**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

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 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.

**What’s Needed: DevSecOps**

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.

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**

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

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

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.

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

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

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.

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.)

**Learn More**

1. DevSecOps from Cisco Blogs

*2. DevSecOps: Another Side of Cybersecurity Culture*