

DevSecOps Guideline Contents

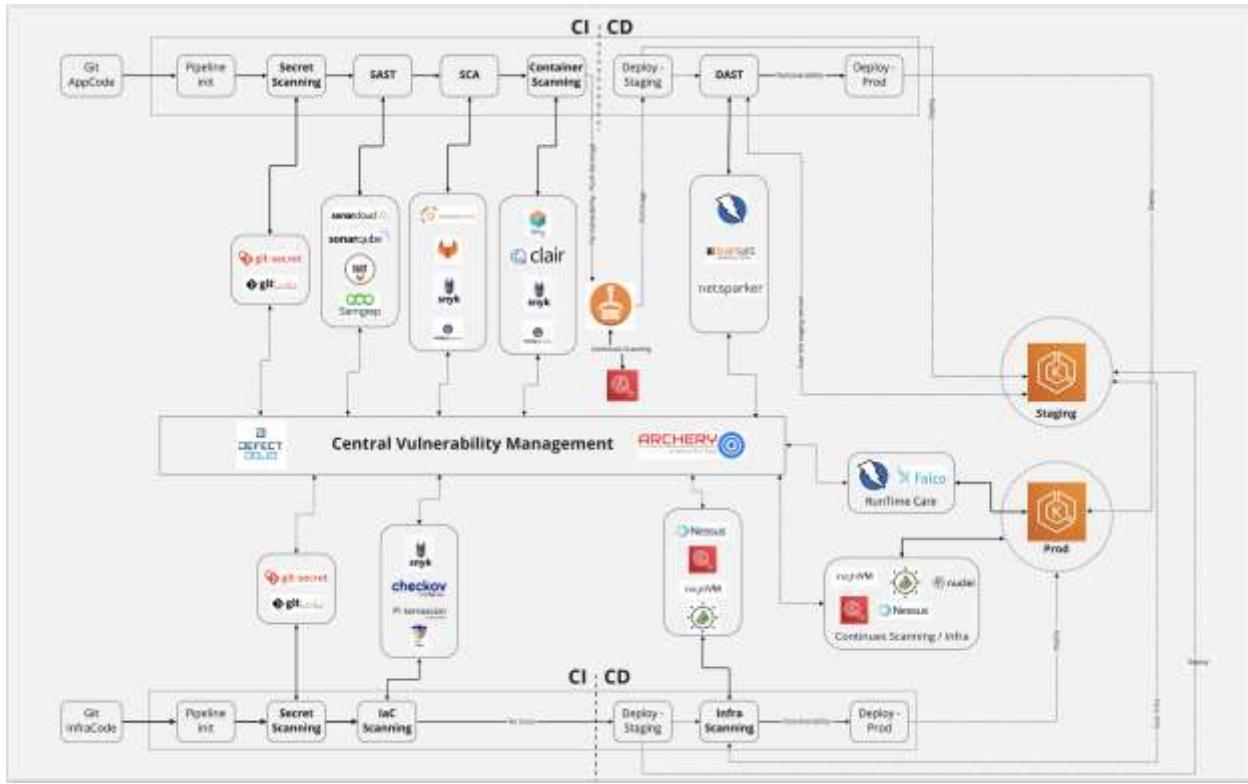
- [0 - Intro](#)
- [0-a - Overview](#)
- [0-b - Threat Modeling](#)
- [1 - Pre-commit](#)
- [1-a - Secrets Management](#)
- [1-b - Linting Code](#)
- [2 - Vulnerability Scanning](#)
- [2-a - Static Application Security Testing](#)
- [2-b - Dynamic Application Security Testing](#)
- [2-c - Interactive Application Security Testing](#)
- [2-d - Software Composition Analysis](#)
- [2-e - Infrastructure Vulnerability Scanning](#)
- [2-f - Container Vulnerability Scanning](#)
- [2-g - Privacy](#)
- [3 - Compliance Auditing](#)

Introduction to the OWASP DevSecOps Guideline

The OWASP DevSecOps Guideline explains how we can implement a secure pipeline and use best practices and introduce tools that we can use in this matter. Also, the project is trying to help us promote the shift-left security culture in our development process. This project helps any companies of each size that have a development pipeline or, in other words, have a DevOps pipeline. We try to draw a perspective of a secure DevOps pipeline during this project and then improve it based on our customized requirements.

The Ideal goal is “**detect security issues (by design or application vulnerability) as fast as possible.**”

Maybe the following picture can describe the goal of securing pipelines better. As you can see, we can add more steps in Dev pipelines and deliver products more secure and reliable to the products’ customers. We just put some open-source and commercial tools as an example in this image. As always, you are free to select tools and the exact location where you want to implement the tools. Besides implementing more security steps, Having a central vulnerabilities management solution can help to have a good view of the application security outlook in one picture.



DevSecOps Intro

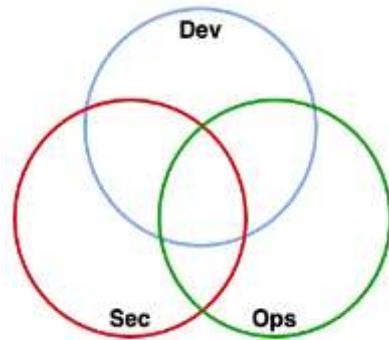
Today, DevOps is empowering any organizations to deploy changes to production environments at blazing rates. Since time to deliver is so important feature during this process, the main question for a security person is “How I can secure this process?” or “How much our deliverable products are secure?”. In this regard, we can embed some security-related steps entire our DevOps process to perform some automated tests. So considering the DevSecOps or secure DevOps culture helps us to promote the shift-left security strategy in our company, at least in the tech department.

[What's the Shift-left security strategy?](#)

As a simple definition, the shift-left security strategy is a way or solution to embedding security as a part of our development process and consider security from the inception steps of application or system design. In other words, security is responsible for everyone who works in the software development and operating process. Of course, security is a profession and we need highly skilled people to play security-related roles; but in this approach, any designer, software architect, developer, DevOps engineer, and ... together with security guys have liability about security.

