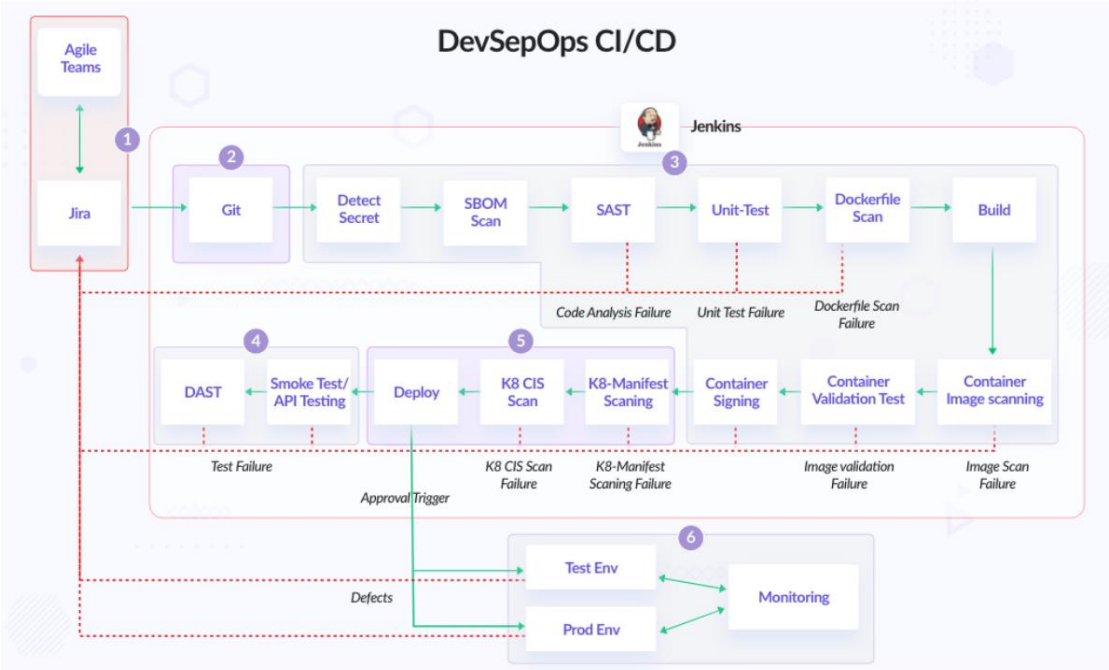
1. Plan/Design
2. Develop
3. Build and Code analysis
4. Test
5. Deploy
6. Monitor and Alert

TechStack :

Sl.No	CheckPoints	Tools
1	Git Secrets Verifications	1. Scan 2. Git-Secrets 3. Detect-Secrets
2	SBOM/SCA	1. OWASP Dependency-Check 2. Trivy 3 CheckOV 4. Anchore 5. Mend.
3	SAST	1. Mend 2. SonarQube 3. Veracode 4. Fortify Static Code Analyser 5. Checkmarx CxSAST
4	Dockerfile static scanning	CheckOV
5	Docker Image Scanning	Trivy
6	Container image signing and verifying	1. Cosign 2. Fulcoi 3. Rekor
7	Container Structure Test	dgoss
8	Dynamic application security testing (DAST)	1. Invicti 2. Acunetix 3. PortSwigger 4. Detectify 5. CheckMarx
9	Container Repository Scanner	Snyk
10	Static scan of Kubernete manifest file or Helm chart	CheckOV
11	Trivy for Kubernetes	Trivy
12	Kubernetes CIS	Kuber-bech

13	Monitoring and Alerting	1. Metrics monitoring : * Prometheus & Grafana * Nagios & Zabbix 2. Log Monitoring : * OpenSearch/Elasticsearch * GrayLog * Grafana-Loki 3.Alerting : * Prometheus Alertmanager * Slack * Grafana OnCall
14	Server Vulnerability Scanner	1.Nikto2 2.Netsparker 3.Nessus Professional 4.OpenVAS 5.W3AF

