Take a risk-based approach to DevSecOps
Companies embracing digital transformation are looking to DevOps and agile development methods to accelerate the release of new applications. Unfortunately, in this need for speed, security is often left behind.

Just as rapid, secure and modular code development can help drive digital transformation, insecure code can add risk and ultimately undermine digital transformation efforts. The release of “buggy” software — that requires time-consuming revisions or introduces unknown vulnerabilities that result in costly breaches — can lead to a loss of trust among developers, security teams, customers and business leaders. There is also the hidden cost of unmanaged patch processes.

Per numerous estimates, 50 percent to 90 percent of all successful exploits against enterprises and systems can be traced back to vulnerabilities in software as the root cause. So companies must take proactive steps to protect against software vulnerabilities by baking security into the DevOps process, rather than bolting it on at the end. Firms should also be better prepared for their reactive processes, such as vulnerability disclosure programs, penetration testing and vulnerability bounties.

If we consider DevOps to be an environment in which developers who think and act like operators work collaboratively with operators who think and act like developers, then DevSecOps is the extension of that orientation. In such an environment, security concerns that affect activities from development through release and run would be addressed up front, ensuring confidentiality, integrity and availability. The entire process requires the CIO’s commitment to introducing new policies and processes to bring developers and security teams together.

Organizations today should consider operating in a comprehensive DevSecOps model to enable secure digital transformation. DevSecOps supports faster release schedules and innovation, but for this approach to succeed, organizations must make fundamental commitments, including changing the culture of collaboration, building security throughout the development life cycle, and evaluating technical and business risks.

**Identifying the problem**

The issue of security is so broad and diverse that embedding it in a continuous integration and delivery pipeline can be a daunting task. For example, in the development phase your developers will need access to build and test environments. How will these be managed to ensure that appropriate controls are in place? Will you enforce controls on local versions of applications services, such as messaging and database, so they align with access controls that will be applied to the application in production? How will you manage credentials needed by applications to authenticate application services?

**Align development, operations and security**

Critical to the facilitation of DevSecOps is the separation of concerns, e.g., separating configuration from application, which is a facet of cloud-native application development. Many organizations are finding it complex to merely align development and operations; certainly, this third dimension will have its challenges as well. Automation, frameworks, and strong policy and governance will, ultimately, simplify the cooperation required for this type of environment. However, you should expect that at the beginning, it will be seen as disruptive and burdensome.
A few companies have long understood that security should be a part of the software development process, and the industry has come up with a variety of frameworks over the years, such as the Open Software Assurance Maturity Model (OpenSAMM) and the Building Security In Maturity Model (BSIMM). A general term for security in the software development life cycle, which we’ll use in this paper, is the secure software development life cycle (Secure SDLC).

**Implement a Secure SDLC**

The idea is to spot early in the process any potential problems, such as missing or incomplete security requirements and vulnerable architecture, design, libraries, platforms, function calls or code. In practice, however, few companies have successfully implemented a Secure SDLC. Or if they have, it has been applied ad hoc and with limited scope.

A key inhibitor is the fundamental tension between the developer’s desire to move quickly and the security practitioner’s need to minimize vulnerabilities, even at the risk of slowing things down. That tension was already present in the older world of waterfall-style application development and has only been exacerbated by the move to agile development, where release cycles are quicker, the lines between the various steps in the process have become blurred and processes can overlap. In the agile development world, the increase in velocity also increases the risk that errors will be introduced into the process.

**Understand the impact of “technical debt”**

The authors would like to introduce here the concept of technical security debt. In software development, technical debt refers to suboptimal decision making that sacrifices long-term benefits for short-term gains; for example, quick-to-implement hacks, poor requirements gathering or just lack of knowledge. However, over time and with increasing changes, this debt accumulates. The higher debt then needs to be paid off by implementing increasingly difficult-to-make changes. Let’s take this concept of technical debt and apply it to DevSecOps, where debt can be in the form of known or unknown vulnerabilities (security defects) that arise through poor security governance of the software development and release processes.

When developers race through the software development life cycle without appropriate support and governance, hidden technical debt (e.g., unknown vulnerabilities) builds up.

**CATA: Foundation for future development**

The good news is that there are ways to design security into the DevOps process, such as using DXC Technology’s Comprehensive Applications Threat Analysis (CAT-A) service. The long-term benefit of CAT-A is ensuring that the architectural and security recommendations become the foundation for future development projects. DevSecOps enables teams to create infrastructure and security as code, going forward. The long-term results are improvements in consistency, quality and reliability. The challenge is that full-blown threat analyses such as a CAT-A review could take 3 months or more, so it’s critical for organizations to perform routine scanning more frequently to meet the demands of a weekly or biweekly code release schedule.
Let’s consider: If the technical debt is not addressed, applications might need minor revisions or complete rewrites, which effectively cancels out the benefits of agile development. Even more critical, vulnerabilities remain undetected and uncorrected, which could ultimately cause major security breaches or exploits. Real examples from the news include hospitals being held hostage to ransomware; or organizations having many people’s credit card and Social Security numbers stolen, leading to the risk of mass identity theft or disclosure of Protected Health Information (PHI). In the case of software vendors, this feeds the patching cycle of their customers, which assumes a vulnerability disclosure process with which to notify the market of such issues.

It’s the old “you can pay me now or you can pay me later” scenario, and similarly, the economics reflect that of personal finance. Paying later comes with interest, except that in the case of defects and vulnerabilities, the interest rates are exorbitant (fixing issues post-release can cost orders of magnitude more than during the early life cycle). The preferred approach is to reduce/repay technical debt as early in the pipeline as practical, before it accumulates. Periodic security reviews are able to turn hidden technical debt into visible technical debt, which is then drained down far sooner and at lower cost than without such security reviews.

