# DevSecOps: How Proactive Security Integration Reduces Your Agency's Risks & Vulnerability

The software development lifecycle (SDLC) hasn't changed all that much since Agile was introduced in the early 2000s, but the need for speed in the development process certainly has. Applications that once took weeks or months to build can and are turned around in days or even hours to support a mission. However, very often this speed comes with a caveat, according to Sandy Carielli, a principal analyst for Forrester Research. "Applications remain the most popular attack vector, open source continues to infect everything, and too many industries are not investing in the application security controls they need," she writes.

The reason is simple: Security, for most organizations and agencies, is reactive. It happens at the end of the SDLC. It's something that occurs when the application is complete, and can be lengthy and inexact—and often does not test the code in its entirety, frequently skipping testing code that is not externally facing. Some organizations focus on only one testing method while others may rely on infrequent external penetration testing that meets minimum compliance requirements. Security professionals come in and, during a process that's completely disparate from the development process, run tests to find vulnerabilities, reporting back to the development team when they are finished.

Once the security testing process is complete, developers must go back and make changes and find fixes after the fact. The development process is then lengthened, and labor costs may go up as developers are stuck fixing problems when they should be on to their next projects. In addition, invariably some security holes aren't found at all – or not until the application is deployed, which creates a huge security risk for the mission and the agency.

"You don't know what you don't know until you get to the end of the process when you're running development the traditional way. That creates substantial volatility in the delivery schedule and in budgets when effective predictability is not there," explains Traci Robinson-Williams, technology business strategist for Regulated Industries at GitLab. "You can end up with more risks and vulnerabilities as well as lower-quality code." GitLab is an open core software company that promotes stewardship of its open source community contributions, while carefully balancing the additional security requirements of its subscription-based customers.

This traditional SDLC paradigm is what most developers are facing today, according to the 2019 GitLab Global Developer Report, which found that 50 percent of those surveyed say that security vulnerabilities are discovered by the security team after code is merged and in a test environment. In addition, today's security practices get dismal ratings by developers. Only a quarter say their security practices at their organizations are good, while 30 percent say they are just fair. Nearly a quarter (23%) say their security practices are poor.

## Improving Results by Reducing Risk

There are other issues with the traditional SDLC as well. For instance, while the Agile development process allows agencies to speed development along, it's also brought with it an ever-burgeoning set of tools. Today, the average development team may have up to a dozen or more tools to help them with the process—including, version control and source code management, project management and collaboration, continuous integration, continuous delivery, code review, security testing, and continuous monitoring, among others.

The plethora of applications adds cost to the mix in the form of license fees, integration efforts, and maintenance expenses—what can be deemed a 'toolchain tax'—but it also creates silos of data and introduces reporting challenges and compliance issues. Complexity is further compounded since all of the tools must be integrated, and different members of the DevOps team may

use disparate tools. "Every time you bring on a new developer from outside of the agency, there's a huge learning curve to get that person up to speed on the toolchain you're using internally," says Robinson-Williams.

There is a solution for these issues, though. Moving from a reactive to a proactive security strategy that features continuous security testing in a single, integrated development and testing platform, will improve overall security. Using a single platform, code changes are automatically tested with every code commit and results are presented to the developer for instant remediation. This significantly improves application security, and helps developers achieve authority to operate (ATO) more quickly. Developer job satisfaction—and retention—shoots up, too since they aren't stuck wasting time doing labor-intensive mitigation.

The most significant change with a single integrated development and testing platform is the ability to make security an end-to-end part of the entire development process. Vulnerabilities are found earlier in the cycle and can be mitigated as they are uncovered by the developer, who doesn't need to wait for a security professional to find them in the finished product. This is extremely beneficial because when developers can take an active role in identifying and remediating vulnerabilities as they are working on code, they can speed the development process along, says Robinson-Williams.

"Every time the developer is committing code back to the repository, there are checks that happen. In real-time, they get feedback that there's a problem; there's something going on with it and they need to fix it, instead of building on top of problems and having more vulnerabilities to fix at the end," she says. "That's going to enable the DevSecOps team as a whole to deliver applications faster with fewer vulnerabilities."

The GSA agrees. "Further, by incorporating security into the coding process (i.e. DevSecOps), loopholes and weaknesses are exposed early on so that remediation actions can be implemented."

## Taking Development into the Future

Having automation in a single platform speeds ATO as well. By creating what is called a 'software factory', agencies can automate governance policies and testing, bringing them into the continuous delivery process. Every merge request is automatically tested using static application security testing (SAST), dynamic application security testing (DAST), dependency scanning, container scanning, and license management. There's more collaboration, too, since everyone in the SDLC has access to an integrated security dashboard that displays a list of unresolved vulnerabilities and metrics around development. Anyone on the team can see the code itself and details about its problems in real-time.

One agency's experience is a good example of how automation not only reduces security risks but improves ATO. A security agency was looking to deploy mission-critical code updates to the field faster than the two weeks it was routinely taking to get ATO approval. Using GitLab's dependency scanner along with some of its own security framework requirements, the agency transformed its entire development process.

"The agency was able to reduce time to ATO—in addition to their number of vulnerabilities and level of risk—from two weeks to two hours. Using GitLab Ultimate with built-in security scanning, they were able to meet their mission requirement for continuous monitoring of code being pushed to production," explains Robinson-Williams.

Today, the agency can test every piece of code without adding cost or time into the equation. It's something that every agency and organization will need to do in the future in order to meet mission goals in a timely fashion, while improving security and compliance as well as end-user satisfaction, says Robinson-Williams.

"Because the application security tool and the source control tool are the same, there's no need to look at and identify incremental new code. The code repository is showing you precisely what's changed. The tracking flows all the way through the process. You know exactly what's changed, who has changed it, and when it was changed. You know definitively whether there are any new vulnerabilities that have been introduced into your code—and that lowers the agency's risk."