Altimetrik

Operationalizing DevSecOps:

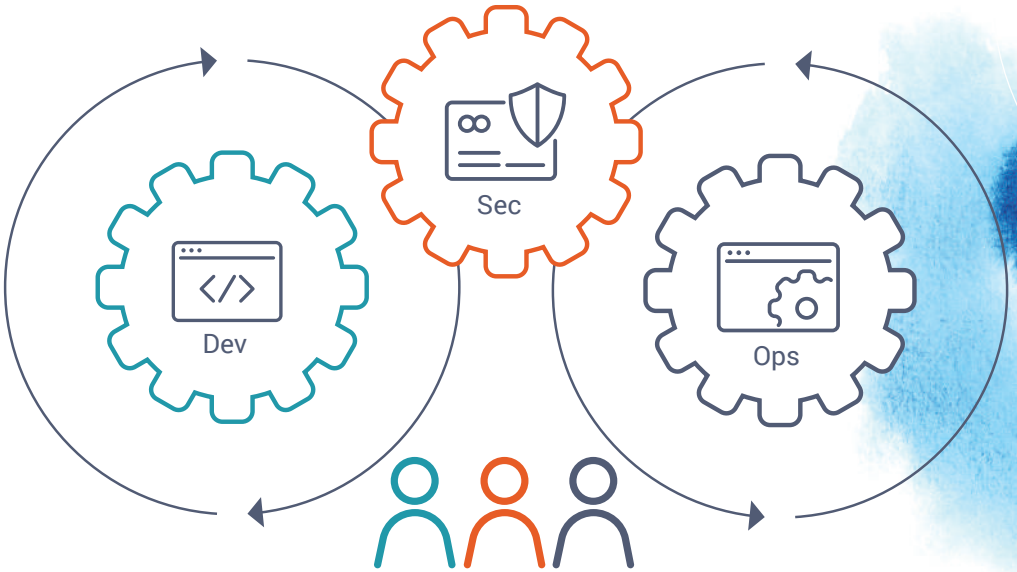
How to Overcome the Challenges and Get Started

DevSecOps can reduce **80%** of production application security vulnerabilities and related incidents

Table of Contents:

1. Introduction

DevOps standardizes and automates continuous code integration, delivery, and deployment workflow orchestration. As we know, consistency in development, test and deployment are key to avoid mishaps and failures. DevOps is all about improving collaboration between development, release engineering, quality assurance, and operations teams. DevOps helps improve collaboration and communication among teams and deliver applications at a faster pace without sacrificing quality. Microservice architecture or modular application architecture is also one of the key factors for the successful implementation of fast-paced DevOps, where each component can be developed, revised, and deployed on its own, and any issues within an individual project have only a minor impact on the entire software project. All the modular components or microservices are integrated using APIs.



Traditionally, a security engineering team handles application security testing, but in many enterprises, security is an afterthought. Security testing starts when an application is ready to go to production and majority of the issues found cannot be fixed before it is deployed. The goal of DevSecOps is to ensure that you've identified majority of application security vulnerabilities before deploying code to production, without compromising the speed or quality.

In this paper, we will discuss in detail the challenges involved in operationalizing DevSecOps, integrating application security testing into DevSecOps, and how to systematically approach DevSecOps maturity. With infrastructure as a code and policy as a code, the scope of security code is no longer limited to application code. DevSecOps is a slow process and it requires good planning, CI automation with security test requirements, and redefining the roles of developers, security engineers, and operations engineers.

To ensure consistent use of DevSecOps, it is important to define or enhance enterprise security policies and security exception waiver policies, as we know many applications will be released into production with known security issues. Many factors contribute to these issues, including third-party libraries, ineffective mitigation, vulnerabilities highly likely to be exploited vs. vulnerabilities inaccessibility from the external world, and in some cases, a rush to meet urgent business needs.

This paper shares what we've experienced and learned while deploying DevSecOps at scale. We will also share best practices for security policies, security gates, effective tools, and a case study on effective implementation and operationalizing of DevSecOps at scale.

2. What is DevSecOps?

DevSecOps is an extension of DevOps that leverages DevOps workflow automation and efficiencies for application security testing of every code release to production. DevSecOps brings consistency to security testing and prioritizes it at the same level as functional testing.

The workflow automation and orchestration enables a culture of close collaboration and communication among application security teams, development teams, operations teams, and network and infrastructure security teams.

DevSecOps helps identify security vulnerabilities early in the development cycle not only for application security but for infrastructure and network security as well, as validation tools are integrated with code deployment. The security tools required for application security are different from those for network and infrastructure security. DevSecOps is the most effective and economical way to reduce production security incidents.

While actual implementations will vary, DevOps generally includes four stages, each of which integrates security tools and workflow. DevSecOps embeds security testing in each stage of DevOps:

Plan

- Threat Model
- Security Requirements
- Security Training
- On boarding to Security Tools