**Short- and long-term assessments**

This is a continuous process that needs to be prioritized, especially for specific applications, based on their risk profile. Such a process most benefits those organizations that know the value of their information assets and have a policy and process[es] in place to classify them.

Gearing to technical debt refers to a way of calibrating the weekly or biweekly pace of agile development sprints against the much slower cycle of security reviews. Applying the principles of gearing to technical debt is critical to the successful implementation of DevSecOps.

Here’s how it might work. A security scan might be applied every five iterations or sprints for a high-risk application and applied every 20 sprints for a low-risk application. And a full CATA review might be conducted once a year for a low-risk application. For a high-risk application, it might be conducted twice a year or at the time of a major release, whichever comes first. This CATA assessment would cover all changes made to software applications over the period in question.

**Understanding the roles of CIO and CISO**

Ideally, a move to DevSecOps should start at the top, although it doesn’t always have to. But a chief information officer (CIO) who is driving digital transformation should create policies designed to bridge the gap between developers and security teams. There must also be an effort to change the culture and to bring developers, operations teams, security groups and business units together so that they all understand how important DevSecOps is to the company’s overall success (Figure 1).
The risk assessment piece should be owned by the chief information security officer (CISO), who defines policies that can identify the risk profile of each new application. Meanwhile, risk ownership should remain with the line of business that is supported through the process. One useful resource might be DXC’s Cyber Reference Architecture, a framework of security capabilities that can be used to create a blueprint for complex security transformation efforts.

The risk assessment should include not only the technical risk, but also factor in the business risk. For example, is this a mobile application going out to millions of customers, or is it an internal application designed to streamline a business process?

In addition, the CISO should be making sure that key security reviewers and software developers are trained in the use of security assessment tools. Also, the CISO needs to identify which steps in the process require a hands-on security practitioner.

When it comes to training, developers need to understand that DevSecOps is more than a tool or a process. It’s a way of thinking about application development, in which security needs to be considered at each step of the way. Over time, this cultural change can have a powerful effect on how developers go about their work, as they embrace security in the development life cycle.

Where to begin

The best approach is to start small with one high-risk project and build out from there. If we envision the DevOps pipeline beginning with such areas as architecture, design and platforms, then moving into the actual writing of code, then to prerelease testing, postrelease testing and finally patching, we realize that few companies have applied security to those earlier stages. However, most companies are well versed in patching.

Tools and assessment strategies

Here are some of the specific tools and techniques that can be applied at each stage of the development life cycle (Figure 2). Note that while the terms sound like waterfall terms, they represent layers of abstraction in any software or system, regardless of whether agile or DevOps life cycles are used. It is important to note that some of these automated tools, rather than slowing things down, can actually help developers work faster and more effectively.
• **Requirements, architecture and design.** The CATA service has two components: Security Requirements Gap Analysis (SRGA), which addresses security at the requirements layer, and Architectural Threat Analytics (ATA), which looks at the robustness and resilience of the architecture from a security perspective. SRGA determines what security controls are missing and closes those gaps. The result is a more secure application architecture, going forward.

• **Code.** As long as humans (or even code generators) are writing code, errors will inevitably be introduced into the process — and increasing the speed of software releases magnifies the likelihood of errors. Static application security testing (SAST) tools scan for those errors. In fact, the newer versions, designed specifically with agile development in mind, can pinpoint mistakes even as developers are typing, much the way the spelling checker works in Microsoft Word. However, a human security expert review of source code, a second set of eyes, is also required as part of the DevSecOps process, as electronic scanners can find only some classes of vulnerabilities.

• **Runtime.** For checking applications at runtime, there is dynamic application security testing (DAST), which includes runtime scanners, application vulnerability assessment and penetration testing. As with all stages in this process, an assessment by a security professional is recommended.

• **Patching.** This is the one piece that most enterprises have adopted. From a DevSecOps perspective, however, patching occurs far too late in the process and has its own pitfalls and gotchas, such as change windows and legacy systems. Patching is reactive, while proactive avoidance of releasing buggy code in the first place is always preferable.
Next steps

If done right, DevOps can help drive digital transformation. But if security is left out of the equation, companies risk security breaches — devastating, harder to detect and caused at the application level — and the release of buggy applications. This can undermine digital transformation efforts and erode trust between teams and clients.

The answer is to embed security into the application development life cycle through DevSecOps, which requires a culture change, an organizational change, and the application of specific security tools and techniques.

Here are four tips for getting started with DevSecOps:

1. **Change the culture.** CIOs need to set the tone and bring security teams, developers, operations groups and business unit leaders together in pursuit of a unified approach to secure application development and deployment.

2. **Build in security throughout the development life cycle.** DevOps accelerates the pace of application development, which creates a commensurate increase in security risks. By taking a DevSecOps approach, companies can apply security policies and procedures at each stage of the development process.

3. **Bring risk criteria into the process.** CISOs must develop a system for evaluating each new application on the basis of its technology risk. Then, DevSecOps teams will calibrate the application of security policies based on the severity of the technology risk.

4. **Add business requirements into the mix.** In addition to technology risk, companies should factor in business risk as they determine the overall risk profile of an application. Factors to consider include: compliance requirements, data protection requirements and the impact to the business if something goes wrong.

A risk-based approach to DevSecOps enables an enterprise to apply security policies to each new application based on a clear-eyed understanding of how much risk the company is willing to accept. Technical debt is made visible and managed in a controlled manner. And the use of these security measures early in the process has a cumulative effect, creating a more stable and secure development pipeline for future applications.

Learn more at www.dxc.technology