DevSecOps: Injecting Security into CD Pipelines

Learn how DevSecOps impacts on the CD pipeline and the security posture of agile development teams.

BY JUNI MUKHERJEE

What is DevSecOps?

The term DevSecOps is used to describe a security focused, continuous delivery, software development life cycle (SDLC). DevSecOps builds on the learnings and best practices of general DevOps. The application of DevOps values to software security means that security verification becomes an active, integrated part of the development process. Traditionally, and often times unfortunately, security has been treated as a secondary system. InfoSec often engages with development teams towards the end of the SDLC. Noble as their intentions are, it can be frustrating to discover security vulnerabilities at the end of the SDLC.

DevSecOps promotes traditional security engagement to an active process of the SDLC. General DevOps has introduced processes like continuous integration (CI) and continuous delivery (CD). These processes ensure the active testing and verification of code correctness during the agile development process. Similarly, DevSecOps injects active security audits and penetration testing into agile development. DevSecOps advocates that security should be built into the product, rather than applied to a finished product.

Why DevSecOps?

In short: security. The need for safe and secure software is paramount, or else our technology-driven livelihoods will be at risk. Security breaches are one of the largest threats organizations and our governments face today. Several major organizations have been breached in recent times, causing huge fallouts and abrupt C-suite resignations. Failed executives make headlines as consumers continue to lose trust in the compromised service providers.

DevSecOps principles foster collaboration and avoid late handoffs to security professionals. The value is obvious when you review cycle times before and after. Before DevSecOps, your product may be deemed insecure at the last minute, causing multiple costly iterations. After DevSecOps, security gold standards are baked into your product. It is possible that you may find unexpected issues in the last minute, however, the probability is much lower.

So, the point is not as much "Why DevSecOps?" as it is how we can successfully execute in this DevSecOps era. For those entangled in traditional security measures, DevSecOps is a breath of fresh air. Solutions could vary based on your tech stack and your architecture; this is not a "one size fits all" mandate from a centralized organization.

Overall, your security posture enhances your credibility in the market, and builds trust with consumers. With that in mind, this is a good segue way to discuss how DevSecOps ties into the continuous everything paradigm.

DevSecOps and Continuous Everything

Security vulnerabilities can exist in OSS (open source software) libraries that we import just as much as in the code we write. Tons of developers are programming every day and manual code reviews don't scale.This is where the real power of DevSecOps lies.

DevSecOps and continuous everything are like two sides of a coin. DevSecOps works alongside the continuous everything paradigm, and brings continuity to securing our software deliverables.

Continuous delivery pipelines are implementations of the continuous everything paradigm and help validate every commit our teams make. Integrate automated security checks with the pipeline to give you early warnings, and monitor escaped security vulnerabilities relentlessly. Integrated **continuous security** approaches scale as your business expands.

Both unit tests and static code analysis operate closest to source code, and run checks without executing the code. Remember, the cost of a defect is low in test, medium in staging, and high in production. So, invest in security unit tests and static analyzers, since these are inexpensive and fast, and can save trouble further down the pipeline.

Implementing continuous security: unit tests

Your first implementation of continuous security should be into security unit tests.

In continuous delivery pipeline 101, we defined components as the smallest distributable and testable units. They can be validated by unit tests. **Security unit tests** are just as important as the other unit tests we write, but some teams still manage to overlook this category completely.

SAST

Alongside detecting violations in coding best practices, static code analyzers detect security vulnerabilities in code that you own and in (possibly insecure) libraries that you import. This is called **SAST (static analysis security testing)** and modern tools integrate well with the continuous delivery pipeline. Make sure you choose a SAST scanner that's compatible with the programming language of your choice.

A word of caution: SAST can often report false positives and hence plan for a layer of persistence that helps pipelines "remember". False positives can annoy the team to the point where they stop responding to broken pipeline notifications, and that's dangerous. Once teams have identified an error as a false positive with proper justification, don't let the pipeline flag it again and again. This can lead to teams disabling SAST or letting pipelines ignore SAST errors altogether.

DAST

Loosely coupled components make up a subsystem. Subsystems can be deployed and tested for security vulnerabilities using **DAST (dynamic analysis security testing)**. Unlike SAST, DAST examines an application from the outside in its running state, much like what an attacker would do. DAST scanners may not have a dependency on specific languages since they interact with the application from the outside.

The important thing is to include both SAST and DAST in your security strategy since each brings its unique benefits to the table. Integrate both approaches with the CD pipeline so that you get early feedback.

DevSecOps Is the Future of Security

In today's world, just like quality, security is everyone's job. Don't let your vision be limited by a silo of self-proclaimed experts. Reactive corporations and executives that once did so face dire consequences, and are now revitalizing their security strategy with fresh budget.

Traditional security professionals operate in a silo whose

Traditional security professionals operate in a silo whose capacity is limited by the number of security personnel inside that silo. Instead, embrace the agile, decentralized approach of DevSecOps, and retrain your teams to control their own destiny. Additionally, make your product development teams accountable, so that there is no mudslinging between them and the InfoSec department.

With the DevSecOps community thriving, security is not just a business priority, it is also the latest and greatest thing to integrate with the continuous delivery pipeline! Continuity, armed with security, ensures that the best days of software delivery are ahead of us.

SHARE THIS ARTICLE

JUNI MUKHERJEE

Juni is a thought citizen in the DevSecOps space and has made deep investments in the field of Continuous Delivery. She has helped organizations build Continuous Delivery Pipelines, and would love to solve the problems that plague our industry today. She has authored a couple of books.

TUTORIAL

Continuous Delivery Tutorial

In this guide, we'll see how you can use Bitbucket Pipelines to adopt a continuous delivery workflow. Read on!

Try this tutorial →

ARTICLE

Feature Branching Workflows for Continuous Delivery

Using feature branching workflows in your continuous delivery pipeline keeps your most important branches in a clean and releasable state and allows develo

Read this article →

Up Next

Feature Branching Workflows for Continuous Delivery →