



Application Security Assessment

Gustavo Vázquez & Roger Carhuatocto
2021/10/30

1. Getting the context



- Review of the system from a vulnerability standpoint.

I don't know what are the vulnerabilities. A feature today, tomorrow may have a vulnerability. So we better talk about **Threats**.

Has the application been built with security in mind? Whatever your answer, we need to dig into technical details.

- Technical Documentation:
 - Architecture (Infra, App, DB, ...), Functionalities, Use Cases, Stack, CI/CD approach and Tooling, etc.
- Current Security Plan: Threats, Risks and Security Controls.
- Tools recommended:
 - ❑ Threat Modelling.
 - ❑ OWASP Security by Design Principles.
 - ❑ The Twelve Factors.
 - ❑ OWASP Top 10.
 - ❑ MITRE ATT&CK.
 - ❑ **Shift-Left Testing.**
 - ❑ **Pareto Principle.**

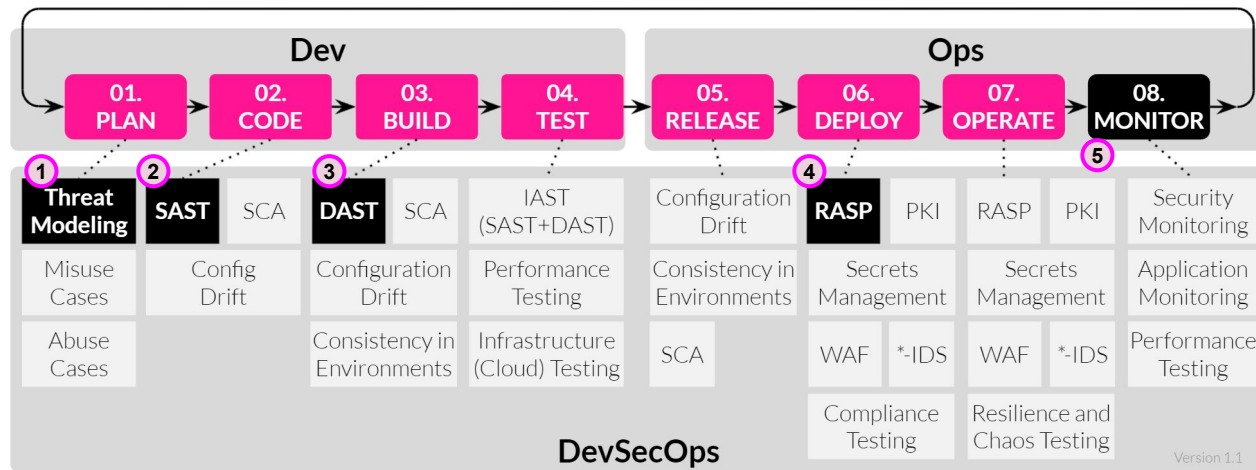
2. Single Source of Truth (Criteria)



- Definition of criteria for classification/ranking of vulnerabilities.
- Based on the Threats Modelling Tool criteria
- Based on MITRE ATT&CK
- Based on OWASP Top 10
- Based on Twelve-Factors App

3. The Assessment

- The MVP is conducting (1) Threat Modelling, (2) SAST and (3) DAST.



SAST.

Scan the code statically to find vulnerabilities. Can be automated and triggered from CI Server or Source Code Server. Moderns SAST includes SCA. Can be implemented in few months and be integrated into your CI/CD workflow. The difficult part is tuning it to avoid false positives and train to developers to handle the findings.

Threat Modelling.

It's a continuous process. It helps to explore all use cases, misuse cases, abuse case. With all use case we can elaborate a Security Test Master Plan to be implemented and executed in Pentesting and/or DAST continuously.

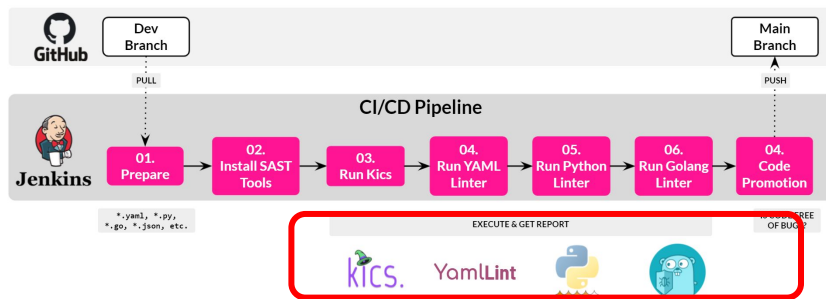
Test use cases and Threats to analyze












DAST.

Test the security of your Application. The tool can be deployed and configured easily (weeks), the hard part is define the Use Cases (dependency with Threat Modelling and QA) to assess the critical Application Features. Can not be automated fully.

4. Conducting an Assessment

- Static Application Security Testing (SAST)

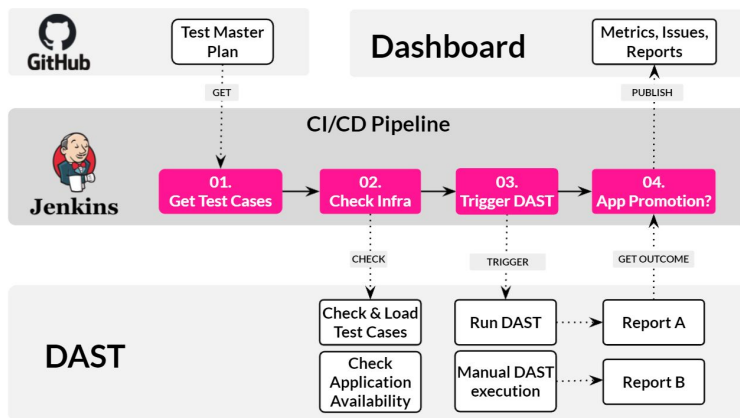










Name		Category
Semgrep		SAST
Anchore Grype		SAST
AquaSec Trivy		SAST
AWS CloudFormation Linter		SAST
Bandit (Python)		SAST
Checkmarx KICS		SAST
Checkov		SAST
Clair		SAST
Dockle		SAST
GolangCI-Lint		SAST
GoSec		SAST

Name		Category
huskyCI		SAST
PHPStan		SAST
Pylint (Python)		SAST
Skyscanner CFripper		SAST
SonarQube CE		SAST
SpotBugs (Java)		SAST
Stelligent cfn_nag		SAST
tfsec (Terraform)		SAST
AWS Serverless Rules		SAST
Dagda		SAST
Terraform Linter (tflint)		SAST

4. Conducting an Assessment

- Dynamic Application Security Testing (DAST)



Name		Category
Nuclei	 nuclei	DAST
Burp Suite CE		DAST
AquaSec Kuber-Hunter	 kube-hunter	DAST
Arachni	 arachni	DAST
OpenSCAP	 OpenSCAP	DAST
OWASP Zed Attack Proxy (ZAP)	 ZAP	DAST
w3af	 w3af	DAST
Wapiti	 Wapiti	DAST
Grabber		DAST