Dev+Sec+Ops

Suppose that these 3 different areas for covering each other is something like the image, so in conclusion with the above words, we need to implement some tools and working on promoting a DevSecOps culture too.



As Shannon Lietz - founder at DevSecOps foundation - said:

The purpose and intent of DevSecOps is to build on the mindset that “everyone is responsible for security” with the goal of safely distributing security decisions at speed and scale to those who hold the highest level of context without sacrificing the safety required.”

The DevSecOps culture

As you heard before we want to talk about the Shift-left security. It means we should consider security from design (in a simple definition) which target is moving security earlier in the development process.

Let's suppose that you are working in a DevOps team and you are traditionally doing security test (yes, end of all QA tests and before going to production), what happens? Well, all bugs should be fixed ASAP, Developer team is under pressure to fix issues, QA tests should be performed again, and security test again. This means that the costs, money and time, increase. In the end, you sacrifice agility for security things do not like a business team.

The solution is introducing security earlier in the process instead of having it in the final steps. Considering security in design by threat modeling and break down huge security tests in smaller security testing and integrating them in the development pipeline.

The following picture shows the differences between DevOps and DevSecOps lifecycles.



Privacy

Privacy has become a major topic for companies of all sizes, since GDPR (Europe's General Data Protection Regulations), CCPA (California Consumer Privacy Act), LGPD – Brazil's Lei Geral de Proteção de Dados, and other laws and regulations are being enforced around the world.

Applications that will process a large volume of PII (personally identifiable information) should adapt the DevSecOps to follow the Privacy by Design approach, where the development process addresses privacy concerns thru the whole cycle.

Software testing strategies

When, we talk about testing we should have in mind we have different definitions in testing which can change our route to achieving a secure DevSecOps cycle. Let's take look into this definitions.

Different software testing strategies

1. Positive testing

Positive testing assumes that, under normal conditions and inputs, everything will behave as expected. It is performed with the assumption that only valid and relevant things will happen: data set and all other functionalities will be as expected.

1. Negative testing

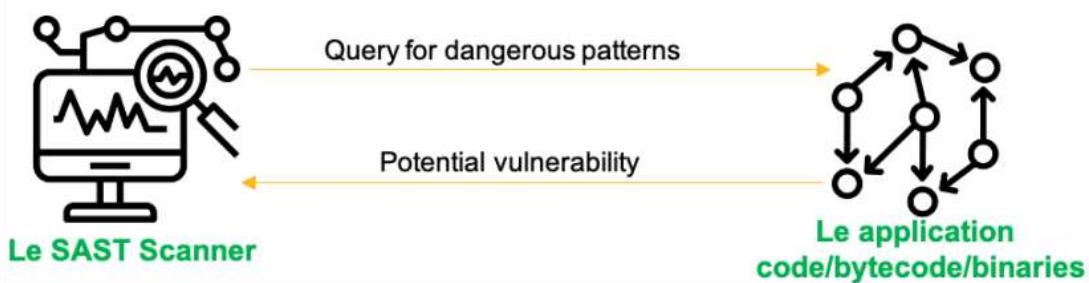
Negative testing checks the system behavior under unexpected conditions. Negative testing plays a much important role in high-performance software development: it checks the software behavior under unexpected conditions and inputs.

Methods of testing

1. Static testing

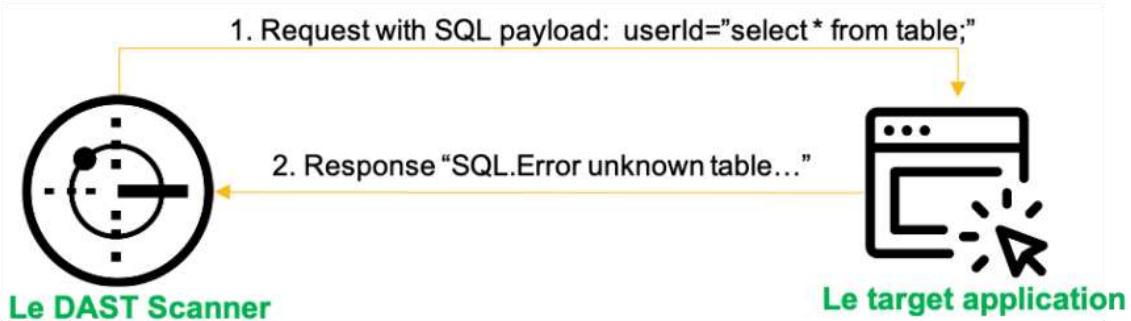
Static Testing checks software defects without executing the application code. It is performed in the early stage of development to avoid errors, as it is easier to find sources of failures and it can be fixed easily. Some issues that can't be found using Dynamic Testing, can be easily found by Static Testing. Such issues consist of hard coded credentials, deprecated encryption algorithms, 2nd order injections, weak random, etc. Most static analysis tools have the testing scope limited to one component and can not perform tests across different components. (EG. for a microservice architecture, static

analysis tools will test each microservice independently)



