SUBJECT7

QUALITY IS A CULTURE

DevOps and SecOps finally intersecting, what this means for your process

By Matt Heusser for Subject7

DevOps started with a few revolutionary observations in their day yet seem obvious today. For example, before DevOps, you had the programmers, whose job was to create change, and operations, whose job was to ensure reliability or uptime. The easiest way to ensure uptime was to prevent change. Framed that way, operations and programming are enemies; their goals are *oppositional*. DevOps brought operations into the fold, giving them all the tools to manage production deployments, making them responsible for production uptime and system reliability. The timing was perfect – the adoption of the cloud, infrastructure as code, feature flags, and the standard use of Continuous Integration (CI) tools all happened about that time. These changes made on-demand environments possible. In some cases, automated checks increased reliability further. If DevOps had arrived five years earlier, it would have sputtered out.

Now, people have started talking about SecOps in the same way.

What does it even mean to be "doing" SecOps, anyway? And the new mega buzzword, DevSecOps, what exactly does that mean? Could the ideas ever work, how is it beneficial, and is the timing right to get started? Today we'll aim to answer these questions, plus give you some ideas on how to assess your team, how to get ready – and when you may consider getting started. To begin: What do those terms even mean?

PUTTING SECUIRITY INTO OPS

As it is classically presented, Security is sort of the enemy… of everyone. The Security "team" locks down the computers so installing applications is hard. They make installing anything hard, insisting on internal controls for changes, locking technical staff out of production data, forcing any "live" data be anonymized, and enforcing audit trails for artifacts. Like testing, this work is a bit like an insurance policy – it costs time and money, and you hope it isn't needed.

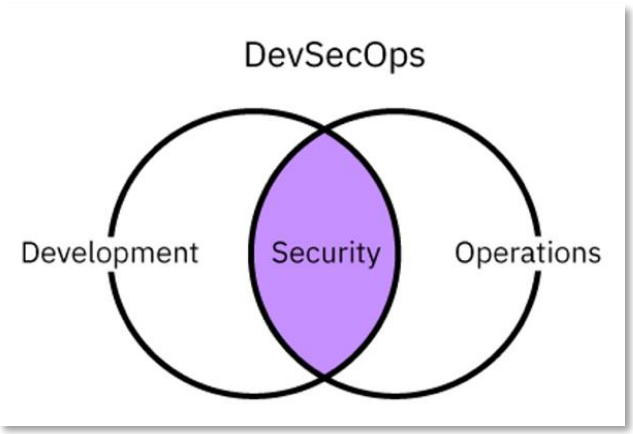


Image Credit: IBM

In testing, a job well done usually means bug reports and changes that make the software better or provide some level of confidence in reliability. In Security, a job well done means a lack of incident reports.

In other words: Security costs money and slows the team down, while the value it adds is invisible. This is a problem.

SecOps is an attempt to resolve that problem by creating a functionally integrated Security+Operations team. That team would be different from the "DevOps people" but would have an adjoined mission. Before the "security team" would create "security requirements" that involved Operations creating a printout of some random tickets every quarter, the SecOps team would have both disciplines and work together to streamline operations while ensuring that security controls are being met.

Enter Shift Security Left, with a goal of designing software with security in mind. Rather than giving "Security Requirements" to a product owner, shifting security left involves providing the development team with the tools, skills, and processes to have security-checked software as an outcome of the development cycle. That means when Continuous Integration finishes, the code is security-checked, ready to go to production, with a complete audit trail. Another term for this kind of team is DevSecOps.

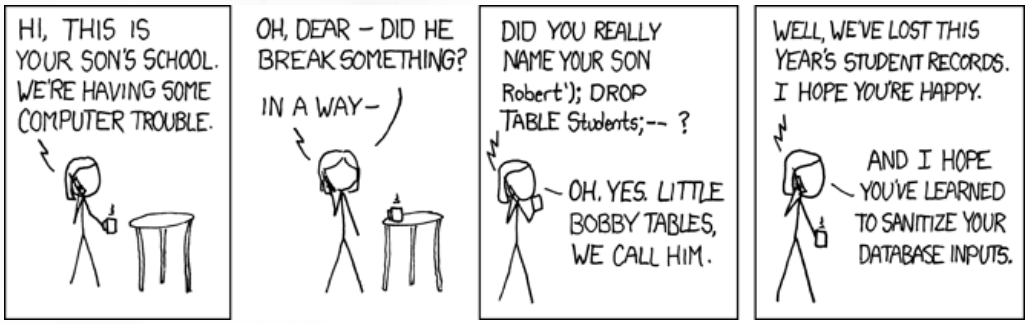
DevSecOps has great promise, but in most cases, it's been watered down to a sound byte that doesn't convey all the different components of security and how those would work within a development team.

Even the overlapping circles diagram above is a little misleading. It is not about what these things have in common, as much as how they can be used to leverage each other. If the goal is streamlined DevOps starting to intersect with security, we need to ask how security can be automated and connected to Continuous Integration.

