VERACODE

Practical DevSecOps :

Reduce Risk and Go to Market Faster

Table of Contents

03 Introduction

04 DevSecOops - A Cautionary Tale

06 What is DevSecOps?

07 Practical Steps for Adopting DevSecOps

- 07 Establish a Common Vernacular
- 07 Integrate Security Early and Often
- 11 Automate Security with a Developer-Centric Approach

12 Conclusion

Introduction

Implementation of a DevSecOps approach is the most impactful key factor in the total cost of a data breach, **according to IBM**. This eBook will explore what successful DevSecOps looks like and how it can be implemented in practice. By the end, you'll know how and why to build a modern software development workflow around security from the get-go.

DevSecOops – A Cautionary Tale

Before we dig into the elements of a successful DevSecOps implementation, let's cover an example of what happens when the "Sec" is missing from DevOps – or is siloed and left to the end of the software development lifecycle (SDLC).

The following is a fictional story, though it relays the all-too-real story of many companies you've heard of, buy from, or maybe even show up to work for every day.

Fail Fast Technologies was a company like any other attempting to develop modern software at a rapidly increasing rate using agile processes and trying to move to DevOps.
However, they had a problem with security flaws in their applications. The development team was focused on meeting deadlines and didn't prioritize secure coding or thorough code testing. They only performed basic scans on critical applications before adding them to the main repository. As a result, their technical debt kept growing, and by the five-year mark, 70% of their applications had security flaws, because like most companies, **their average application footprint grew by about 40% per year.**

Due to limited resources, there was only one person responsible for security coordination between the development and security teams. This person would manually create JIRA tickets for each developer after routine security tests. However, this process took a lot of time, causing delays in fixing vulnerabilities. The security team struggled to keep up with the pace of development, and the liaison often wondered why these issues weren't caught earlier. Although there were suggestions to organize workshops for developers to code more securely, there was never enough time to make it happen.
The security team was overwhelmed with testing new applications and patching urgent vulnerabilities that made the news before Fail Fast even knew about them. The liaison knew they needed to take a comprehensive look at the entire application ecosystem, but there was just never enough time.
Then, one day, the liaison had to deliver devastating news to the CISO. A malicious actor had exploited a SQL injection vulnerability, resulting in a significant breach. Millions of customer records containing highly sensitive personal information had been stolen.

It was an ordinary day until the CISO received that call**...**

What Went Wrong?

As the Fail Fast teams investigated the breach and tried to find the main cause, they discovered that the vulnerability could have been identified during the application's development through a **Static Analysis** scan. However, the scan was skipped because the development team was under pressure to release the app quickly before an analyst review deadline. The vulnerability could have also been found through a **Dynamic Analysis** scan after the app was launched, but since the app was older and not as important to the company's operations, the security team didn't prioritize it for scanning.

They realized that they didn't have a consistent way of examining and prioritizing the results of scans. Sometimes, developers would ignore issues if they seemed unimportant or if fixing them would delay the release of new features. To compound the issue of ignoring flaws, there was the development team's use of AI to assist with code creation. While it helped developers release code faster, it also introduced more flaws more quickly.

The developers needed to be able to perform critical scans efficiently and handle security issues on their own, instead of relying heavily on the security team. This required integrating security processes, tools, and methods into their current workflow. The security team and operations team needed to trust that the developers were taking all necessary precautions to prevent introducing new flaws into the code. They also needed a way to assess the risk to their systems in an ongoing manner.

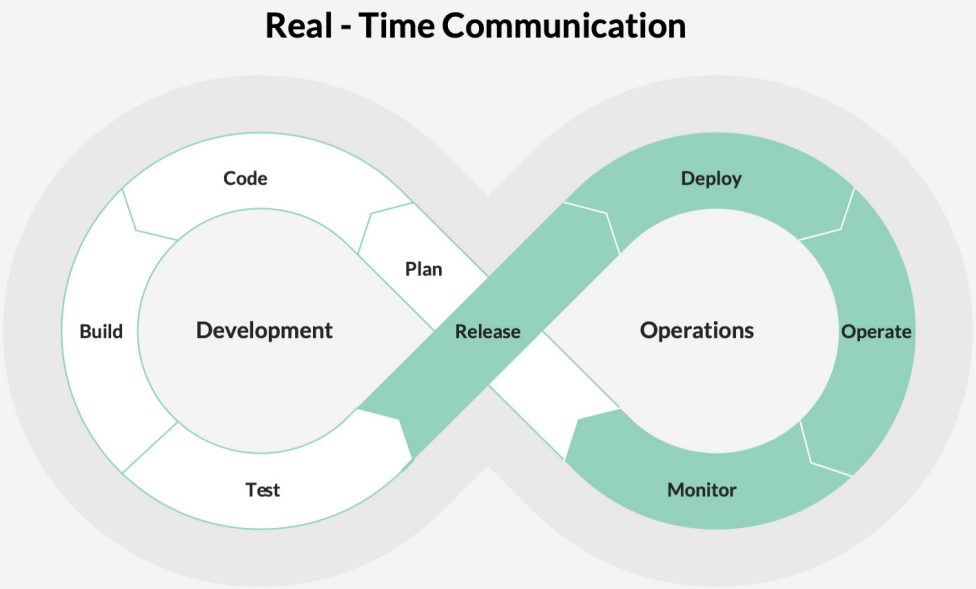
Upon closer examination, they realized that some manual processes, like assigning tickets for fixing vulnerabilities, were slowing down the remediation process. However, they understood that these were just symptoms of a larger problem. While adding some automation could help, the root cause was the inconsistent collaboration between security, development, and operations. They considered using AI-assisted remediation tools, but they needed to make sure they could trust the fixes and know it was not just brushing things under the rug to check a box.