2. Dynamic testing

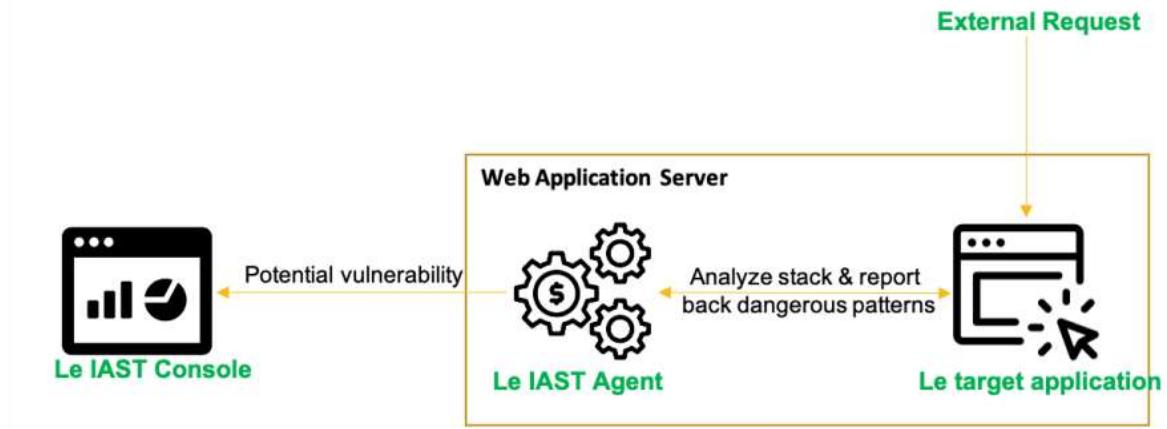
Dynamic Testing analyzes the behavior of the application code at runtime. Scanners send specially crafted requests to the target application. Request parameters are constantly modified during testing to try and expose a range of vulnerabilities. Based on the response of the application the tool can then identify potential vulnerabilities and report back. Some issues that can't be found by static analysis are easily detected by dynamic analysis. Such issues include client side vulnerabilities like authentication & session issues, sensitive data sent in plain text, etc. Dynamic analysis tools have the possibility of testing the entire application flow(multiple components at once). (Eg. for a microservice architecture, dynamic analysis tools can point to one microservice, but as they interact with each other results will represent the behaviour of the entire application)



3. Interactive analysis

Also known as Interactive Application Security Testig (IAST) monitors the application while other systems interact with it and observe vulnerabilities. This is achieved via sensors or agents deploy with the application. The sensors can see the entire flow from HTTP request down to the executed code, tracing the data through the application. Similar to static analysis, it can test one component at a time, but not multiple components. However, if agents/sensors are deployed on all components, when they interact with eachother this could reveal vulnerabilities in each component used in the application. (Eg. for a microservice architecture, only the microservices that have

agents/sensors attached will report vulnerabilities)



Threat Modeling

Since an important part of the DevSecOps initiative is the threat modeling and evaluating the risk of them. So in this part, we want to take look into this topic.

What is Threat Modeling?

That is a systematically listing all the potential ways one can attack an application. So Threat Modeling is a process for looking at attacks actively. The output of this process is a list of threats or probable threat scenarios also our approach should be Holistic to consider all threats not a specific part of an application. On the other hand Threat modeling is a **Collaborative** and **Repeatable** process.

Process Outputs:

- Diagrams
- Security requirements
- Non-requirements
- List of threats / vulnerabilities

Terms:

- **Weakness:**
A software defect or bug (Ex: missing user email validation)
- **Vulnerability:**
A weakness that can be exploited. (Ex: The user email field can be abused to insert SQL statement)
- **Attack**
 - **Target:** the value of an attack
 - **Attack Vector:** the path that attacker can take to exploit a vulnerability
 - **Threat Actor:** threat source.

- Attack Surface:
Anything that can be obtained, used, or attacked by a threat actor
- Risk:
 $\text{Risk} = \text{Impact} * \text{Likelihood}$

Why Threat Modeling?

As main reasons for using threat modeling we can say:

- Pro-active approach to finding threats
- Increasing efficiency by reducing cost
- A better prioritization based on bugs and mitigation plan
- A better understanding of the system

Who should the threat model?

- Architect, who knows how the application has been designed and how data flows across.
- Developer, who knows the elaborate details on how the application was built, the detailed interactions between components.
- Tester, who knows the requirements, and what the application is supposed to do.
- Security expert, who knows about specific attack factors and vulnerabilities. However, the best answer to this question is, **It depends on the organization.**

When to perform Threat modeling?

Since, the earlier you find vulnerabilities in software, the easier and cheaper it is to fix them, Threat Modeling should be as early as possible in the software design process. If you are working in an agile environment, ideally the threat modeling should be done during each sprint.

Select the Right approach & Methodology

Here we have to term 1st is Approach that describes how one could start with the process with threat modeling and 2nd is Methodology which describes the process itself. So each of the methodologies is based on one of the 3 available approaches.

Approaches

- **Asset-centric Approach:**
According to the name, it turns around assets. So the producing steps are as follows:
 1. Create a list of assets
 2. Draw assets, components and data flows
 3. For each element, check for threats

By repeating these steps for all assets, a list of threats will be produced.

- **Attacker-centric Approach:**
Starting threat modeling from an attacker perspective. So the producing steps are as follows:
 1. Create a list of threat actors
 - Motive
 - Means
 - Opportunity
 2. Create a list of threats
- **Application-centric Approach:** This one starts with visualizing the application instead of thinking about the risks or attacks first. So the producing steps are as follows:
 1. Draw a diagram of the application
 2. List threats for each element
 - STRIDE
 - OWASP TOP 10
 3. Rank threats using a classification model

Methodology

According to the approaches we have a list of methodology which are as follows:

- PASTA
- Microsoft Threat Modeling
- OCTAVE
- TRIKE
- VAST

Pre-commit

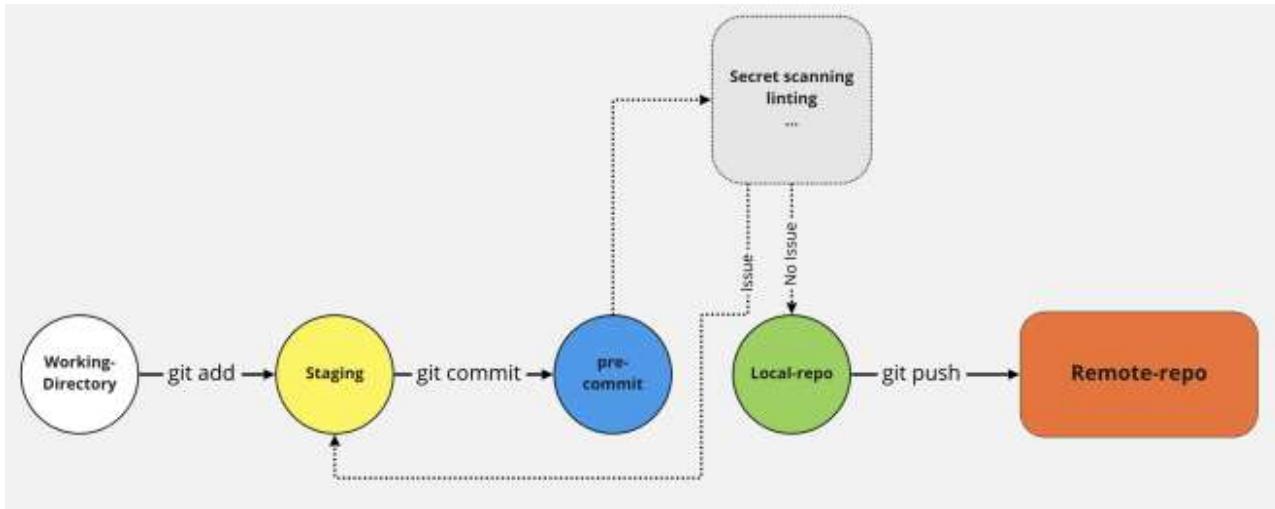
The Pre-commit fase is important because it can prevent security issues before they are submitted to a central (Git) repository.

Making sure that there are no secrets in the code, and that the code follows certain guidelines (According to the Linter rules) will result in a higher quality code.

In the following, we take a look into different types of pre-commit actions that are as follows:

1. Secrets Management
2. Linting Code

The following image can give you a better view of what the pre-commit means and why we must consider it.



Tools:

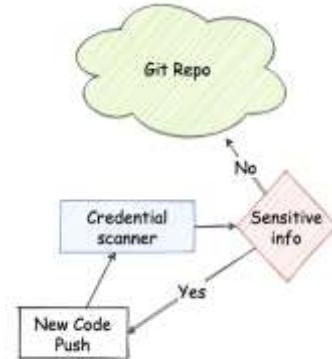
- [Pre-Commit](#) - A framework for managing and maintaining multi-language pre-commit hooks.

Take care secrets and credentials in repositories

How can you ensure that sensitive information are not pushed to a repository?

This is one of the [OWASP Top Ten issues](#) and several bug bounties write-ups are related to this kind of issue, eg hard-coded credentials pushed by mistake.

You should scan your commits and your repository, and detect any sensitive information such as password, secret key, confidential, etc. following the process shown in the picture.



The ideal approach is detecting and preventing the exposure of sensitive data before that they hit the repository, because they are then visible in the history. In case of code hosting platforms, secrets can still linger on the web and be searchable after you remove them from the repository.

A complimentary approach is scanning the repo for sensitive information, and then remove them; note that when a credential is leaked, it is already compromised and should be invalidated.

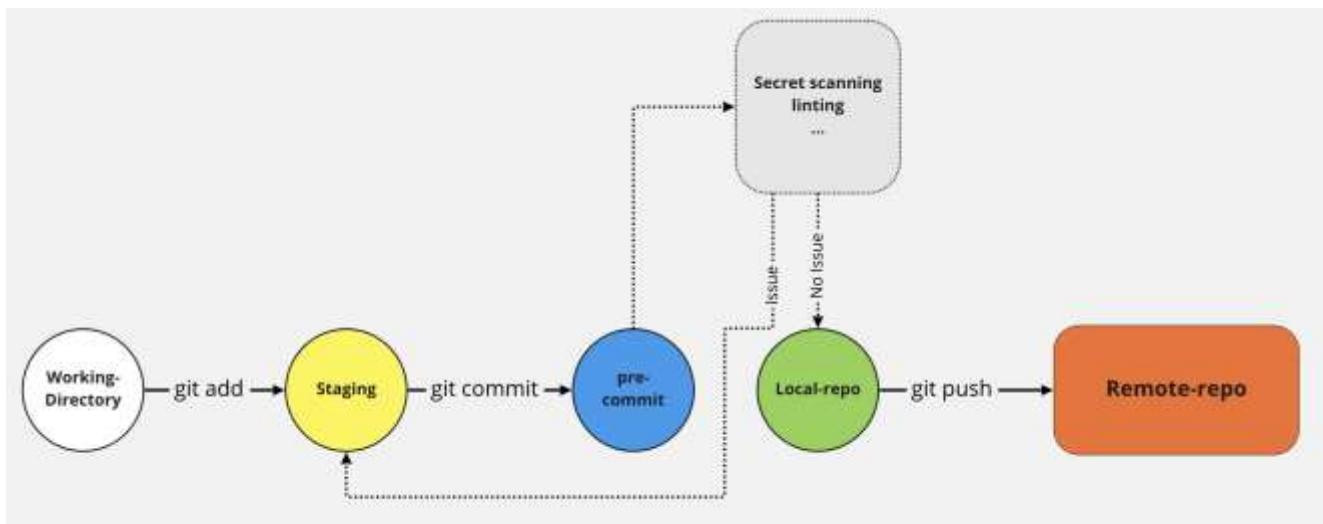
Detecting secrets in several locations

- **Detecting existing secrets** by searching in a repository for existing secrets.
- **Using Pre-commit hooks** in order to prevent secrets for entering our code base.
- **Detecting secrets in a pipeline**.

Why Detecting Secrets?

- The secrets should not be hardcoded.
- The secrets should not be unencrypted.
- The secrets should not be stored in source code.
- The code history does not contain inadvertent secrets.

Where and when to Detect Secrets?



Well, the best location is the **pre-commit** location, This ensure that before a secret actually enters your code base, it is intercepted, and the developer or committer gets a message. Another location is the build server or the **build** process. The build server retrieves source code, which is already committed and then it can analyze the source code where it contains new secrets or when it contains known secrets that the secrets are actually validated or audited.

Here are some helpful tools to automatically scan repositories for sensitive information. Scans can be implemented directly in our pipeline, and be repeatable and efficient.