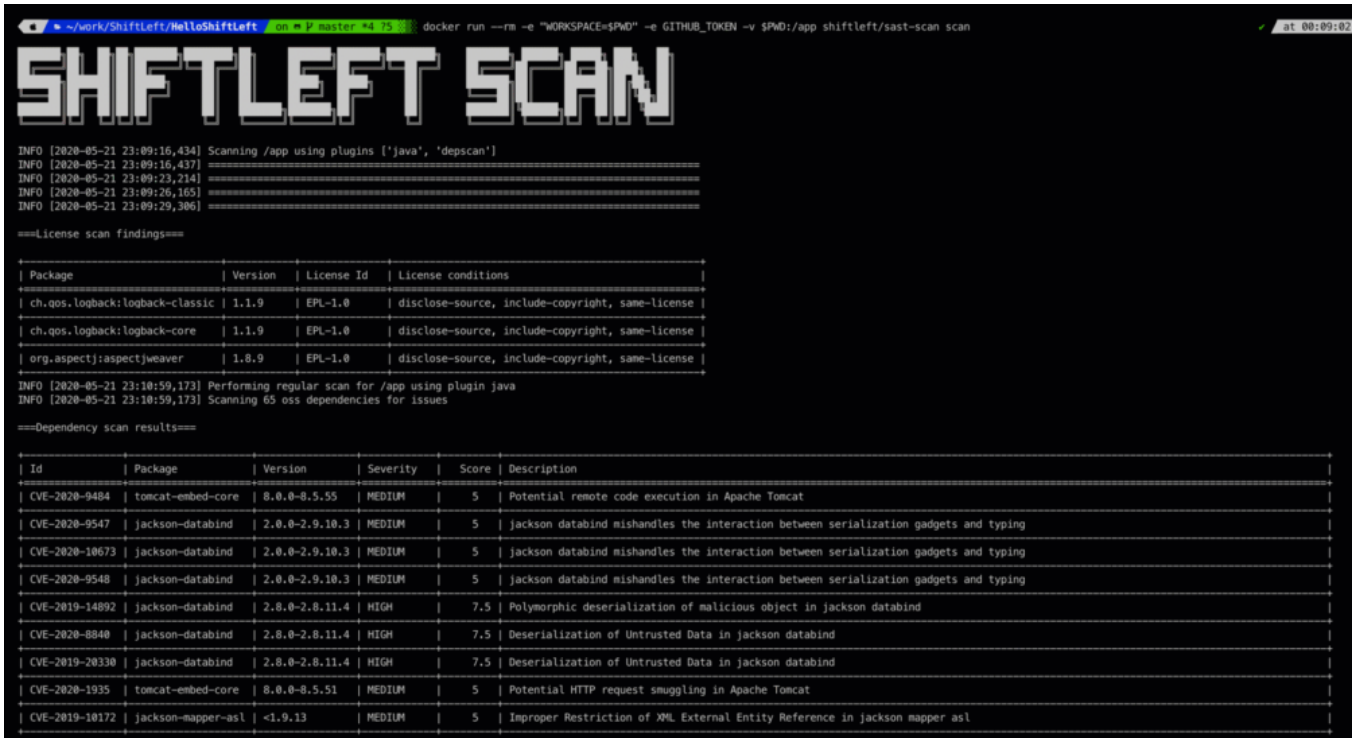
CI/CD Pipeline for DevSecOps :



1) Secrets
1.1 Scan

Scan is a comprehensive open-source security audit tool. It provides strong integration with popular repositories and pipelines such as Azure, BitBucket, GitHub, GitLab, Jenkins, TeamCity, and many more.

Scan also supports a broad section of popular frameworks and languages, integrates into the CI/CD pipeline to provide real-time commit protection, and provides extensive reporting capabilities.



1.2 Git-Secrets

Git-Secrets is an open-source command-line tool used to scan developer commits and "–no-ff" merges to prevent secrets from accidentally entering Git repositories. If a commit or merge matches a regular expression pattern, the commit is rejected.