Security needed to be integrated into every stage of Fail Fast's software development process. The teams understood that simply using various security tools wouldn't solve the problem or change the company's culture. It was challenging to onboard and integrate multiple tools, and they needed a better way to implement security policies, track progress in reducing technical debt, and address new flaws in the code. They needed a common language and a unified approach to viewing their security status as a team. The Fail Fast team needed a DevSecOps makeover.

What is DevSecOps?

The cornerstone of a successful DevOps practice is automation; this is why adopting DevSecOps (security automated within workflows) makes so much sense. DevSecOps is surrounding each step of the DevOps process and practice with security. By adding security into each step of the SDLC from coding and building to operating and monitoring compliance to policy – code bases, applications, and APIs are designed, built, and deployed with security in mind. Integrating security into each DevOps function effectively creates DevSecOps – an overarching layer covering all activity along the SDLC.

The thing is, successfully implementing DevSecOps is one of the most difficult problems to solve due to a variety of factors. Cultural adoption is one of the biggest blockers to shifting left and optimizing the "right" shifted evaluation and testing of the attack surface. Now we will return to our story of Fail Fast Technologies and see what steps they took to successfully incorporate security into DevOps to create DevSecOps.



Practical Steps for Adopting DevSecOps

While Fail Fast was already implementing DevOps, they needed to holistically automate security in their process to successfully adopt DevSecOps and avoid running into another crisis like the SQL injection.

Establish a Common Vernacular

The Fail Fast team started by making sure everyone understood the same definitions of terms. This helped prevent confusion and made it easier for team members to know what they needed to do before moving their code to the next stage of development.

CWE vs CVE

CWE stands for **Common Weakness Enumeration**. It's a list of common weaknesses in software and hardware that have security problems. There are thousands of weaknesses, but the most well-known ones are the **SANS Top 25** and the **OWASP Top 10**. These weaknesses give teams a common language to talk about security problems and help them know what to prioritize and fix. CVE stands for **Common Vulnerabilities and Exposure**. It's a way to talk about specific security issues. Each CVE has a unique number and is scored for severity. CVEs can only be issued by a certified authority and are real vulnerabilities.

Security Flaw vs Vulnerability

A security flaw is an issue in how something is implemented that can lead to a vulnerability. A vulnerability is a weakness in the code that can be exploited by a bad actor. Not all flaws are vulnerabilities, but all vulnerabilities are flaws. Any changes to an application can expose it to attacks.

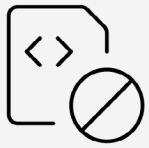
Integrate Security Testing Early and Often

By integrating security testing early and often, the Fail Fast team was able to identify and address potential vulnerabilities and flaws in a timely manner.

Based on data from the **State of Software Security 2024**, roughly 63% of applications have flaws in first-party code and 70% contain flaws in third-party code. That's why testing both throughout the SDLC is so critical.

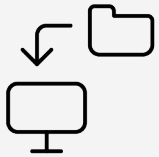
In addition to scanning early and often, the team began with an iterative approach to **threat modeling** before any code was even written. This creates guidelines and stops problems before they ever start.

Types of Testing



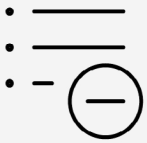
Static Application Security Testing (SAST):

Scanning source or binary code in a-pre-production or “static” state to find security flaws or CWEs. Also called “white box” testing.



Software Composition Analysis (SCA):

Analyzing the open-source libraries and third-party components in an application for CVEs. Used in the creation of Software Bill of Materials (SBOM) that supports one or both standard formats: CycloneDX and SPDX.



Dynamic Application Security Testing (DAST):

Simulating attacks in a production, runtime, or “dynamic” state to expose configuration and end-point weaknesses also known as “black box” testing. Note: SAST finds security flaws while DAST finds vulnerabilities.