Tools:

- [gitleaks](#) - Find sensitive information for a git repo
- [git-secrets](#) - Prevents you from committing secrets and credentials into git repositories
- [Repo-supervisor](#) - Scan your code for security misconfiguration, search for passwords and secrets
- [truffleHog](#) - Searches through git repositories for high entropy strings and secrets, digging deep into commit history
- [Git Hound](#) - Git plugin that prevents sensitive data from being committed

Linting Code

What Is Linting?

Linting is the automated checking of your source code for programmatic and stylistic errors. This is done by using a lint tool (otherwise known as linter). A lint tool is a basic static code analyzer.

What can Linting do?

- Linting can **detect errors** in a code and errors that can lead to a security vulnerabilities.
- Linters Can Also **detect formatting or styling issues** and makes the code more readable for more secure code.
- Linters can **suggest best practices**.
- Also they can **increases overall quality of the code**.
- Since everybody follows the same linting rules it **makes maintenance of code easier**.

Basic Lint Tools

Lint tools are the most basic form of static analysis. Using lint tools can be helpful for identifying common errors, such as:

- Indexing beyond arrays.
- Dereferencing null pointers.
- (Potentially) dangerous data type combinations.
- Unreachable code.
- Non-portable constructs.

Advanced Static Analysis Tools

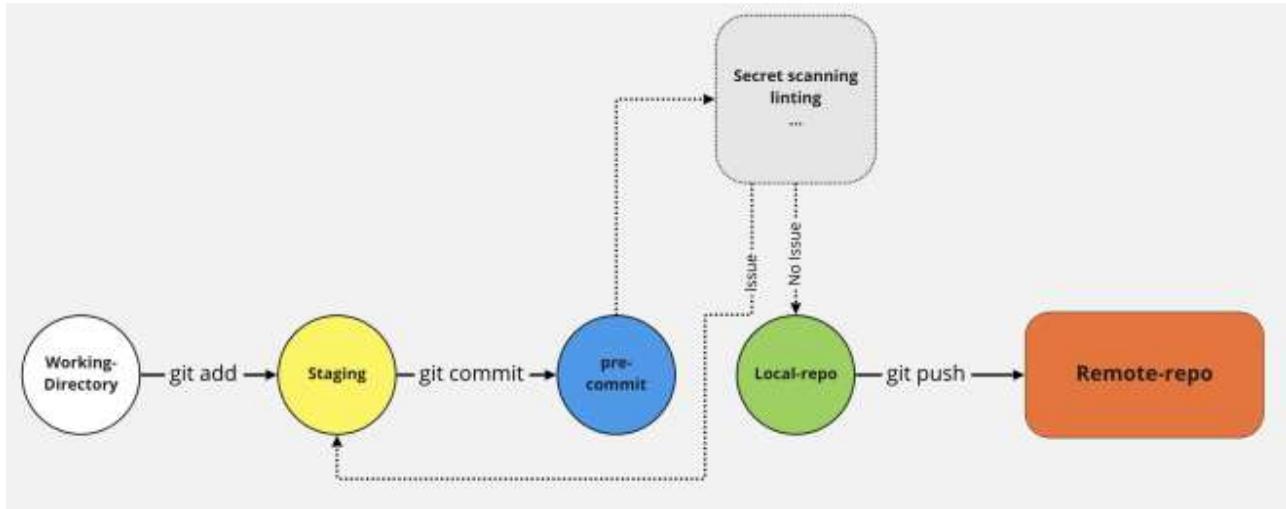
Advanced static analysis tools typically deliver:

- Pattern-based simulation.
- Quality and complexity metrics.
- Best practice recommendations for developers.
- Support for multiple safety and security-focused coding standards.
- Out-of-the-box certification for use in the development of safety-critical applications.

Issues with Linters

- Not every language has “quality” standard linter tools available, each framework usually has one or several linters.
- Different versions or configurations can lead to different results.
- Since some linters are very verbose and information overload can lead to focusing on “unimportant” issues.

Where and When to Use Linter



You can perform it in the **pre-commit** phase, so locally before actually committing code to your local repository to your local clone. Another phase where you often see linting is during the **build** phase, So here the build server pulls the code from the Git repository and performs linting on it and reports back that results from linting phase.

Vulnerability Scanning

Vulnerability scanning is an inspection of the potential points of exploit on a computer, application, endpoints, and IT infrastructure (including network) to identify security holes.

Performing vulnerability scanning is a common requirements for regulatory compliance and can help to minimize an organization's cybersecurity risk. An Approved Scanning Vendor (ASV), for example, is a service provider that is certified and authorized by the Payment Card Industry (PCI) to scan payment card networks.

In the following, we take a look into different types of vulnerability scanning that are as follows:

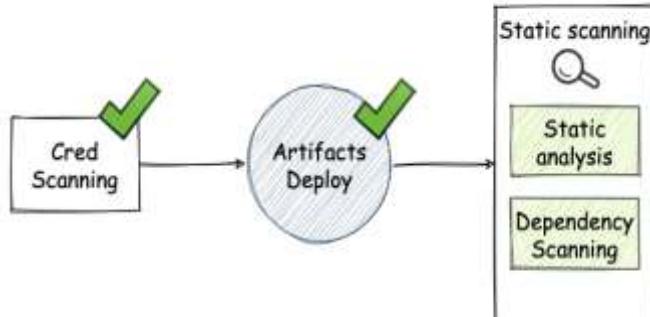
1. Static Application Security Test - SAST
2. Dynamic Application Security Test - DAST
3. Interactive Application Security Testing - IAST
4. Software Composition Analysis - SCA
5. Infrastructure Vulnerability Scanning
6. Container Vulnerability Scanning

Static scanning is an important part of the process!

Static Code Analysis or Source Code Analysis is usually part of a Code Review (white-box testing) and it is a method of computer program debugging that is done by examining the code without executing the program.

Static scanning is a good way of finding coding issues such as:

- Syntax violations
- Security vulnerabilities
- Programming errors
- Coding standard violations
- Undefined values



For more information about the Static Code Analysis please visit [the OWASP page](#). To achieve a better result we can combine static security scanning and 3rd party code (open-source libraries (dependency)) scanning. To do this better and more complete (prevent misconfigurations), here we can bring up IaC (Infrastructure as code) security scan too. For example check Terraform, helm, Ansible code, etc.

