INTEGRATING INFOSEC INTO THE DELIVERY LIFECYCLE

Arguably the DevOps movement is poorly named—ignoring functions such as testing, product management, and information security. The original intent of the DevOps movement was—in part—to bring together developers and operations teams to create win-win solutions in the pursuit of system-level goals, rather than throwing work over the wall and pointing fingers when things went wrong. However, this kind of behavior is not limited to just development and operations, it occurs wherever different functions within the software delivery value stream do not work effectively together.

This is particularly true when discussing the role of information security teams. Infosec is a vitally important function in an era where threats are ubiquitous and ongoing. However, infosec teams are often poorly staffed—James Wickett, Head of Research at Signal Sciences, cites a ratio of 1 infosec person per 10 infrastructure people per 100 developers in large companies (Wickett 2014)—and they are usually only involved at the end of the software delivery lifecycle when it is often painful and

expensive to make changes necessary to improve security. Furthermore, many developers are ignorant of common security risks, such as the OWASP Top 10,[1] and how to prevent them.

Our research shows that building security into software development not only improves delivery performance but also improves security quality. Organizations with high delivery performance spend significantly less time remediating security issues.

SHIFTING LEFT ON SECURITY

We found that when teams "shift left" on information security—that is, when they build it into the software delivery process instead of making it a separate phase that happens downstream of the development process—this positively impacts their ability to practice continuous delivery. This, in turn, positively impacts delivery performance.

What does "shifting left" entail? First, security reviews are conducted for all major features, and this review process is performed in such a way that it doesn't slow down the development process. How can we ensure that paying attention to security doesn't reduce development throughput? This is the focus of the second aspect of this capability: information security should be integrated into the entire software delivery lifecycle from development through operations. This means infosec experts should contribute to the process of designing applications, attend and provide feedback on demonstrations of the software, and

ensure that security features are tested as part of the automated test suite. Finally, we want to make it easy for developers to do the right thing when it comes to infosec. This can be achieved by ensuring that there are easy-to-consume, preapproved libraries, packages, toolchains, and processes available for developers and IT operations.

What we see here is a shift from information security teams doing the security reviews themselves to giving the developers the means to build security in. This reflects two realities: First, it's much easier to make sure that the people building the software are doing the right thing than inspect nearly completed systems and features to find significant architectural problems and defects that involve a substantial rework. Second, information security teams simply don't have the capacity to be doing security reviews when deployments are frequent. In many organizations, security and compliance is a significant bottleneck for taking systems from "dev complete" to live. Involving infosec professionals throughout the development process also has the effect of improving communication and information flow—a win-win and a core goal of DevOps.

Compliance in the Federal Government

Federal information systems are subject to the Federal Information Security Management Act of 2002 (FISMA). FISMA requires that federal agencies follow NIST's Risk Management Framework (RMF). The RMF includes multiple steps, such as the preparation of a System Security Plan which documents how

the relevant information security controls (325 for a moderate-impact system) have been implemented, and then an assessment resulting in a report (the security assessment report or SAR) which documents the validation of the implementation. This process can take from several months to over a year, and is often only *begun* once the system is "dev complete."

In order to reduce the time and cost taken to deliver federal information systems, a small team of civil servants at 18F created a platform as a service called cloud.gov based on an open-source version of Pivotal's Cloud Foundry, hosted on Amazon Web Services. Most of the controls in systems hosted on cloud.gov—269 of the 325 required for a moderate-impact information system—are taken care of at the platform level. Systems hosted on cloud.gov can go from dev complete to live in weeks, not months. This significantly reduces the amount of work—and thus cost—needed to implement the requirements of the Risk Management Framework.

Read more at https://18f.gsa.gov/2017/02/02/cloud-gov-is-now-fedramp-authorized/.

When building security into software is part of the daily work of developers, and when infosec teams provide tools, training, and support to make it easy for developers to do the right thing, delivery performance gets better. Furthermore, this has a positive impact on security. We found that high performers were spending 50% less time remediating security issues than low performers. In other words, by building security into their daily work, as opposed

to retrofitting security concerns at the end, they spent significantly less time addressing security issues.

THE RUGGED MOVEMENT

Other names have been proposed to extend DevOps to cover infosec concerns. One is DevSecOps (coined by a few in the industry, including Topo Pal of Capital One and Shannon Lietz of Intuit). Another is Rugged DevOps, coined by Josh Corman and James Wickett. Rugged DevOps is the combination of DevOps with the *Rugged Manifesto*.

- I am rugged and, more importantly, my code is rugged.
- I recognize that software has become a foundation of our modern world.
- I recognize the awesome responsibility that comes with this foundational role.
- I recognize that my code will be used in ways I cannot anticipate, in ways it was not designed, and for longer than it was ever intended.
- I recognize that my code will be attacked by talented and persistent adversaries who threaten our physical, economic, and national security.
- I recognize these things—and I choose to be rugged.
- I am rugged because I refuse to be a source of vulnerability or weakness.
- I am rugged because I assure my code will support its mission.

- I am rugged because my code can face these challenges and persist in spite of them.
- I am rugged, not because it is easy, but because it is necessary and I am up for the challenge (Corman et al. 2012).

For the Rugged movement to succeed—and in line with DevOps principles—being rugged is everybody's responsibility.

---

[1] For more information, see https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.