Benefits of Scanning Early:

Early Identification of Vulnerabilities:

By scanning for security vulnerabilities early in the SDLC with SAST and SCA, you can identify and address potential security flaws at an early stage. This allows for prompt remediation, reducing the risk of these vulnerabilities being exploited later in production. Using a trusted AI tool to aid the remediation cuts down significantly on time spent remediating.

Cost and Time Savings:

Addressing security vulnerabilities early in the development process is more cost-effective and time-efficient compared to fixing them later. Early identification and remediation of vulnerabilities help prevent the accumulation of technical debt and reduce the need for extensive rework or redesign.

Improved Code Quality:

Scanning early enables developers to receive fast feedback on potential security issues in their code. This feedback loop promotes better coding practices and encourages developers to write more secure code from the start. As a result, the overall code quality improves, leading to more robust and secure applications. Promoting the use of **hands-on, interactive training tools** that use real code to understand the potential risk being introduced helps developers eliminate a flaw before the first line of code is even committed.

Enhanced Security Awareness:

Early scanning raises security awareness among developers and the development team as a whole. It helps them understand the importance of security and encourages them to prioritize security considerations throughout the development process. This increased awareness fosters a security-conscious culture within the organization.

Benefits of Scanning Often:

Continuous Risk Assessment:

Regular security scanning allows organizations to continuously assess the risk posture of their applications. By scanning often, they can identify new vulnerabilities that may arise due to changes in the codebase, third-party libraries, or emerging threats. This proactive approach helps organizations stay ahead of potential security risks.

Rapid Vulnerability Remediation:

Regular scanning enables organizations to quickly identify and remediate newly discovered vulnerabilities. By addressing vulnerabilities promptly, organizations can minimize the window of opportunity for attackers and reduce the potential impact of security breaches.

Compliance and Regulatory Requirements:

Many industries have specific compliance and regulatory requirements related to security. Regular scanning helps organizations meet these requirements by ensuring that their applications adhere to the necessary security standards. It provides evidence of ongoing security efforts and helps organizations demonstrate compliance during audits.

Continuous Improvement:

Scanning often allows organizations to gather valuable data and insights about their application's security posture over time. By analyzing trends and patterns in the scan results, organizations can identify areas for improvement, track progress, and refine their security practices. This continuous improvement cycle helps organizations strengthen their overall security posture.

> **Veracode was really easy to use, and developers were able to go in and scan early and often. In the first eight months, we had 18,000 flaws fixed. It was just phenomenal.**
>
> *- Nikki Veit, Director of Application Development for the State of Missouri*

Automate Security with a Developer-Centric Approach

When Fail Fast automated the steps above, they knew they had the recipe for DevSecOps nirvana. Automation plays a vital role in DevSecOps, and it is important to adopt a developer-centric approach to security automation. This means providing developers with the necessary tools, resources, and training to automate security processes seamlessly within their existing workflows.

Fail Fast asked themselves, "What will make security easier for developers? What makes it easy for them to have a high-quality, secure application?" The answer was clear: the integration and automation of security should grow out of development processes – out of the developer tool chain and workflow.

By integrating security tools and practices into the **development environment, IDE, and source code managers (SCMs)**, like **GitHub**, developers can easily perform security scans, receive real-time feedback, and remediate vulnerabilities efficiently. This developer-centric approach empowers developers to take ownership of security and enables them to proactively address security concerns throughout the development process.

Continuous scanning must be accompanied by continuous remediation to be effective.

The automation of remediation is a critical piece. Development teams that fix flaws the fastest are **4x less likely to let critical security debt** materialize in their applications.

Veracode's AI-driven remediation tool, **Veracode Fix**, can address many CWE categories with severity ratings ranging from medium to very high. This innovative approach, leveraging a highly curated set of reference patches from our security research team, enables organizations to proactively reduce security debt and strengthen their software security posture.

Conclusion

DevSecOps is not a destination; it is a journey. It is not a state to achieve, but a continuous process to implement and uphold. The key is to integrate and re-integrate security seamlessly, adapting to evolving development needs and changing tooling. By staying pervasive without being invasive, organizations can ensure a secure and efficient development environment. Remember, DevSecOps is an ongoing commitment to maintaining a strong security posture while enabling agile and innovative software development practices.

Veracode's world-class platform enables you to establish continuous security around your legacy and cloud-native applications. We support you in seamlessly integrating and automating security into your entire SDLC, bringing security and development together, and providing a vehicle to define and implement a set of security policies that align to the business criticality and operating environment of software in production. With Veracode solutions, support, and services, you can avoid and overcome the challenges of securing your software from start to finish.

Click here to schedule a demo of Veracode today and let us show you how easy we can make DevSecOps for your organization.