Build

- Static code analysis (SAST)
- Open source vuln scan
- Container vuln scans

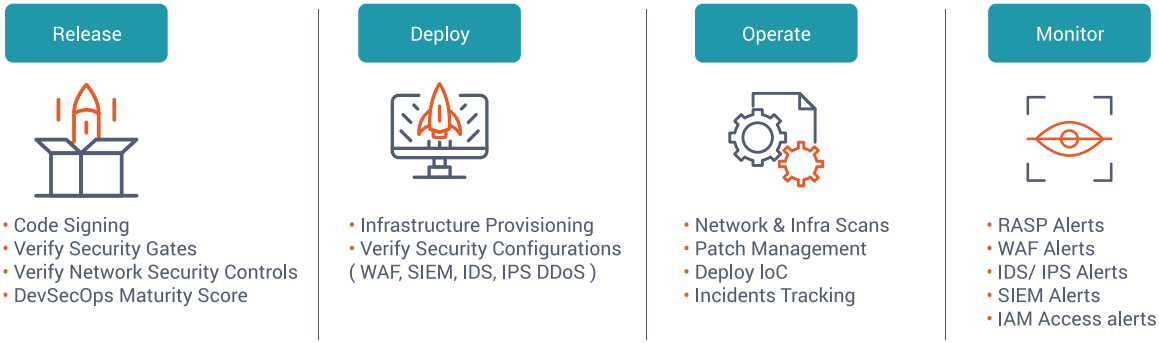
Test

- Dynamic AppSec Testing (DAST)
- Interactive AppSec Testing(IAST)

Deploy

- Interactive AppSec Testing(IAST)

DevSecOps also adds security controls and configuration checks during deployment:



3. Challenges in Implementing DevSecOps

Adopting DevSecOps is a slow process. Organizations can easily spend one to two years reaching full maturity, depending on their enterprise's focus, funding, and strategic vision. One of the major challenges involved in operationalizing DevSecOps is matching the speed with which applications are developed and released using agile with DevOps. In some cases, applications are released on a daily basis in majority of other cases agile scrums typically lasting two weeks for a code release.

One of the key challenges in successfully adopting DevSecOps is to identify the high-risk changes versus low-risk changes on a continuous and near real-time basis. These might be development changes—i.e., code related—or operational changes—i.e., incidents or known network and infrastructure security issues.

Beyond that, here are five common - challenges organizations experience when adopting DevOps:

1. **Traditional security testing tools and processes are designed for the waterfall model.** Adopting DevSecOps requires reviewing and updating your existing application security policies and security requirements. You should consider potential exceptions and scenarios where the applications dev team has late discovery due to a security tool upgrade or late code change, but delaying the release is not an option. It is common to find previously unreported issues for the same code base after a security tools upgrade.

2. **Most security engineers are trained for manual security testing.** Application security engineers and secure software development life cycle (SSDLC) engineers usually come with a penetration testing background, network security background, or software testing/quality assurance (QA) background. They typically have knowledge of secure coding practices, OWASP, or SAN25 application security vulnerabilities and ways to fix them. But they don't usually have software development knowledge. It is challenging for such security teams to adopt to DevSecOps, which relies on automated security testing and shifts much of the responsibility for secure coding and testing to the development and quality assurance teams. Application security engineers are still needed, however, as they need to make sure false positive rates remain low, provide guidance to developers on fixing security issues, and offer continuous training on application security testing, threat modeling, and secure coding practices.

3. **Developers and quality assurance engineers are not trained for secure coding and security testing.** The success of DevSecOps relies on developers adopting secure coding practices; using security testing tools, such as static code analysis, as early in the process as possible; and continuously testing and fixing the security issues on a daily basis. Interactive Application Security Testing (IAST) relies on QA test coverage. If the QA test coverage is low, then security testing will be incomplete. This is one of the major challenges with IAST.

4. **Traditional security tools lack automation support.** Adopting DevSecOps requires careful evaluation of security tools to ensure they support automation and integration with Jenkins, Bitbucket, and Jira. Allow your team to define rules that mark particular types of defects to suppress and no longer report if marked as false positives. Consider the programming languages you're using, and look for tools that keep false positive rates low, and allow you to easily configure rules to detect and suppress them. Tools required for application security testing are distinct from the tools required for security configuration and monitoring checks.

5. **Absence of security gates, security testing enforcement, and exceptions.** This is one of the common sources of DevSecOps mishaps and failures, as there's no clear definition or automation for which release has to go through which type of application security testing. This creates ambiguity and confusion with the development and operations teams. The security policy should also consider sensitive applications in the scope of PCI or HIPPA, as some of them may require manual review and approval for critical releases.

4. Application Security Testing Workflow

Application security testing is usually divided into the following areas:

- **Security architecture review is one** of the critical steps to understanding threat vectors, and identifying design and implementation issues as early in the process as possible.
  Security architecture review is usually done for new applications or a major change in key design areas, such as Identity Access Management (IAM), data protection and privacy, API security, and connectivity and data sharing with external partners. The security threats identified during security architecture review are converted to security requirements to prevent those threats. The implementation of these security requirements must be verified during source code review as part of SAST.

