**DevSecOps Defined**

**Where We Are Now**

We can all be grateful for the recent developments in cybersecurity involving observability and AI, which have increased visibility and granular control, thus greatly improving threat detection and response. The Shift-left paradigm and the programmability and automation introduced by modern DevOps have facilitated increased security throughout the CI/CD pipeline via automated tests for configuration, compliance, and other factors, therefore reducing the opportunity for human error as well.

A picture containing diagram

Description automatically generated

Basic DevOps Lifecycle

APIs, often thought of as the most vulnerable attack vector into disrupting microservices-based architectures, have seen increased hardening thanks to initiatives like the open-source tool, [API Insights](https://developer.cisco.com/site/api-insights/), that Cisco has built to help developers improve API quality and security. This is accomplished through validation and scoring of API specifications as well as trending of API quality with changelog and diff comparisons of AP versions to identify breaking changes.

Cisco’s [Application-First Security](https://developer.cisco.com/application-first-security/) approach leverages [AppDynamics](https://www.appdynamics.com/) with Cisco Secure Application, [Secure Workload](https://www.cisco.com/c/en/us/products/security/tetration/index.html) (Tetration), [Secure Cloud Analytics](https://www.cisco.com/c/en/us/products/security/stealthwatch-cloud/index.html) (Stealthwatch Cloud), and [Cisco Secure Access by Duo](https://www.cisco.com/c/en/us/products/security/duo/index.html) to secure every stage of the application lifecycle. This is implemented across cloud and on-premises deployments, on any workload and integrated into any application.

These DevOps tools and approach go a very long way toward securing our apps and networks, end-to-end. DevOps also takes us further into Shift-left territory by requiring collaboration between security, development, IT, OT, and other stakeholders; this necessary communication gets the conversation started and has dramatically improved readiness, but, as always, even more is needed. Enter DevSecOps.

**What’s Needed: DevSecOps**

DevSecOps is an aspect or offshoot of DevOps which facilitates collaboration between the traditionally siloed IT teams mentioned just above. DevSecOps is the practice of baking security into the DevOps Lifecycle and CI/CD pipeline. We’ve already discussed how DevOps, by necessity, includes security teams in the SDLC process. DevSecOps, by incorporating DevOps values into software security, ensures security becomes an active, integrated part of the development process, at *every*step.

Diagram

Description automatically generated

Basic DevSecOps: Continuous Security in the DevOps Lifecycle

According to [Atlassian](https://www.atlassian.com/devops/devops-tools/devsecops-tools), to implement successfully DevSecOps, teams should:

* Introduce security throughout the software development lifecycle to minimize vulnerabilities in software code.
* Ensure the entire DevOps team, including developers and operations teams, share responsibility for following security best practices.
* Enable automated security checks at each stage of software delivery by integrating security controls, tools, and processes into the DevOps workflow.

Shift-left is a paradigm and DevOps is the strategic avenue to incorporate security in all aspects of the CI/CD pipeline; tools are the tactical arsenal that will get the job done.

**DevSecOps Tools**

Service Mesh, the embodiment of visibility and granular control, is an integral component of DevSecOps, as it helps developers improve security monitoring. [Calisiti](https://calisti.app/" \t "_blank), the Cisco service mesh manager, automates lifecycle management and simplifies connectivity, security & observability for microservice based applications.

New tools have given development teams the ability to chain together interdependent tools and tasks needed to build, test and deliver a working solution.

These new tools enable:

* automation
* container security
* issue tracking
* Software Composition Analysis (SCA)
* cloud testing
* application security
* Static Application Security Testing (SAST)
* Dynamic Application Security Testing (DAST)
* Interactive Application Security Testing (IAST)

All these tools are helpful, if not necessary, but can result in a spaghetti monster of complexity that can end up slowing down a SDLC. One way of optimizing testing tools, including deriving meaningful insight from their data, is with an application security orchestration and correlation (ASOC) solution.

ASOC tools combine the capabilities of application vulnerability correlation (AVC) and application security testing orchestration (ASTO) tools. This provides a management framework for tools, workflows, and prioritization of security activities. An effective ASOC tool is key to DevSecOps because it enables security and development teams to orchestrate testing intelligently, consolidate data from all tools, deduplicate any redundant results, correlate this data based on threat intelligence, and contextualize software risk to prioritize critical findings.

With DevSecOps, security, via these tools, should be applied to each phase of the typical DevOps lifecycle: plan, build, test, deploy, operate, and observe.