So according to the above lines the possible actions in this step are as follows:

- Static Code Analysis (known as SAST)
- Open-source libraries (3rd party / dependency) scanning (known as SCA)
- IaC Security scanning

Tools

- *Static Code Analysis:*
 - [SonarQube](#) - An open-source web-based tool, extending its coverage to more than 20 languages, and also allows a number of plugins
 - [Veracode](#) - A static analysis tool that is built on the SaaS model. This tool is mainly used to analyze the code from a security point of view
 - [security code scan](#) - Vulnerability Patterns Detector for C# and VB.NET
 - [Brakeman](#) - A static analysis security vulnerability scanner for Ruby on Rails applications
 - [Enlightn](#) - A static analysis vulnerability scanner for Laravel PHP applications
 - [Inquisition](#) - A set of tools for convenient technical analysis of web applications built with Ruby and Ruby on Rails. Now you don't need to set up and configure every single gem. Use Inquisition gem instead
 - [CodeSweep](#) - Static Analysis tool for GitHub that's free to use and can scan code on pull request. Support over 20 languages and IaC (docker, k8s). VS Code version can be found [here](#)

- [HCL AppScan on Cloud](#) - SAST tool built as a service. The tool can perform traditional SAST, SCA and IaC scanning.
 - [Semgrep](#) - Lightweight static analysis for many languages. Find bug variants with patterns that look like source code.
 - [Checkmarx SAST](#) - A static analysis security vulnerability scanner
 - [Fortify](#) - A static analysis security vulnerability scanner
- *IaC scanning:*
 - [Checkov](#) - Prevent cloud misconfigurations during build-time for Terraform, Cloudformation, Kubernetes, Serverless framework and other infrastructure-as-code-languages with Checkov by Bridgecrew
 - [ansible-lint](#) - Best practices checker for Ansible
 - [puppet-lint](#) - Check that your Puppet manifests conform to the style guide
 - [tfsec](#) - Security scanner for your Terraform code
 - [terrascan](#) - Detect compliance and security violations across Infrastructure as Code to mitigate risk before provisioning cloud native infrastructure
 - [tflint](#) - A Pluggable Terraform Linter

Dynamic Application Security Testing (DAST)

DAST is a “Black-Box” testing, can find security vulnerabilities and weaknesses in a running application by injecting malicious payloads to identify potential flaws that allow for attacks like SQL injections or cross-site scripting (XSS), etc. DAST tools are especially helpful for detecting:

- Input or output validation
- Authentication issues
- Server configuration mistakes

DAST tools allow for sophisticated scans on the client side and server side without needing the source code or the framework the application is built on. They usually require minimal user interactions once configured and can be run as part of a nightly scan. As more important DAST tools we can look at the following:

- Dynamic security scanner
- Fuzzers
- Attack Proxies

Tools

- *Open-source:*
 - [ZED Attack Proxy](#) - It is an open source tool which is offered by OWASP for performing security testing
- *Commercial:*
 - [Acunetix](#) - An automatic web security testing scanner that accurately scans and audits all web applications, including HTML5, JavaScript and Single Page applications (SPAs)

- [Netsparker](#) - It can identify vulnerabilities in all types of modern web applications, regardless of the underlying architecture or platform
- [Escape DAST](#) - Escape DAST is purposely built for testing for business logic vulnerabilities and handling complex auth scenarios in modern applications: APIs (including GraphQL) and Single Page Apps (SPAs)
- [InsightAppSec \(AppSpider\)](#) - Application security testing for the modern web
- [Veracode Dynamic Analysis](#) - Veracode Dynamic Analysis helps companies scan their web applications for exploitable vulnerabilities at scale
- [Burp Suite](#) is an integrated platform for performing security testing of web applications. Its various tools work seamlessly together to support the entire testing process, from initial mapping and analysis of an application's attack surface, through to finding and exploiting security vulnerabilities.
- [HCL AppScan on Cloud](#) - DAST tool built as a service. It can scan both public and privately hosted application. Can explore and test modern web applications, leverage manually recorded steps and handle complex login scenarios.
- [Nuclei](#) - Fast and customisable vulnerability scanner based on simple YAML based DSL.

Interactive Application Security Testing

IAST (interactive application security testing) is an application security testing method that tests the application while the app is run by an automated test, human tester, or any activity “interacting” with the application functionality.

The core of an IAST tool is sensor modules, software libraries included in the application code. These sensor modules keep track of application behavior while the interactive tests are running. If a vulnerability is detected, an alert will be sent.

The process and feedback are done in real time in your integrated development environment (IDE), continuous integration (CI) environment, or quality assurance, or while in production. The sensors have access to:

- Entire code
- Dataflow and control flow
- System configuration data
- Web components
- Back-end connection data

Examples of such vulnerabilities could be hardcoding API keys in cleartext, not sanitizing your users inputs, or using connections without SSL encryption.

IAST vs SAST

Static Application Security Testing method examine source code in a non-runtime environment early in the SDLC. They look for suspicious code patterns that indicate security risks. Even though they are easy to deploy, SASTs throw too many false positives because

SASTs do not take into account the presence of other security countermeasures, and they lack visibility during runtime. SAST tools normally run inside the IDE as part of the compilation phase, and introduce delays as the scan process takes time to finish. IASTs are more flexible than SASTs, because they are applicable in production runtime environments (SASTs require direct access to the source code).

IAST vs DAST

Dynamic Application Security Testing method works like a black-box scanner that executes requests against the application to find security issues. DASTs look at the applications from the exterior and determine the presence of risks by looking at the response (including body and headers) of the server to a battery of tests, but DASTs have no visibility of the internal workings of the app. Furthermore, DAST tests are hard to automate, because DASTs must be operated by experienced appsec teams, such as penetration testers, to be truly useful. Forrester estimates that the duration of a DAST scan can take around 5 to 7 days, while testing with IAST is a real-time (zero minutes) operation.