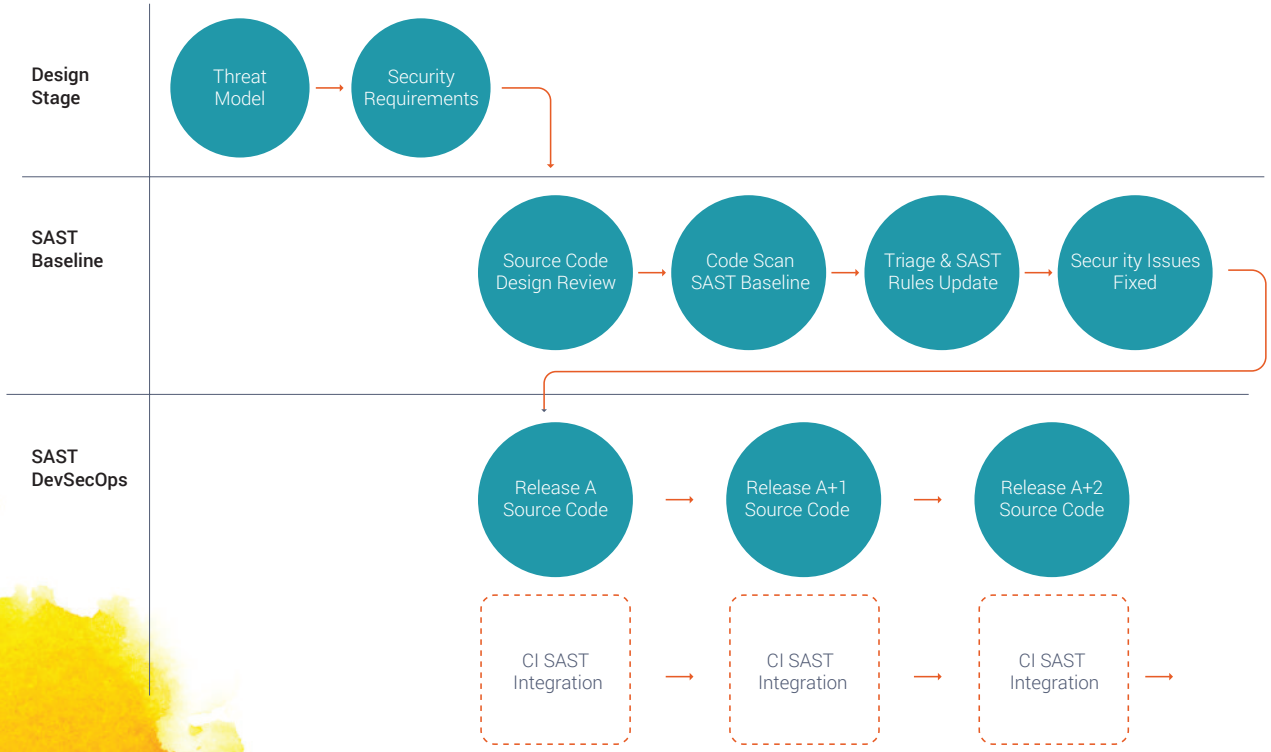
- **Static Application Security Testing (SAST)** involves source code security scans using automated tools. Source code security testing is also referred to as static code analysis. Integrating SAST in DevSecOps requires good planning and training to developers on project naming convention, identifying release scans, ways to mark false positive or when requires support from security engineers. SAST can generate hundreds to thousands of findings, depending on the size of applications. Hence baseline for SAST is an important step to be performed before integrating with DevSecOps automation, to ensure smooth onboarding to the DevSecOps process.

  Source code design and implementation review is important to identifying critical parts of the code because SAST will not be able to identify the following security vulnerabilities:
  - Any issues related Identity Access Management for authentication and authorization, Password change and Password reset flow etc.
  - Data encryption/decryption algorithms and key security implementation for data in use and data protection in transit
  - API authentication and authorization
  - Data masking and protection at rest

Project baselining to SAST

When you introduce DevSecOps for existing applications, you need source code baselining. This process scans all of the application source code to identify false positives, then updates the rules to suppress the false positives if there are input sanitizers outside of the source code—e.g., a third-party library that needs to be configured. Based on the lines of code and number of modules involved, this could be time-consuming and require careful review and triaging of all the security issues by security engineers in collaboration with developers.

Integrating SAST with Continuous Integration (CI) and scaling to a large number of applications is challenging, especially when those applications have not gone through SAST using the same tool.



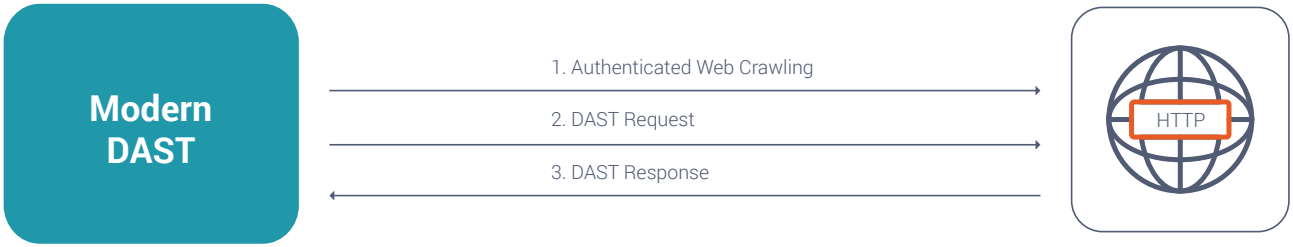
- **Dynamic Application Security Testing (DAST) / Interactive Application Security Testing (IAST)** is used during the QA or testing phase. The integration with DevSecOps depends on the type of application and type of tools.

  - **Traditional DAST** is configured as a reverse proxy during QA testing. DAST captures all QA test request and response calls and then generates new tests by modifying request payloads for security testing. Some of the drawbacks are:

    - Configuration and setup require custom automation, aren't easy for developers to set up, and may require a security engineer to set up and configure
    - High false-positive rate
    - Take much longer to run

  - **Modern DAST** has evolved from web security scanners and provides good coverage for OWASP type of security and can also perform API security testing. Some of the advantages:
    - Easy to configure and integrate with CI tools
    - Low false-positive rate and easy to use

    Drawbacks:
    - Only covers the endpoints discovered by the web crawler



- **IAST** has been evolving for the last couple years and provides good security coverage, but it heavily relies on QA tests and code coverage. IAST requires integration with web applications and you have to install an agent with your web server.