Diagram

Description automatically generated

DevSecOps Lifecycle and Tool Implementation

Image via [Atlassian](https://wac-cdn.atlassian.com/dam/jcr:5f26d67b-bed6-4be1-912b-4032de4d06b0/devsecops-diagram.png?cdnVersion=573)

**Software Factory for DevSecOps**

While the tools mentioned in the section above are sure to improve efficiency and productivity, they can come with a large amount of overhead themselves. This ‘tool chain tax’ is best mitigated by putting in place a streamlines software factory. Essentially, this is an end-to-end assembly line of tool consolidation and interaction which makes it easier to manage all the DevSecOps tools for building, testing and delivering our end products.

You can start building your software factory by identifying the scope, high-level architecture and required security practices (and their degree). Tools will need to be standardized and engineering services will largely be implemented based on the available resources as well as speed of adoption.

Building a software factory is an incremental process and thus takes time and effort, but it can be well worth it. As the architecture and tools are chosen and verified, training will be provided to team members on each tool and pipeline.

According to ‘[Building a Modern DevSecOps Software Factory’](https://learn.gitlab.com/devsecops-aware/building-modern-devsecops-software-factory?_pfses=7prCKwunyqUEahBuJa3z2ob1) from GitLab, a software factory *must* address the following:

* Issues & Planning
* Distributed Source Code Management
* Continuous Integration for Every Commit
* Code Reviews & Approvals
* Repository to Manage Binary Assets
* Incremental Deployment
* Continuous Delivery
* Application Monitoring
* Dynamic Test Environments & Infrastructure

**Lay the Groundwork for DevSecOps in Your Organization**

Ready to get started and hit the ground running? Follow these recommendations from [JFrog](https://jfrog.com/devops-tools/what-is-devsecops/" \t "_blank) and start laying your DevSecOps groundwork today!

Maturity Action Plan (MAP)

A DevSecOps Maturity Action Plan (MAP) helps your team develop standardized policies for automated security testing and remediation activities in your DevOps workflows

DevSecOps Culture

A DevSecOps culture is one in which every. single. person. takes responsibility and ownership for security. Taking a cue from DevOps best practices, development teams should assign a volunteer security champion to lead the security and compliance processes in the team in order to elevate the security posture of the processes, products and services therein.

Easily Accessible Data

We’re not having fun unless we’re funneling pertinent security information straight at the developers. Security needs a greater voice because developers vastly outnumber operations and security professionals. According to [Synopsis](https://www.synopsys.com/blogs/software-security/devops-security-training/), that ratio is around 100:10:1.

By using dedicated IDE plugins, leaders can channel security data straight into the tools developers are already using, facilitating adoption of security-driven best practices. A great example of this is API Insights, from Cisco, mentioned earlier, which has an excellent IDE plugin for VS Code.

Automated Governance

Security checkpoints can be distributed at each phase of the DevOps pipeline to determine if your software can continue to the next phase. A governing system can automatically enforce company policies and be able to take action, accordingly, without human intervention. Examples of automate governance include:

* Alerts for security or compliance violations (Webex, Email, Slack, Instant Messages, Jira, etc.)
* Blocking of nefarious downloads
* Failing of builds that depend on vulnerable components or fail to comply with license policies
* Prevention of the deployment of vulnerable release bundles

Practical Best Practices for DevSecOps

* Educate and train developers to develop secure code
* Use automation whenever possible, but do it thoughtfully
* Check the code dependencies
* Track security issues the same as software issues/defects
* Consider a Red Team a Blue Team and Bug Bounties to discover vulnerabilities
* Integrate security controls in the software development pipeline
* Automate security tests in the build process
* Look for known vulnerabilities everywhere in the pipeline
* Inject failure to ensure security is hardened
* Monitor security even after it’s in production

**Learn More**

1. DevSecOps from Cisco Blogs

<https://blogs.cisco.com/tag/devsecops>

*2. DevSecOps: Another Side of Cybersecurity Culture*

<https://www.cisco.com/c/dam/en_us/about/doing_business/trust-center/docs/devsecops-infographic.pdf>

*3. DevSecOps - Addressing Security Challenges in a Fast Evolving Landscape White Paper*

<https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/devsecops-addressing-security-challenges.html>

*4. The Agile Software Factory* – Video from the Defense Acquisition University

<https://media.dau.edu/media/The+Agile+Software+Factory+-+What+the+Acquisition+Team+Needs+to+Know%21/0_lyoz0g52>

*5. Guide to Implementing DevSecOps for a System of Systems in Highly Regulated Environments* – Technical Report from The Carnegie Mellon University Software Engineering Institute

<https://resources.sei.cmu.edu/asset_files/TechnicalReport/2020_005_001_638589.pdf>

6. Enterprise Strategy Group (ESG) Research Report: The Benefits of a Mature DevSecOps Program

<https://www.appdynamics.com/resources/reports/benefits-of-mature-devsecops-program?utm_source=cisco&utm_medium=referral&utm_campaign=cisco-security_spotlight_121820_security-tetration>