Tools

- [Contrast Community Edition \(CE\)](#)
- [Checkmarx Interactive Application Security Testing\(CxIAST\)](#)
- [Seeker Interactive Application Security Testing](#)
- [HCL AppScan on Cloud](#)

Software Component/Composition Analysis (SCA)

Component Analysis is the process of automating application security for managing third-party and open source components of codebase. SCA will find any potential vulnerable components in our codebase to prevent high security risks like **Supply-Chain Attack**, not only that but also provide licensing about each components. By doing this, it helps organization to reduce security risks in their codebase libraries and needed to be early in modern software development life cycle.

For more information about the Component Analysis please visit [the OWASP page](#)

We should put the Component Analysis earlier, before security testing like SAST, DAST to prevent any vulnerable libraries pushed to live environment (Production) and implemented Continuous Monitoring of its libraries to reduce Supply Chain Attack risk rapidly.

Tools

- *Open-source:*

- [OWASP Dependency-check](#) - Software Composition Analysis (SCA) tool that attempts to detect publicly disclosed vulnerabilities contained within a project's dependencies and it supports Java, .NET, JavaScript, Ruby
 - [RetireJS](#) - JavaScript-specific dependency checker
 - [Safety](#) - Python dependency checker for known security vulnerabilities
 - [bundler-audit](#) - Patch-level verification for Bundler (Auditing Ruby 3rd party libs versions)
- *Commercial:*
 - [Hakiri](#) - A commercial tool that offers dependency checking for Ruby and Rails-based GitHub projects using static code analysis
 - [HCL AppScan on Cloud](#) - SAST tool built as a service that can perform both SAST, SCA & IaC at the same time.
 - [Snyk](#) - SCA tool offer as a SaaS solution.
 - [WhiteSource](#) - WhiteSource identifies every open source component in your software, including dependencies. It then secures you from vulnerabilities and enforces license policies throughout the software development lifecycle.
 - [Synopsys BlackDuck](#) - Black Duck automated policy management allows you to define policies for open source use, security risk, and license compliance up front, and automate enforcement across the software development life cycle (SDLC).

Infrastructure Vulnerability Scanning

DevOps does a great job in automating the development and deployment process, but since all moving parts (containers, libraries etc.) are being updated frequently, it is imperative to make sure the infrastructure where you deploy your code is safe.

The best way to do that is to incorporate vulnerability scanning into your pipeline.

A vulnerability scanner is a computer program designed to assess computers, networks or applications for known weaknesses. These scanners are used to discover the weaknesses of a given system. They are utilized in the identification and detection of vulnerabilities arising from mis-configurations or flawed programming within a network-based asset such as a firewall, router, web server, application server, etc. Modern vulnerability scanners allow for both authenticated and unauthenticated scans. Modern scanners are typically available as SaaS (Software as a service); provided over the internet and delivered as a web application. The modern vulnerability scanner often has the ability to customize vulnerability reports as well as the installed software, open ports, certificates and other host information that can be queried as part of its workflow.

- **Authenticated scans** allow for the scanner to directly access network based assets using remote administrative protocols such as secure shell (SSH) or remote desktop protocol (RDP) and authenticate using provided system credentials. This allows the vulnerability scanner to access low-level data, such as specific services and configuration details of the host operating system. It's then able to provide detailed and accurate information about the operating system and installed software, including configuration issues and missing security patches.

- **Unauthenticated scans** is a method that can result in a high number of false positives and is unable to provide detailed information about the assets operating system and installed software. This method is typically used by threat actors or security analyst trying determine the security posture of externally accessible assets.
-

Tools

- [SAINT \(Security Administrator's Integrated Network Tool\)](#) is computer software used for scanning computer networks for security vulnerabilities, and exploiting found vulnerabilities.
- [Nessus](#) scans cover a wide range of technologies including operating systems, network devices, hypervisors, databases, web servers, and critical infrastructure.
- [Arachni](#) is a Web Application Security Scanner Network that could be used to scan the vulnerability such a XSS, SQL Injection, NoSQL Injection, Code Injection, File Inclusion variants, etc.
- [Burp Suite](#) is an integrated platform for performing security testing of web applications. Its various tools work seamlessly together to support the entire testing process, from initial mapping and analysis of an application's attack surface, through to finding and exploiting security vulnerabilities.
- [Nexpose](#) is one of the main vulnerability evaluation tools which provides a simple to-utilize, useful dashboard, and, as the vast majority of the tools we have seen, it bolsters a broad scope of consistency reporting, including PCI compliance.
- [Nikto](#) is a free software command-line vulnerability scanner that scans webservers for dangerous files/CGIs, outdated server software and other problems.

Container Vulnerability Scanning

As containers become an almost ubiquitous method of packaging and deploying applications, the instances of malware have increased. Securing containers is now a top priority for DevOps engineers. Fortunately, a number of open source programs are available that scan containers and container images. Let's look at five such tools.

What can Container Security Scanning do?

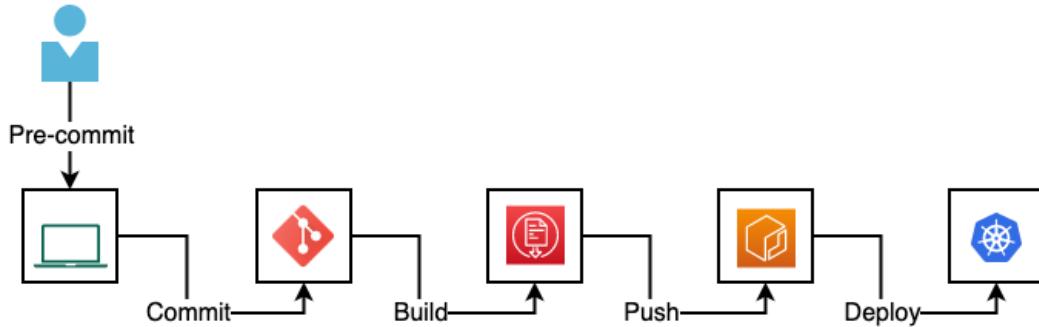
- Detect insecure containers
 - Detect outdated libraries
 - Detect incorrectly configured containers
 - Detect outdated operating system
- Detect compliance validations
- Suggest best practices

Issues with Container Security Scanner

- Level of depth depends on tool being used, So the results that you'll get are very dependent on the type of tool you choose.