  Some of the advantages:

  - It can find security vulnerabilities based on code executed during QA tests; coverage is based on QA test code coverage
  - Able to identify security vulnerabilities to the outside of OWASP, such as vulnerabilities with 3rd party libraries, clear text credentials & keys etc.
  - Easy to set up and use and integrate with CI tools

- **Open source analysis,** also known as binary composition analysis, is required to identify open source or third-party library vulnerabilities. There are several challenges in resolving open source vulnerabilities, and in many cases, fixes are not available or there are no guidelines on how to fix the issues.

  o It's possible that your application is not invoking any of the vulnerable methods in runtime
  o The most challenging issues are when you have open source vulnerabilities with transitive dependencies. For example:

  - Project ACME imported a third-party Library A – One possibility that Library A has vulnerabilities and which is straight forward and easy to identify.

  - Another possibility : Library A imports third-party Library B, and Library B has vulnerabilities, it is called transitive dependencies order 1, and difficult to identify exploitability factor

  - Another possibility : Library B imports third-party Library C and Library C has vulnerabilities, it is called transitive dependencies order 2 and so on

  Open source vulnerabilities are difficult and time consuming to triage. Currently, there are no good tools that can help track open source vulnerabilities with source code to identify if any of the known vulnerable libraries is invoked in runtime.

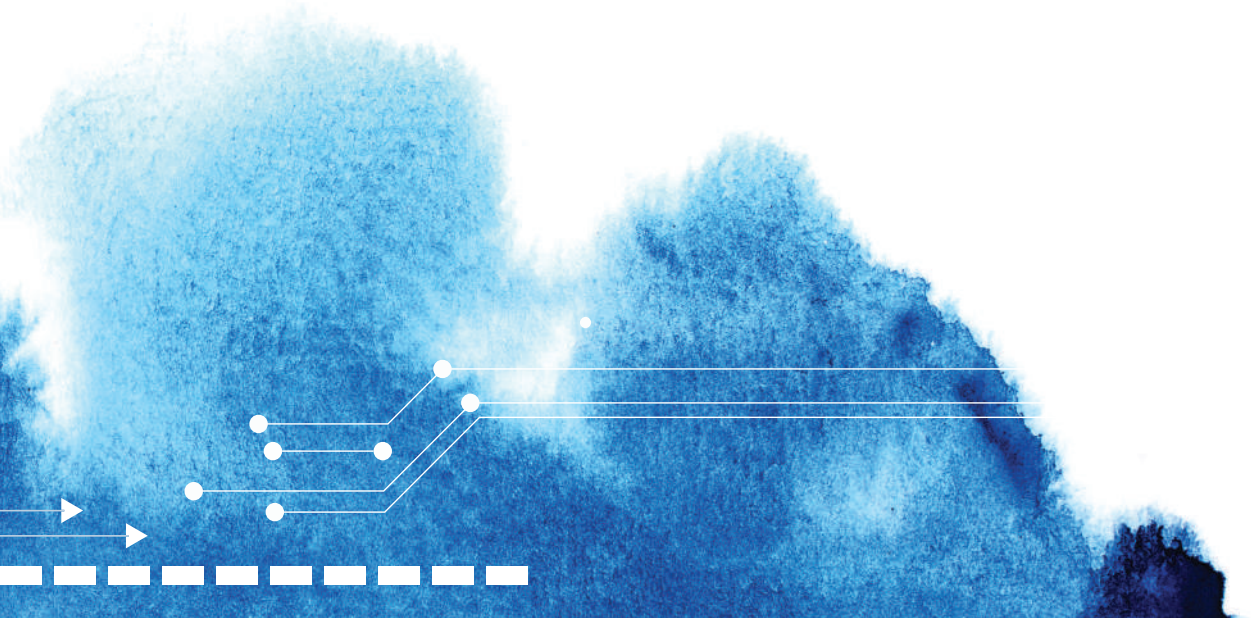
  Practically, it is best to remediate all critical and high severity rated open source vulnerabilities before the release instead of triaging each one of them.

5. Security Operation and Monitoring Workflow

According to Gartner, organizations spend around 95% of their InfoSec budget on building perimeter defenses and security operations, and invest only 5% in application security. DevSecOps is going to have a major impact on the InfoSec budget. With a well-proven "shift left" strategy, a mature DevSecOps implementation can drive down the number of security incidents drastically. DevSecOps helps complete the feedback loop from production to development using automated tools and helps correlate security vulnerabilities and incident-related data from the production environment to release time.

Integrating DevSecOps with security operations and monitoring tools is one of the key benefits and an area of focus for many startups.

- **DevSecOps integration with security operations:** The goal of DevSecOps is to ensure that application production environment security is verified during the deployment time. The most useful security controls include:



	Example of Security Controls
1	All security agents are functional in the production server
2	Web application firewall (WAF) configuration matches the expectation for this application
3	CDN and DDoS protection meet expectations for this application
4	Intrusion detection and prevention technology configuration meets expectations
5	SIEM centralized logging meets the OS and application-level logging requirements
6	User entity behavior and authentication solution is integrated with the application requirements
7	Any incoming attachments or files are scanned for malware before processing and storing
8	Security configurations for web applications, IAM, and access controls are checked
9	Verify firewall configuration and open ports for this application
10	Verify data protection controls for in transit (TLS), in use (HSMs, and key protection security), and at rest
11	Verify all critical and high vulnerabilities detected by network and infrastructure security scanning tools are resolved
12	There are no known security incident issues open

- **DevSecOps integration with security monitoring:** Security monitoring tools play a critical role in discovering security incidents. Many of these tools generate millions of alerts each day, including a lot of false positives. Many of these alerts indicate issues, but in the absence of appropriate security analytics, it is hard to find any specific threat data points from these alerts. The objective behind security alert monitoring is to identify the alerts related to known issues in your application security or network and infrastructure environment.