1.3 Detect-Secret

detect-secret is an enterprise-friendly tool for detecting and preventing secrets in the code base. We can also scan the non-git tracked files. There are other tools as well like Gitleaks which also provide similar functionality.

detect-secrets scan test_data/ --all-files

2) SBOM/SCA

SBOM and SCA lets us Identify all software components, libraries, and modules that are running in our environment, even their dependencies. It speeds up response time for new vulnerabilities - including zero-day vulnerabilities like Log4j.

List of SBOM Tools:

1. OWASP Dependency-Check
2. Trivy
3 CheckOV
4. Anchore
5. Mend

2.1) OWASP Dependency-Check

OWASP Dependency-Check a Software Composition Analysis (SCA) tool that attempts to detect publicly disclosed vulnerabilities contained within a project's dependencies. It does this by determining if there is a Common Platform Enumeration (CPE) identifier for a given dependency. If found, it will generate a report linking to the associated CVE entries. We can also publish our SBOM report to Dependency-Track and visualize our software components and their vulnerabilities.

dependency-check.sh --scan /project_path

Once we know what type of vulnerabilities are present in our software, we can patch them and make our application safe and secure.

3) SAST

Static Application Security Testing (SAST) tools are solutions that scan your application source code or binary and find vulnerabilities. It is known as White-box testing, and developers can use it within the IDE or integrate it into CI/CD pipelines.

Key Features of SAST:

* SAST is a White-Box testing
* Requires Source code or the binary access for analyzing the source-code.
* Finds the Vulnerabilities in the earlier stage of SDLC.

List of SAST Testing Tools:

1. Mend
2. SonarQube
3. Veracode
4. Fortify Static Code Analyser
5. Codacy
6. AppScan
7. Checkmarx CxSAST

4) UNIT TESTING

In Unit tests, individual software code components are checked if it is working as expected or not. Unit tests isolate a function or module of code and verify its correctness. We can use tools like JaCoCo for Java and Mocha, and Jasmine for NodeJS to generate unit test reports. We can also send these reports to SonarQube which shows us code coverage and the percentage of your code covered by your test cases.

Once SAST is done, we can scan our Dockerfile as well.

5) Dockerfile static scanning

Checkov is a static code analysis tool for infrastructure as code (IaC) and also a software composition analysis (SCA) tool for images and open source packages.

It scans cloud infrastructure provisioned using Terraform, Terraform plan, Cloudformation, AWS SAM, Kubernetes, Helm charts, Kustomize, Dockerfile, Serverless, Bicep, OpenAPI or ARM Templates and detects security and compliance misconfigurations using graph-based scanning.

It performs Software Composition Analysis (SCA) scanning which is a scan of open source packages and images for Common Vulnerabilities and Exposures (CVEs).

Example:

Docker File Scan : docker run -i -v $(pwd):/output bridgecrew/checkov -f /output/Dockerfile -o json

Checkov -f /path/of/your/file

Checkov -d /path/of/directory/having/code/files

Checkov –output=json(you can select any output type like json, Html) -d /path/of your terraform code

6) **Docker Image Scanning**:

Trivy is an open-source tool by aqua security to scan for vulnerabilities and misconfiguration errors.

Key Features of Trivy:

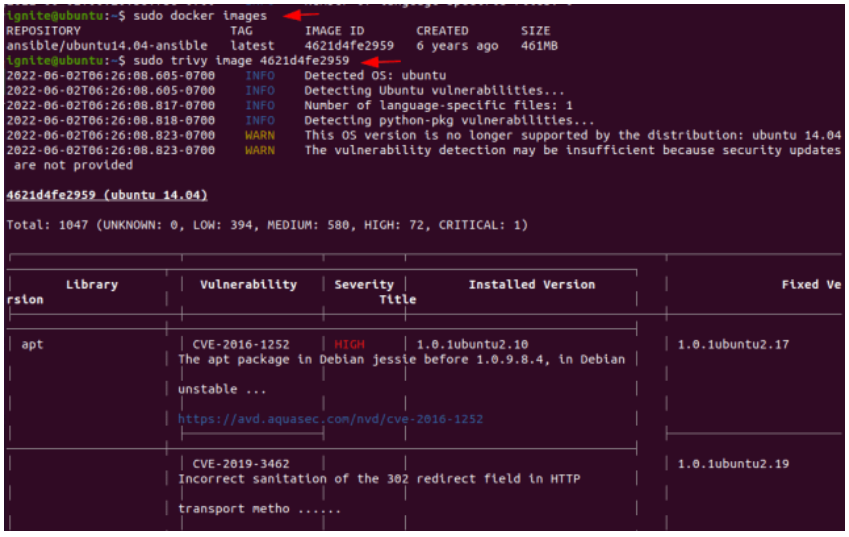
* It can evaluate Infrastructure as Code