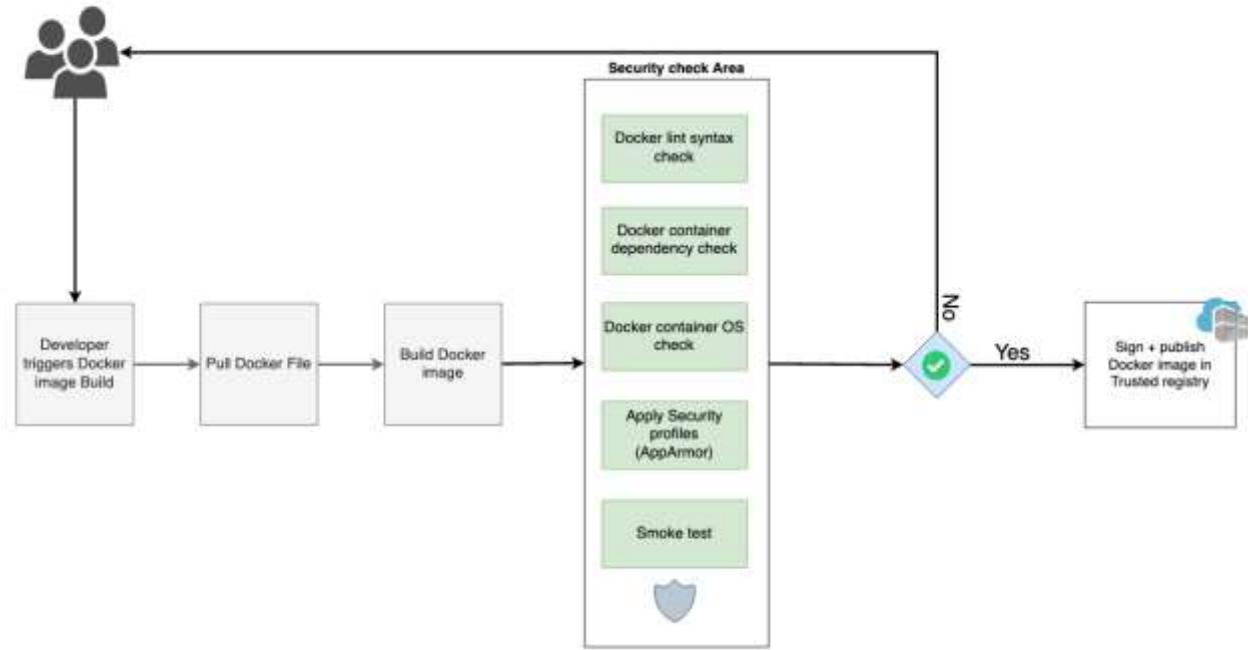
- Easy to go “too far” with configuration, there are tools where you can configure so much different settings, that it’s easy to jump overboard.
- The scan results will lead to actionable events?

Where and When to use Container Scanner?



You can use it at the build phase when you’re actually building for instance a Dockerfile and looking at the resulting image that you’re creating. Another location to perform container scanning would be when you push a container to the registry or when you pull a container from a registry. However, a good approach is scanning before pushing into a trusted container registry then you can say we have a container registry with a scanned version of all images and for deploying in production you can pull from this trusted container registry. (Please take a look into the following image)

Container Security - pipeline



Tools:

- [Clair](#) - Vulnerability Static Analysis for Containers
- [Anchore](#) - Open-source project for deep analysis of docker images
- [Dagda](#) - A tool to perform static analysis of known vulnerabilities, trojans, viruses, malware & other malicious threats in docker images/containers and to monitor the docker daemon and running docker containers for detecting anomalous activities
- [Falco](#) - Falco, the cloud-native runtime security project, is the de facto Kubernetes threat detection engine
- [Harbor](#) - Harbor is an open source registry that secures artifacts with policies and role-based access control, ensures images are scanned and free from vulnerabilities, and signs images as trusted.
- [Trivy](#) - Trivy is a simple and comprehensive vulnerability/misconfiguration scanner for containers and other artifacts.

Privacy

Privacy has become an important aspect of application security. GDPR, LGPD, CCPA and other laws and regulations have imposed several controls over processing PII (Personally Identifiable Information). To comply with those regulations, DevSecOps have to be sure to use PII accordingly and protect data against leaking.

To start, it is important to have a good understanding of what is considered PII:

- First and last name;
- Identifiable email (name.lastname@domain.com);
- Identity card numbers;
- Location data (mobile);
- IP Address.
- This is not a complete list.

Some PII are considered sensitive, and require even more protection, such as:

- Racial or ethnic origin;
- Political opinions;
- Religious or philosophical beliefs;
- Trade union membership;
- Genetic data; and
- Biometric data (where processed to uniquely identify someone).

Most privacy regulations promote Privacy by Design, the approach where the development process addresses privacy concerns thru the whole cycle. Even before coding starts.

You have to create a PII data flow to make sure you apply the required protection to the data thru its lifecycle. You also have to follow all the requirements related to the quality and volume of processed data.

All PII data processing requirements have to be specified. you have to create an inventory of all PII and evaluate the processing activity to make sure it follows all requirements, such as:

- Lawfulness, fairness and transparency
- Purpose limitation
- Data minimization
- Accuracy
- Storage limitation
- Integrity and confidentiality (security)
- Accountability

After you make sure that the PII and the PII processing activity are mapped and comply to the privacy regulation you have to follow, it is important to apply the security measures to protect the data, accordingly with its criticity.

Compliance Auditing

TBD