The data from security alerts is checked against any known issues on an ongoing basis and any security incidents or likely true positive threats should be checked during deployment time.

#	Example of Security Alerts to analyze
1	WAF alerts
2	IDS/IPS alerts
3	SIEM security alerts
4	OS security agent alerts, such as for integrity protection, software signature violations, and privilege access alerts
5	IAM access alerts
6	Firewall alerts
7	DB Access Alerts
8	RASP Alerts
9	Any other security tools alerts that could be correlated to help identify attack path

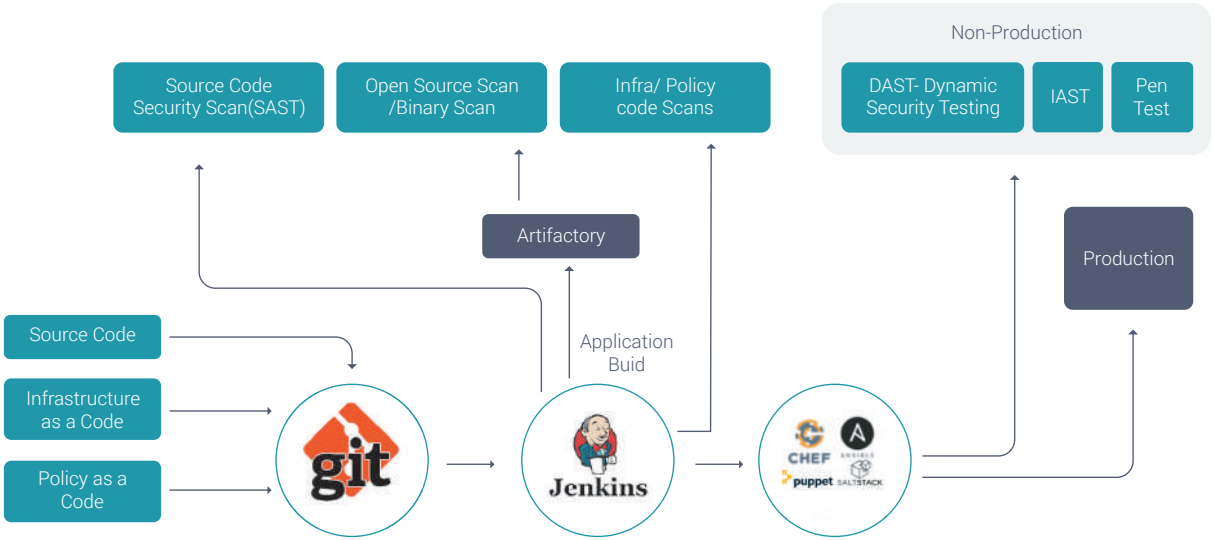
6. CI/CD Pipeline Overview

In this section we share the high-level architecture for Continuous Integration, Continuous Delivery, and Continuous Deployment workflow integration.

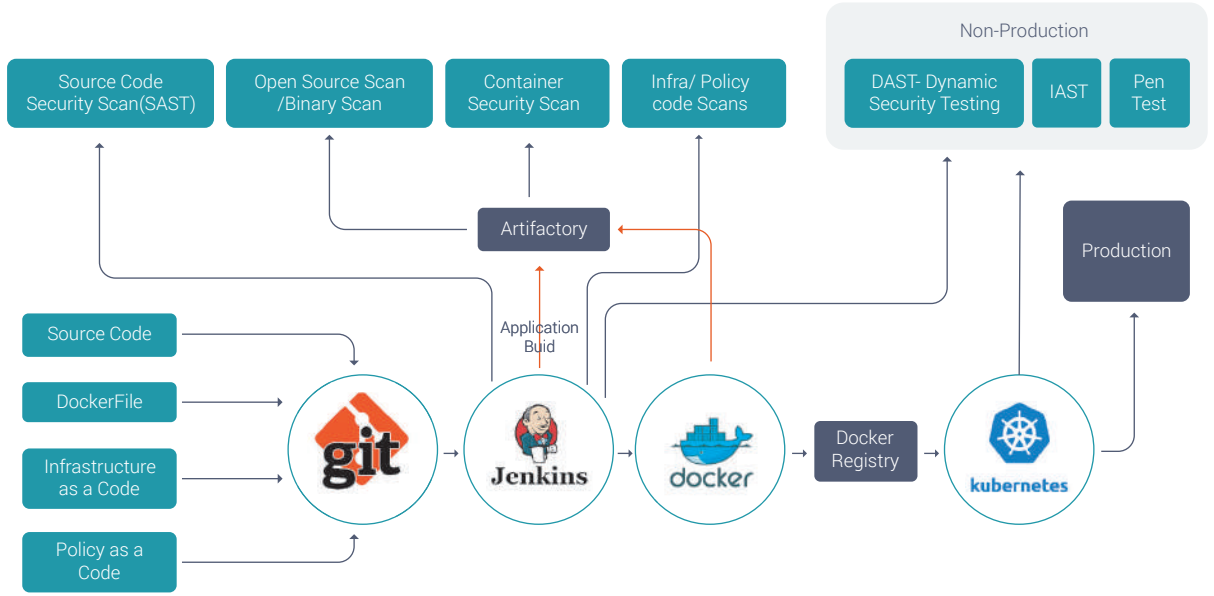
Continuous Integration (CI): The CI kicks in with code check-in, which can trigger an automated build process. The key is a smaller and regular code check-in, which could occur a few times every day, and performing build and test for every incremental code check-in. Though this sounds simple, it requires careful automation from multiple tools and based on different conditions. With every successful build, packaged code is passed to the next stage from binary security scans, automated infrastructure provisioning, code deployment, regression testing, and security testing.