* Inspect container images and we will be able to add a separate layer is the dockerfile while building the image.

* Analyze Kubernetes implementations, helps to analyze kubernetes manifest file , pods and deployments.

* Review the code in a Git repository

* Analyze the file systems



Best Practices :

Always scan the Dockerfile for vulnerabilities as while writing Dockerfile we may miss some of the best practices which may lead to vulnerable containers. To name a few common mistakes that we can avoid.

Do not use the latest docker image tag

Ensure that a user for the container has been created

Embed Trivy in Dockerfile

You can also scan the image as part of the build process by embedding Trivy in the Dockerfile. This approach can be used to update Dockerfiles currently using Aqua's Micro scanner. Follow the below-given steps to scan the docker file while building it.

Step1: Add trivy to the docker file.

```
FROM alpine:3.7
RUN apk add curl \
&& curl -sfL https://raw.githubusercontent.com/aquasecurity/trivy/master/contrib/install.sh | sh -s -- -b /usr/local/bin \
&& trivy filesystem --exit-code 1 --no-progress /
```

Step2 : Build the image.

sudo docker build -t vulnerable image .

It will scan the docker file while the image is being built and gives the CVE report .

7) Container image signing and verifying :

If the container build process is compromised, it makes users vulnerable to accidentally using the malicious image instead of the actual container image. Signing and verifying the container always ensure we are running the actual container image.

Using distroless images not only reduces the size of the container image it also reduces the surface attack. The need for container image signing is because even with the distroless images there is a chance of facing some security threats such as receiving a malicious image. We can use cosign or skopeo for container signing and verifying. You can read more about securing containers with **Cosign**.

cosign sign --key cosign.key custom-nginx:latest

cosign verify -key cosign.pub custom-nginx:latest

List of Container Image Signing and Verifying tools:

1. Cosign
2. Fulcoi
3. Rekor

8) **Container Structure Test**:

The Container Structure Tests provide a powerful framework to validate the structure of a container image. These tests can be used to check the output of commands in an image, as well as verify metadata and contents of the filesystem.And an extra layer of security on the container image to verify if it is working as expected and has all required files with correct permissions.

We can use **dgoss** to do validation tests of container images.

For example, let's do a validation test for the nginx image that is running on port 80, has internet access, and verifies the correct file permission of /etc/nginx/nginx.conf, and the nginx user shell in the container.

Example :

dgoss edit nginx

goss add port 80

goss add http <https://google.com>

goss add file /etc/nginx/nginx.conf

goss add user nginx

Once we exit it will copy the goss.yaml from the container to the current directory and we can modify it as per our validation.

Validate

[root@home ~]# dgoss run -p 8000:80 nginx

INFO: Starting docker container

INFO: Container ID: 5f8d9e20

INFO: Sleeping for 0.2

INFO: Container health

INFO: Running Tests

Port: tcp:80: listening: matches expectation: [true]

Port: tcp:80: ip: matches expectation: [["0.0.0.0"]]

HTTP: https://google.com: status: matches expectation: [200]

File: /etc/nginx/nginx.conf: exists: matches expectation: [true]

File: /etc/nginx/nginx.conf: mode: matches expectation: ["0644"]

File: /etc/nginx/nginx.conf: owner: matches expectation: ["root"]

