DevOps tech: Shifting left on security

Note: *Shifting left on security* is one of a set of capabilities that drive higher software delivery and organizational performance. These capabilities were discovered by the DORA State of DevOps research program (https://www.devops-research.com/research.html), an independent, academically rigorous investigation into the practices and capabilities that drive high performance. To learn more, read our DevOps resources (/devops).

Security is everyone's responsibility. The 2016 State of DevOps Report (https://services.google.com/fh/files/misc/state-of-devops-2016.pdf) (PDF) research shows that high-performing teams spend 50 percent less time remediating security issues than low-performing teams. By better integrating information security (InfoSec) objectives into daily work, teams can achieve higher levels of software delivery performance and build more secure systems. This idea is also known as *shifting left*, because concerns, including security concerns, are addressed earlier in the software development lifecycle (that is, left in a left-to-right schedule diagram).

In software development, there are at least these four activities: design, develop, test, and release. In a traditional software development cycle, testing (including security testing), happens after development is complete. This typically means that a team discovers significant problems, including architectural flaws, that are expensive to fix.

After defects are discovered, developers must then find the contributing factors and how to fix them. In complex production systems, it's not usually a single cause; instead, it's often a series of factors that interact to cause a defect. Defects involving security, performance, and availability are expensive and time-consuming to remedy; they often require architectural changes. The time required to find the defect, develop a solution, and fully test the fix are unpredictable. This can further push out delivery dates.

Continuous delivery borrows from lean thinking the concept of building quality into the product throughout the process. As W. Edwards Deming says in his Fourteen Points for the Transformation of Management (https://deming.org/explore/fourteen-points?apartner=aarp), "Cease dependence on inspection to achieve quality. Eliminate the need for inspection on a mass basis by building quality into the product in the first place." In this model, rather than taking a purely phased approach, developers work with security and testing experts to design and deliver work in small batches (/architecture/devops/devops-process-working-in-small-batches) throughout the product lifecycle.

Research from DevOps Research and Assessment (DORA) (https://services.google.com/fh/files/misc/state-of-devops-2016.pdf) (PDF) shows that teams can achieve better outcomes by making security a part of everyone's daily work instead of testing for security concerns at the end of the process. This means integrating security testing and controls into the daily work of development, QA, and operations. Ideally, much of this work can be automated and put into your deployment pipeline. By automating these activities, you can generate evidence on demand to demonstrate that your controls are operating effectively; this information is useful to auditors, assessors, and anyone else working in the value stream.

How to implement improved security quality

Shifting the security review process "left" or earlier in the software development lifecycle requires several changes from traditional information security methods, but is not a significant deviation from traditional software development methods on closer inspection.

Get InfoSec involved in software design

The InfoSec team should get involved in the design phase for all projects. When a project design begins, a security review can be added as a gating factor for releasing the design to the development stage. This review process might represent a fundamental change in the development process. This change might require developer training. It might also require you to increase the staff of the InfoSec team, and provide organizational support for the change. While including InfoSec might represent a change in your organization, including new stakeholders in design is not a new concept and should be embraced when considering the benefits.

Develop security-approved tools

Providing developers with preapproved libraries and tools that include input from the InfoSec team can help standardize developer code. Using standard code makes it easier for the InfoSec team to review the code. Standard code allows automated testing to check that developer are using preapproved libraries. This can also help scale the input and influence from InfoSec, because that team is typically understaffed compared to developers and testers.

Develop automated testing

Building security tests into the automated testing (/architecture/devops/devops-tech-test-automation) process means that code can be continuously tested (/architecture/devops/devops-tech-continuous-integration) at scale without requiring a manual review. Automated testing can identify common security vulnerabilities, and it can be applied uniformly as a part of a continuous integration pipeline or build process. Automated testing does require you to design and develop automated security tests, both initially and as an on-going effort as new security tests are identified. This is another opportunity to scale the input from the InfoSec team.

Common pitfalls

Some common pitfalls that prevent teams from shifting security left include the following:

- **Failing to collaborate with the InfoSec team**. The biggest mistake is when teams fail to collaborate with their InfoSec teams. InfoSec is a vitally important function in an era where threats are ubiquitous and ongoing.
- **Understaffing InfoSec teams**. InfoSec teams are often poorly staffed. James Wickett, Senior Security Engineer at Verica, cites (https://blog.xebialabs.com/2017/04/20/10-tips-integrating-security-devops/) a ratio of 1 InfoSec person per 10 infrastructure people per 100 developers in large companies.
- **Engaging too late with the InfoSec team**. In many cases, the InfoSec gets involved only at the end of the software delivery lifecycle, when it is usually painful and expensive to make changes that are necessary to improve security.
- **Being unfamiliar with common security risks**. Many developers are unaware of common security risks such as the OWASP Top 10 (https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project) and how to prevent them.

Ways to improve security quality

You can improve software delivery performance and security quality by doing the following:

- **Conduct security reviews**. Conduct a security review for all major features while ensuring that the security review process doesn't slow down development.
- **Build preapproved code**. Have the InfoSec team build preapproved, easy-to-consume libraries, packages, toolchains, and processes for developers and IT operations to use

in their work.

- **Integrate security review into every phase**. Integrate InfoSec into the daily work of the entire software delivery lifecycle. This includes having the InfoSec team provide input during the design of the application, attending software demos, and providing feedback during demos.
- **Test for security**. Test security requirements as a part of the automated testing process including areas where preapproved code should be used.
- **Invite InfoSec to demos**. If you include the InfoSec team in your application demos, they can spot security-related weaknesses early, which gives the team ample time to fix.

Ways to measure security quality

Based on the stated ways to improve outlined above, you can measure security in the following ways.

Factor to test	What to measure	Goal
Whether features undergo a security review	The percentage of features that undergo security review early in the design process.	This percentage should go up over time.
Whether security review slows down the development cycle	How much time the review add to the development process.	The time that security reviews take should go down until it reaches an agreed-to minimum.
How well security is integrated into the delivery lifecycle	The degree of InfoSec involvement in each step of the software delivery lifecycle. For example, you can measure the number of security reviews captured at each of the stages of the software development lifecycle (design, develop, test, and release).	This value should go up until it reaches a value that suggests that InfoSec is fully integrated into the lifecycle.
Whether automated testing covers security requirements	The involvement of the InfoSec team in writing automated tests. As InfoSec gains greater input into the testing process, the number or percentage of security requirements that are included in the automated testing process.	This percentage should go up over time.

Factor to test	What to measure	Goal
The use of preapproved libraries, packages, toolchains, and processes	Initially, whether InfoSec is engaged in tools development. As work progresses, the number of InfoSec-approved libraries, packages, and toolchains that are available, or the number of these resources that are used by the development and operations teams.	Engagement should increase over time until the organization agrees that InfoSec oversight of tools is at the correct level. Similarly, the percentage or number of preapproved tools in use should increase until the team uses all the tools that InfoSec has created or approved.

What's next

- For links to other articles and resources, see the DevOps page (/devops).
- See the OWASP Top 10 (https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project) for a list of the 10 most critical web application security risks.
- Read the Site Reliability Engineering (SRE) book (https://landing.google.com/sre/books/).
- Explore our DevOps research program (https://www.devops-research.com/research.html).
- Take the DevOps quick check (https://www.devops-research.com/quickcheck.html) to understand where you stand in comparison with the rest of the industry.

Last updated 2022-01-13 UTC.