You can set up security testing tools using automated scripts, initiate the test configuration required for a specific build, and continuously monitor the progress of the test. Whether pass or fail, the results of each step must be communicated to the next step and back to the CI/CD dashboard as well.

The following picture covers the CI/CD for applications without containers and applications with containers:



CI/CD Architecture Overview



CI/CD Architecture Overview for Container

7. Security Policies and Security Gates

Enterprise security policies and release security gates must be defined clearly to support enterprise needs for a fast-paced application release process. All code releases are not required to go through all types of application security testing. The DevSecOps automation must be able to pick the required security testing rules automatically based on the release type. You must define the code release security gates and enterprise security policies in collaboration with the development team and operations team, as both play a critical role in DevSecOps success for the enterprise. The code release security gates must also provide the guidelines for the exception or waiver.

- **DevSecOps-related enterprise security policies:** The importance of security policy and governance are well known. They help organizations reduce risk and improve control effectiveness, security, and compliance through an integrated and unified approach across the organization.

Here are examples of enterprise security policies around DevSecOps:

#	Proposed DevSecOps Security Policy	Proposed Policy Options
1	Severity of security bugs not allowed to the in-release branch	Critical, high, medium. It could also be based on vulnerabilities from a specific discovery source.
2	Application Security Testing (AST) requirement criteria for code release	SAST, DAST, open source scan, container scans, pen test
3	Similar vulnerabilities from multiple sources	Similar vulnerabilities from multiple sources have higher chances of exploitability
4	Issues related to past security incidents still present in this code release	How to manage deployment if there are known security incidents related to that application
5	Application security controls – WAF, SIEM, UEBA firewall requirement	Preventive application security control requirements such as UEBA, credential protection, document scanning, etc.
6	Already known network and infrastructure vulnerabilities for this applications asset	How to manage if there are already known security issues in production servers
7	Production security configuration scans for the OS and web components	Security configurations for web servers and OS services to be verified
8	Verify all required OS security agents and check on installed images	Verify all the OS security agent's health checks
9	Number of new issues allowed in production	How many issues would be allowed to go in production for medium/low vulnerabilities

- **Release security gates:** Release security gates are the most important aspect of security policy enforcement in CI pipelines. One of the key requirements for these security gates' effectiveness is to have security tools that meet the low false-positive rate and are easy for development teams to use. You must train the development team to use the tools effectively and understand their roles in DevSecOps. Participation from the development team is required for the success of DevSecOps.

#	Proposed Security Gates	Example Security Testing Rules
1	SAST	For all types of source code changes
2	DAST	For all web applications
3	IAST	IAST is available and only for web applications and when QA coverage is ~80%. Otherwise it's optional.
4	Open Source Analysis (OSA)	During build for all types of source code change
5	Container scan	When container is used
6	Pen test	For all internet-facing new applications, sometimes for major changes, and repeated usually every 12 months.
7	Threat model	This is usually a manual step and only required for new applications and in some cases for major releases.
8	Production security configuration check	Production security configurations to be verified before release; define rules for what needs to be verified.
9	Production network and infrastructure vulnerability check	Use data from the latest production server vulnerabilities scanned to ensure there are no additional risks with the current release.

8. DevSecOps Journey

DevSecOps requires that DevOps is operational and a CI/CD pipeline is actively used for building, testing, and deployment.

Plan: Every release must start with a plan to identify application release scope and code changes planned. This helps create the security assessment checklist that you will follow throughout the release.

- Identify the application release scope and code changes planned
- Identify the release security gates early in the planning process
- Plan for required security training

The following is an example of an application release security assessment checklist:

Release Security Assessment:

#	Example of Release Security Assessment
1	This assessment is for one of the following: 1. New applications 2. Major release: This involves changes to user authentication code flow, role-based access control code, data protection code, privacy-related external file processing flow, API authentication and session management flow, new external APIs, and critical and high severity security bug fixes 3. Minor release: Front-end UI changes, back-end changes, functional bug fixes, no major change in the code flow, and medium- and low-severity security bug fixes
2	Exposure of the application: Internet (consumer facing), limited partner facing, internal employee facing
3	Data classifications Involved: PAI, PHR, PII, confidential data (business sensitive, source code, IP, etc.)
4	Application name and application ID
5	Deployment environment: Public (AWS, Azure, GCP, Oracle, Salesforce), private
6	Type of Application: Web, mobile, data science, SaaS, third-party applications
7	Use of container and sidecar container security
8	Use of infrastructure and policy as a code