File: /etc/nginx/nginx.conf: group: matches expectation: ["root"]

User: nginx: uid: matches expectation: [101]

User: nginx: gid: matches expectation: [101]

User: nginx: home: matches expectation: ["/nonexistent"]

User: nginx: groups: matches expectation: [["nginx"]]

User: nginx: shell: matches expectation: ["/bin/false"]

Total Duration: 0.409s

Count: 13, Failed: 0, Skipped: 0

INFO: Deleting container

9) Dynamic application security testing (DAST)

DAST is a web application security test that finds security issues in the running application. DAST tools are also known as web application vulnerability scanners which can detect common vulnerabilities like SQL injection, cross-site scripting, security misconfigurations, and other common issues.

Feature of DAST Testing:

* DAST dosnt requires source code or binary access, it executes on the application level.
* Finds the vulnerabilities towards the end of the SDLC.
* Can discover run time environmental issues.
* Typically scans the web application and the web services.

DAST TOOLS:

#1. Invicti

#2. Acunetix

#3. PortSwigger

#4. Detectify

#5. AppCheck Ltd

#6. Hdiv Security

#7. AppScan

#8. Checkmarx

#9. Rapid7

#10. MisterScanner

10) **Container Repository Scanner**:

A container scanner is an automated tool that analyzes these various container components to detect security vulnerabilities. Besides vulnerabilities introduced directly by the code and tool you add to an image, issues can originate from other images that your containers rely on.

Snyk is one of the best tool to integrate the Git repository, Artifact Repository and the Container Repository for scanning the complete Registry. It has a was integration feature as shown in the below image.

For References : Container Registry Scanning

Image1:

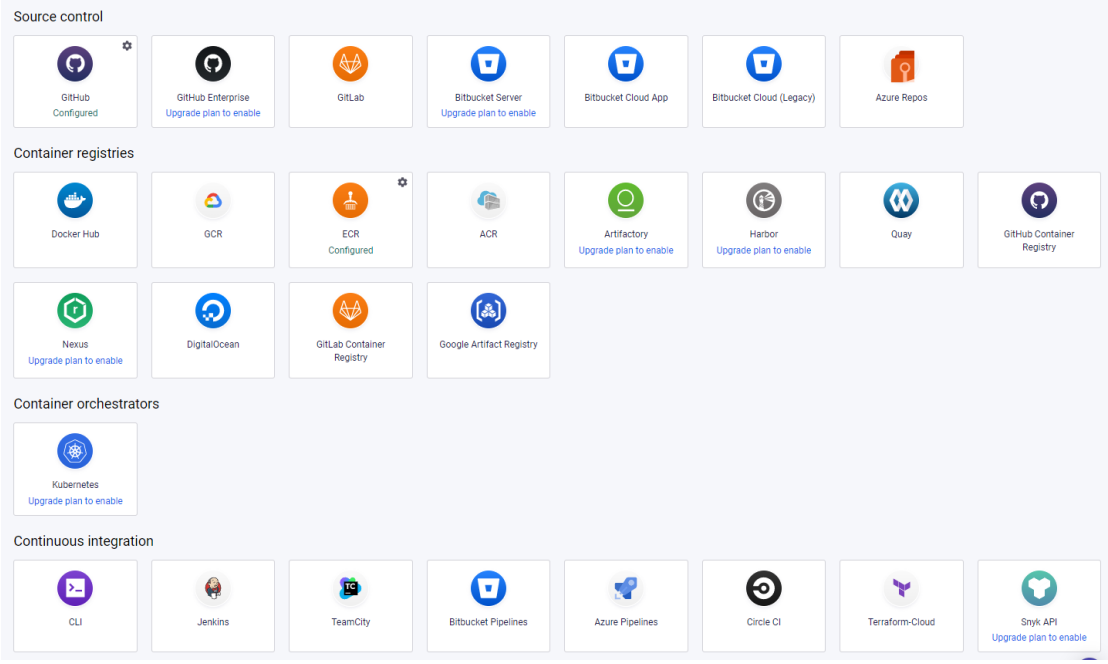
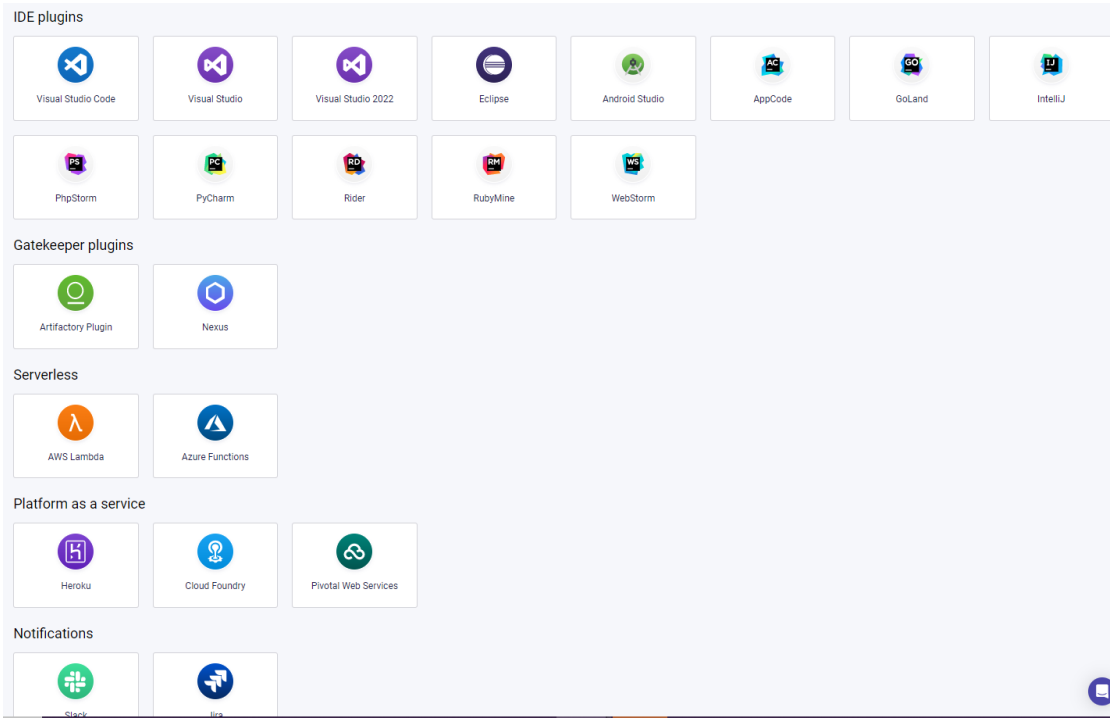


Image 2:



11) Static scan of Kubernete manifest file or Helm chart

It is always a good practice to scan your Kubernetes deployment or Helm chart before deploying. We can use **Checkov** to scans Kubernetes manifests and identifies security and configuration issues. It also supports Helm chart scanning. We can also use terrascan and kubeLinter to scan the Kubernetes manifest.

12) **Kubernetes CIS scan**

Kube-bench checks whether Kubernetes is deployed securely by running the checks documented in the Center for Internet Security (CIS) Kubernetes Benchmark. We can deploy kube-bench as a Job that runs daily and consume its report in CI/CD to pass or fail the pipeline based on the level of severity.

Example:

```
kubectl apply -f eks-job.yaml

kubectl logs kube-bench-pod-name
```

13)Trivy for Kubernetes:

Trivy can be extended with plugins and custom policies. For example, Aqua provides the kubectl plugin to better integrate Trivy with Kubectl. The plugin lets us scan images running in a Kubernetes pod or deployment.

```
$ trivy plugin install github.com/aquasecurity/trivy-plugin-kubectl

# Scan a pod $ trivy kubectl pod mypod

# Scan a deployment $ trivy kubectl trivy deployment mydeployment
```

14) Monitoring and Alerting

Monitoring and alerting is the process of collecting logs and metrics about everything happening in our infrastructure and sending notifications based on the metrics threshold value.