Let's dig into that next…

COMPONENTS OF SECURITY

To call software "secure," we would want a set of checks for certain types of vulnerabilities. Programmers can make code errors that lead to vulnerabilities, such as SQL injection or buffer overflow attacks. Static code scanning tools can do this when the custom code is available. That software is likely to run on operating systems and interact with databases where the code is not available. Still, some versions of the software (or some configurations) have known security holes. Software Composition Analysis, or SCA, can run on the components of the software, particularly open-source components that have no formal vendor or best practices. Dynamic Security Testing, also called "Penetration Testing," involves exercising the software while looking at every exposed element in the user interface, trying to expose and leverage common security vulnerabilities and attack vectors. It's easy enough to point a tool at a website and ask it to look for vulnerabilities – but it is harder to have it successfully navigate through the workflow and test every point.



There is a lot more to security than the items noted above. Access and role permissions, virus scanning, security audits of users and superuser activity, physical security of devices and equipment, threat and risk analysis … and the list goes on. Still, producing running and tested software that has already incorporated those checks will save a lot of effort, time, and money. Arming the team with integrated tools and making security a part of the dev process helps to make those other security "problems" a whole lot easier.

Some of these challenges have already been simplified. SecOps has been running vulnerability scanning for some time. Static code scanning is almost plug and play. The cloud, particularly when leveraging container technology, makes all the components a digital download packet that can be inspected. Every container has a manifest of the components, where they come from, and the version, making Software Composition analysis a snap.

The one challenge that still stands is Dynamic Security Testing. Most companies continue to hire experts to do a "pen test" between major releases. That means either pen testing holds the release hostage, or is sacrificed for a deadline – if it happens at all. Like DevOps, the timing of a major advancement has to be perfect. With Dynamic Application Security Testing products becoming smart enough to navigate the User Interface, that time may be now.

Surprisingly, the final piece of the puzzle might just be the humble, functional automation test tool. Imagine test automation tooling that lets you functionally test the application while simultaneously enabling you to run security tests during every stage of the workflow. What if prioritized vulnerability reports were produced for each run, enabling your team to identify functional defects as well the interwoven security issues? Those integrated and unified tools exist and are helping to shift security left in more meaningful and effective ways as well as fulfilling the DevSecOps vision.

THE BUSINESS CASE

There are several benefits to integrating DevOps and SecOps into a cohesive team. As alluded to in the previous section, the premise is to leverage integrated automation tooling that fulfills a number of testing disciplines in a unified solution. With security expertise embedded in the tooling, traditional functional testers are empowered and upskilled to contribute to the security testing mission.

First, the group can eliminate delays from security checking (and that's not just the overhead of the semi-annual audit) but all of the delays that happen when security comes into the picture just before go-live and stops the release due to issues. By incorporating security testing into the dev process, those unexpected delays will become a legacy of an inefficient past.

Second, this process tends to "cut the fat" from the security organization, allowing it to focus on value-added work, not just repetitive risk reduction. This value is similar to the value that automated testing delivered to functional testing. The integrated security work becomes woven into the automated testing process.

Third, and most importantly, this work detects and prevents security defects from escaping to production. That eliminates fire drills, theft, fraud, misuse and abuse of your software by bad actors. And it's all possible with existing resources, while reducing the cost and reliance on outside pen-testing firms to validate the security of the release.

WHERE TO START

Given that functional test automation tools walk the core workflows of the application, with expected results along the way, there is a natural synergy with Dynamic Application Security Testing. With the right tool in place, the security scans will "just run" automatically along the defined functional workflow. That doesn't have to stop with security testing; similar principles can apply to accessibility testing. That means a programmer that makes a UI change that introduces an accessibility or a security issue can be notified before the software is merged into the master branch. There is no check-later, no fire drill when an external vendor finds four or five security risks the week before the big go-live. This also changes the economics of security testing, as it is suddenly cost-efficient on every build, allowing companies to release much more frequently with less risk.

Where to start depends on the size and scope of the firm and the test automation tooling already in place. Cutting-edge vendors are making integrated functional, security, and accessibility testing plug and play. Since the solutions embed the know-how to identify security and accessibility concerns, nearly any tester can leverage the capabilities to create exception and risk reports. It's even better with the latest wave of codeless test automation tooling that enables non-technical testers and business users to create and run sophisticated automated tests. You'll still want to consult with your security experts to help mitigate any identified threats, but the job of detecting them becomes a lot easier.

About twenty years ago, the Integrated Development Environment started to provide programmers with all they needed to see about programming, including version control, the build, and even the ability to run unit tests. Today the same kind of thing might be happening for risk, with functional test automation tools offering run results that include penetration and accessibility, and Continuous Integration tools providing the rest. As a testing specialist in Michigan, and a bit of a mentor to me, often says "security today is asking for a seat at the design table instead of being a heavy hand that comes later, slowing down the process."

Yesterday, DevSecOps was a buzzword, a slogan, a goal. Today it is possible to run fully automated checks of security, accessibility, and features on every change. The generated "build" package will include an audit trail strong enough for Payment Card Industry (PCI) security standards, Financial Technology, and even HIPPA. These technologies exist and are being used by companies right now, today.

Learn more about how Subject7 is Unifying Test Automation with Codeless Automation that integrates Security and Accessibility testing in a single cloud-based solution….

SUBJECT7
QUALITY IS A CULTURE