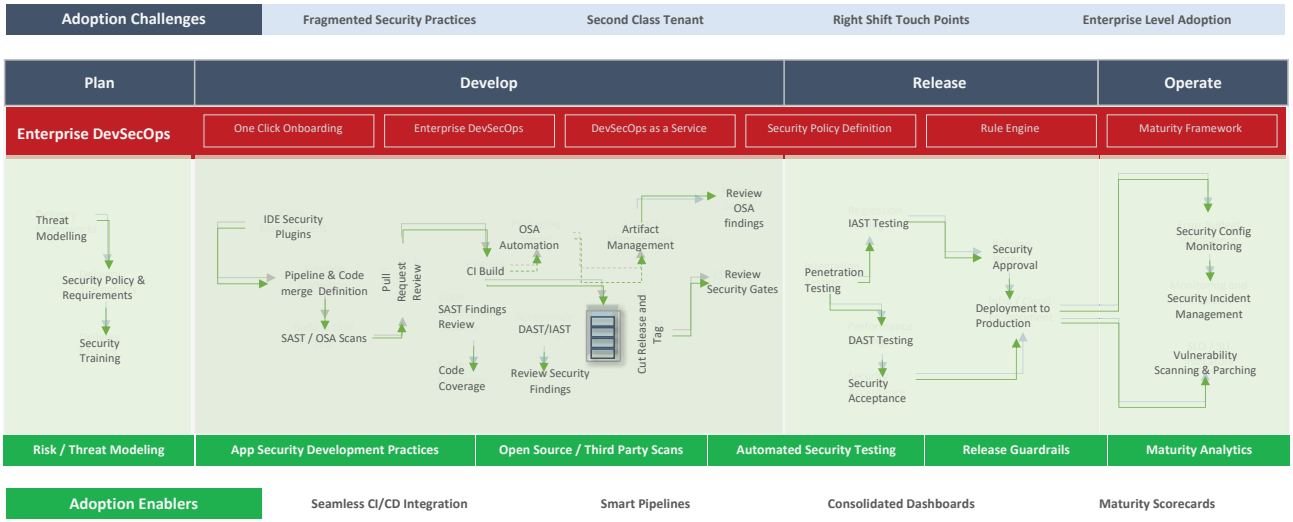
The planning step helps identify appropriate security gates and security training for the development team if this is a new application. It also helps them with the source code baseline review process for SAST.

- **Develop (code, build, and test):** This key point during the development phase is to ensure developers are using SAST as early in the development as possible. This step is part of CI, which involves merging required code branches based on automated rules that drive the rapid integration of iterative software development.
  - **IDE integrated code scans:** Most of the SAST tools offer a plugin to be integrated with IDE such as Visual Studio, Eclipse, and IntelliJ, which will perform continuous code scans or on-demand code scans while the code is being written. This is ideal for resolving any coding issues during development.
  - **Daily build code scan:** Daily build scans are the key CI/CD integration points for SAST, DAST, IAST, OSA, and container scans, based on applicable security gates for this release.

- **Release (pre-release, release, and deploy):** Continuous Delivery (CD) extends CI to incorporate automated software releases within the SDLC pipeline. The builds, with continuously integrated code changes, are automatically released for deployment after completing automated security testing. A release may require further manual approval for business or technical reasons. Based on the release security assessment, this stage can be subdivided into the following steps:
  - **Pre-Release:** DAST/IAST or penetration testing, if applicable, must be performed in the final production build. Based on the pen test findings, there could be additional builds with security fixes.
  - Security code scan for infrastructure as a code and policy as a code

  - **Release:** This is the final build that will be packaged to deploy in the production environment to ship to clients. The following security checks are performed for release:
  - Code signing for the release build
  - Verify security gates and complete security assessment flow
  - Verify production security control checks required for the production deployment

  - **Deploy:** Verify production security configurations and any change in access controls for the services and new infrastructure components.

