Devolutions

Building Your Way to DevSecOps

WHAT STEPS CAN YOU TAKE TO BEST INTEGRATE COMPLIANCE AND SECURITY INTO YOUR EXISTING DEVOPS PROCESS?

Once in a cafe, there was a dialogue between a waiter and a visitor:

- *I asked not for chicken, but beef, and not half-cooked, but fried.*
- *Yes, that is a mistake, sorry, but everything is fresh here, so Bon Appétit!*
- *I am a surgeon. Did I understand correctly that when you get to me and instead of appendicitis, I remove something else, I simply need to apologize?*

Is only medicine or the food industry supposed to be safe and give the desired result? What is the developer's responsibility for their product? Does budget-saving provide a good reason to release vulnerable software?

Since any organization should hypothetically want to achieve the best results, to be in demand by society and applying a reasonable effort to achieve its aims, let us address the problem of moving from DevOps to comprehensive DevSecOps.

DevOps implies the development and the operations teams working together to accelerate release. In the case of DevSecOps, software development also prioritizes the goal of creating a secure product. So what steps can you take to best integrate compliance and security into your existing DevOps process?

Four Important Ingredients

Software development is like baking bread, where the four main ingredients (flour, yeast, water, and salt) are put together to form a strong yet pliable structure. Software development requires a good basis that will provide the desired taste and desired output when the product leaves the oven. A high-quality end product is not only about the neatness of the release; it is also about security.

Trade, finance, healthcare, the public sector — the requirements for software security in these industries are very high. However, the number of coders in these industries is usually hundreds of times greater than the number of information security specialists and auditors. And so, security and compliance become the bottleneck in the development process. Mitigating this issue requires automating security processes and embedding them into the software development and delivery pipeline, which involves four ingredients:

1. Shifting compliance and security assignments to the earliest stages in the development pipeline.
2. Integrating security testing into the continuous delivery process.
3. Tracking the status of each step in product creation.
4. Visibility and assessment of security vulnerabilities.

These four ingredients will help any company in any industry transition from DevOps to DevSecOps in a harmonious way.

Shifting Compliance and Security Assignments to the Earliest Stages in the Development Pipeline

Typically, developers create code and submit it for testing. The vulnerability testing produces a multi-page report, and the implementation of countless recommendations often leads to a delay in release.
DevSecOps is about deploying the identification, remediation, and prevention of security and compliance issues in the development lifecycle at its earliest stages. While still writing the code, the developer can receive recommendations on how to ensure its security. In this case, the team can correct the shortcomings in the normal mode.

Ideally, these assignments should be automated and integrated into the development pipeline. Before introducing this approach, revising the security testing tools is recommended. It is quite possible that some solutions introduced 5 or 10 years ago can be replaced with other tools for monitoring application security and static code analysis. This will make the security testing process simpler and more accurate.

By integrating automated security checks that start during development and continue through production, you can ensure that your code is secure from the first commit to the end of release support.

Shift Left aimed at ensuring safety and compliance does not mean that these issues are left entirely to the developers. It is most effective when developers, operations, and information security work together to achieve it.

Integration of Security Testing into the Continuous Delivery Process

Continuous testing is the best practice for teams adhering to continuous delivery. Automating compliance and security testing by static analysis and code composition tools such as SonarQube or Fortify is a very common choice for developers and information security services. Testing tools allow you to check if a particular piece of code meets security and regulatory requirements. The overwhelming majority of projects actively use open-source code, so tools for automating the verification of open-source components and their dependencies are becoming indispensable. In particular, WhiteSource helps to pick up high-quality components at the stage of their selection and inclusion in the project.

It is best to combine the results of different tests with other information related to the software release so that you can see the whole picture.

In most large companies, security and compliance tests may not always be presented as pass/fail. Often, the product owner, release managers, security, and compliance specialists make decisions to release a product

based on business goals, despite identified issues. An important nuance here is that most of the above specialists are not as close to the creation process as the developers. And developers may never see the final test results. They need access to these results in order to understand what they mean in relation to both individual product features and the release as a whole.

Tracking the Status of Each Step of Product Creation

Shift Left that implies integrating security and compliance audits into the software development pipeline generates a lot of data. One of the options for making use of the data is to establish a chain of events for each release. It is a set of detailed results of all actions in the development process, containing information about what happened, when it happened, and who did it. Automatic data capture and logging enable businesses to track the development process and record the results at each stage, providing testers with the necessary information to confirm that all checks have been completed. Such an approach makes it possible to analyze errors in the case of a failure, accumulate the general knowledge of the company, and facilitate the investigation of incidents. In addition, teams receive a tool for continuous product quality management, detecting all shortcomings at the earliest stages: unfulfilled tasks, tasks that are not regularly performed (or performed with errors), and manual procedures that can be automated.

Visibility and Assessment of Security Vulnerabilities

Collecting security and compliance data and recording actions during the release process is not enough. You need to ensure that everyone involved in the software delivery process can get visual information and compliance assessment based on the requirements for their own part of the process.

An enterprise chain of software delivery tools typically consists of multiple specialized tools, each providing its data and reporting. Individual reports do not allow you to see the overall picture of the development process. And without the big picture, actions required to ensure security and compliance remain hard to envisage.

It is important to automatically extract the data you need from DevOps tools, generate it on demand, and add it to the context through which the data can be understood.

For example, the results of security testing of a specific feature will not help a compliance specialist identify a violation. However, once the professional sees how this functionality operates, interacts with other functions, and is deployed, the violation may become obvious to spot.

DevSecOps' Fundamental Principles

DevSecOps builds upon the following core principles:

1. Worst-case scenarios must be considered when designing. Users do not always follow the optimal or expected scenario. You need to consider and validate malicious user stories, simulate the threat landscape, and simulate different levels of failure.
2. Security testing at every stage of the pipeline. You can use real attack tools like Metasploit. Security should not be in the form of a paper document but in the form of a code stored in a repository, where any developer can retrieve it. DAST, SAST, IAST, and open-source component validation tools should be built into the pipeline.
3. AppSec training is not enough. There is also no need to train all developers to write safe code. People write the code, and for 10 thousand lines of code, there will be a certain number of errors, some of which will lead to vulnerabilities. Training is good but you also need the right tools and automation to solve this problem.