14.1) **Metrics monitoring**

Prometheus & Grafana: It's a widely used open source tool for metrics monitoring. It provides various exporters that can be used for monitoring systems or application metrics. We can also use Grafana to visualize prometheus metrics.

Nagios and Zabbix: These are open source software tools to monitor IT infrastructures such as networks, servers, virtual machines, and cloud services.

Sensu Go: It is a complete solution for monitoring and observability at scale.

14.2) **Log monitoring**

OpenSearch/Elasticsearch: It is a real-time distributed and analytic engine that helps in performing various kinds of search operations.

Graylog: It provides centralized log management functionality for collecting, storing, and analyzing data.

Grafana Loki: Grafana Loki is a lightweight log aggregation system designed to store and query logs from all your applications and infrastructure.

14.3) **Alerting**

Prometheus Alertmanager: The Alertmanager handles alerts sent by client applications such as the Prometheus server.

Slack : By creating a channel for all your projects, your teams, your offices, your departments – everything you're doing at work – you create a space for every conversation to happen in

Grafana OnCall: Developer-friendly incident response with phone calls, SMS, slack, and telegram notifications.

Security-focused logging and monitoring policy is used to prevent sensitive information from being logged in plain text. We can write a test case in our logging system to look for certain patterns of data. For example, a regex to find out sensitive information so that we can detect the logs in a lower environment.

Application performance Monitoring (APM) improves the visibility into a distributed microservices architecture. The APM data can help enhance software security by allowing a full view of an application. Distributed tracing tools like Zipkin and Jaeger kind of stitch all logs together and bring full visibility of requests from start to end. It speeds up response time for new bugs or attacks.

Although all cloud providers have their own monitoring toolsets and some tools are accessible from the marketplace. Also, there are paid monitoring tool providers like Newrelic, Datadog, Appdynamics, and Splunk that provide all types of monitoring.

15) **Server Vulnerability Scanning**:

Vulnerability scanning tools allow for the detection of vulnerabilities in applications using many ways. Code analysis vulnerability tools analyze coding bugs. Audit vulnerability tools can find well-known rootkits, backdoor, and trojans.

There are many vulnerability scanners available in the market. They can be free, paid, or open-source. Most of the free and open-source tools are available on GitHub. Deciding which tool to use depends on a few factors such as vulnerability type, budget, frequency of how often the tool is updated, etc.

Types of Vulnerability Scanners

Vulnerability scanners have their ways of doing jobs. We can classify the vulnerability scanners into four types based on how they operate.

1. Cloud-Based Vulnerability Scanners : Used to find vulnerabilities within cloud-based systems such as web applications, WordPress, and Joomla.

2. **Host-Based Vulnerability Scanners :** Used to find vulnerabilities on a single host or system such as an individual computer or a network device like a switch or core-router.

3. **Network-Based Vulnerability Scanners** : Used to find vulnerabilities in an internal network by scanning for open ports. Services running on open ports determined whether vulnerabilities exist or not with the help of the tool.

4. **Database-Based Vulnerability Scanners :** Used to find vulnerabilities in database management systems. Databases are the backbone of any system storing sensitive information. Vulnerability scanning is performed on database systems to prevent attacks like SQL Injection.

Tools for Server Vulnerability Scanning:

1. Nikto2
2. Netsparker
3. Nessus Professional
4. OpenVAS
5. W3AF
6. Arachni
7. Acunetix

8. Nmap
9. Nexpose

References Document:

1. SAST : https://www.mend.io/resources/blog/best-sast-tools/
2. SAST : https://theqalead.com/tools/best-static-application-security-testing-tool/
3. CheckOV: https://blog.opstree.com/2023/02/07/checkov-a-must-tool-for-infra-ci/
4. DGOSS : https://github.com/goss-org/goss/tree/master/extras/dgoss
5. DAST : https://www.softwaretestingmaterial.com/dast-software/
6. Server Vulnerability Scanning : https://phoenixnap.com/blog/vulnerability-assessment-scanning-tools