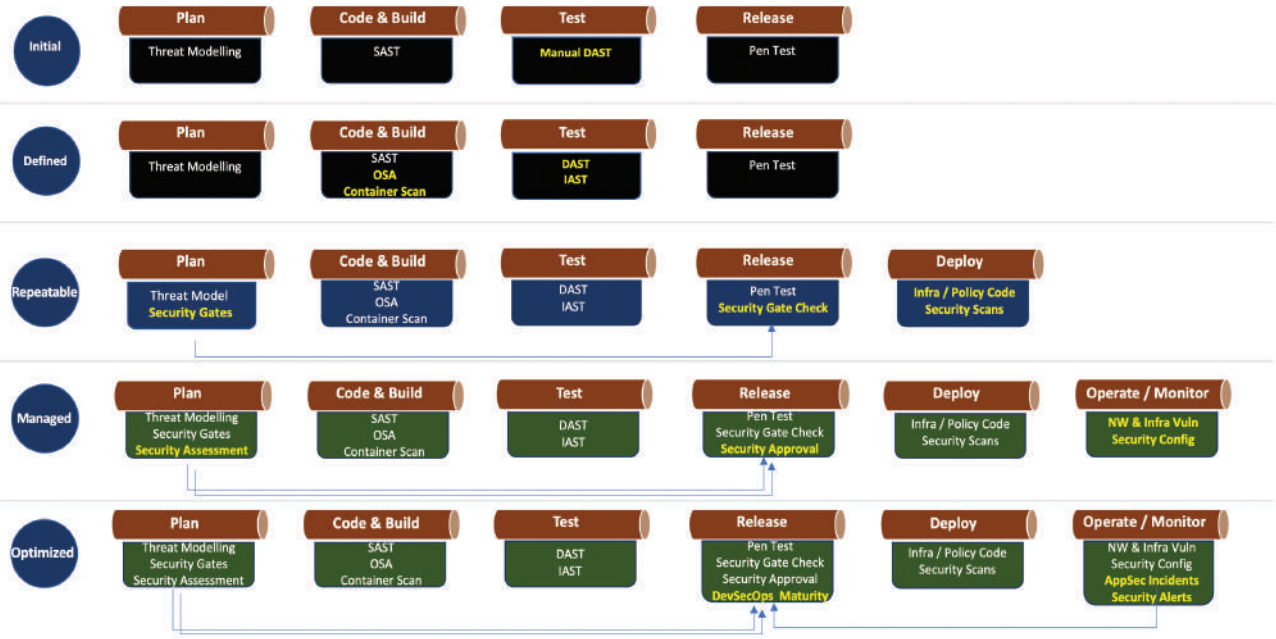


- **Operate (operate and monitor):** Continuous Delivery (CD) extends CI to incorporate automated software releases within the SDLC pipeline. The builds, with continuously integrated code changes, are automatically released for deployment after completing automated security testing. A release may require further manual approval for business or technical reasons. Based on the release security assessment, this

9. DevSecOps Maturity Model

DevSecOps maturity is a key indicator of the consistency of security testing and security policy enforcements, as well as your reduction in production security incidents over time.

The following diagram shows the DevSecOps maturity mapped to NIST cybersecurity assessment model:



Some of the key attributes of a DevSecOps maturity model include:

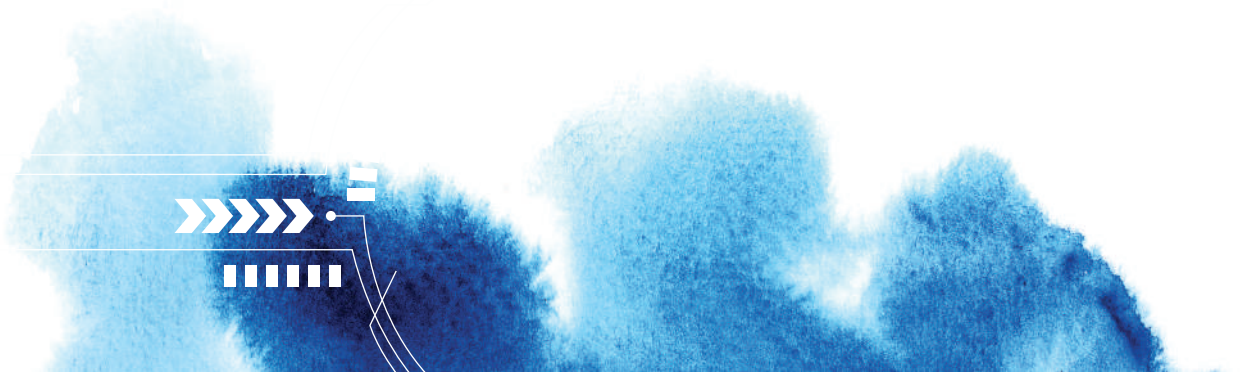
DevSecOps Maturity Model

#	Key Attributes for DevSecOps Maturity
1	Did it meet all security gate/release requirements for the current release?
2	Security exception/waver for the current release
3	Number of security issues identified in the current release (a risk score generated for the application based on application security vulnerabilities detected)
4	Type of release: Major, minor, new application
5	Was it released on time or delayed?
6	Any security incidents for this application detected in production in past six months or one year?
7	Lines of code change introduced – total lines of code changed, to help identify if the change is considered major or minor
8	Past release maturity score
9	Is this application enrolled to production DAST, bug bounty, or RASP?
10	Does this application have any known security issues in production under exceptions for application security (SAST, DAST, IAST, pen test, production DAST, RASP)?
11	Did it verify server security configuration during release/deployment?
12	Did it verify production security control for WAF, SIEM, IDS, IPS, firewall, and any other applicable security controls?
13	Did it use security analytics to help identify any security threats by analyzing alerts and comparing with release security data?
14	Was the infrastructure as a code security review performed?

10. Benefits of DevSecOps

Integrating security with DevOps at a high level of maturity is a long and sometimes challenging process, but one well worth pursuing. Your organization can greatly benefit using DevSecOps, and you will begin to see results with early security testing integrations:

1. Consistent security testing for every release
2. Security is prioritized and is no longer relegated to the last stage of code deployment
3. Developers train on application security, and code security quality improves
4. DevSecOps is the key to shifting left, identifying application security issues as early in the process as possible, and reducing the high cost of security incidents
5. Reduce application security risks
6. Security is in sync with fast-paced development to meet business goals and time-to-market objectives
7. Team silos are removed with security testing automation, close collaboration, and communication
8. Identify any security compliance and audit issues early in the process



Author Bio

Gyan Prakash is a Head of Information Security at Altimetrik. Before joining Altimetrik, Gyan was Global Head of Application Security & Security Engineering at Visa from 2016-2020. He managed application security (SAST, DAST, pen testing, bug bounty, open source vulnerability) and security product engineering. Gyan also led development and deployment of Visa GRC Management Service, Cybersecurity Risk Management Services, and a RASP solution.

Gyan has 20+ years of experience in security technologies. He has implemented mature DevSecOps at Visa and has been consulting with Fortune 500 organizations working to implement DevSecOps at scale. Gyan is a technologist and innovator at heart, with 250 global patents including 152 granted in the areas of system security, mobile security, tokenization, and blockchain.