Three Strategies for a Successful DevSecOps Implementation

The DevSecOps methodology movement began in response to security concerns with the DevOps methodology. DevOps revolutionized the software industry by enabling high-speed software releases through technological and cultural changes. It focuses on automation and integration of processes, but the manual information security processes that traditionally created security assurance in a software release do not fit into this new methodology. Hence, DevSecOps emerged to address the security issues. The experience of applying this methodology resulted in the identification of three key success factors for a DevSecOps implementation.

The Path to DevSecOps

DevOps is defined as a "combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity."¹ DevOps does not treat the development and operations teams as separate entities. Instead, DevOps blurs the lines between the two entities, resulting in greater harmony and alignment, without compromising quality. Estimates say the global market for DevOps will reach US$12.85 billion by 2025.²

DevOps is simply the natural evolution of the Agile methodology, which aims to counter the flaws in a traditional waterfall software development life cycle (SDLC) approach. The Agile movement focuses on improving efficiency by releasing software frequently, in feedback-focused iterative loops. DevOps focuses on automation and integration of the processes that happen before and after the software release into production, such as development, build, test, compilation and monitor. DevOps typically relies on popular open-source tools for a continuous integration/continuous development (CI/CD) pipeline that makes rapid feedback and deployment of code possible and enables enterprises to achieve multiple high-speed software deployments daily.

Although reduced time to market makes business and product teams very happy, cybersecurity professionals have concerns—vulnerable code, if not checked, can also be deployed faster. For example, a developer who hardcodes an application program interface (API) key into code can potentially push this insecure piece of code into a remote repository and trigger testing and publishing to production in minutes. Information security professionals traditionally relied on manual processes, such as change management, secure code reviews, scans and penetration tests, to inject security assurance into software release processes. The traditional way of security assurance cannot keep up and does not work in a CI/CD world; as a result, reengineering steps were needed.

These concerns led to the start of the DevSecOps methodology movement, which applies the same principles to cybersecurity that DevOps applies to traditional IT processes to improve efficiency and

Taimur Ijlal, CISA, CISSP

Is an information security professional with more than 16 years of experience in cybersecurity and IT risk management. He manages the information security portfolio for one of the largest payment solutions providers in the Middle East. Previously, Ijlal was the head of information security at Dubai Bank, where he was responsible for the strategic oversight of its cybersecurity framework and maintaining an information security management system (ISMS). He also set up IT audit and information security departments for leading Pakistan enterprises, including Bank Alfalah and Bank AL Habib Ltd. Ijlal speaks in various information security forums and has written articles for the e-security section of the Pakistan technology magazine Spider.
removes manual intervention wherever possible. DevOps aims to shift security to the left by incorporating security assurance at every stage of the CI/CD pipeline and making it as essential a success criterion as development and testing. This shift is achieved by building and automating security into the CI/CD pipeline and requires a cultural change that must be endorsed by top-level management to succeed.

Lesson 1—Developers Are the Security Team’s Friends

Traditionally, developers and cybersecurity professionals have been on the opposite ends of the security spectrum and viewed each other with some suspicion. Developers have tended to believe that information security professionals introduce roadblocks and delays at the last minute; cybersecurity professionals often have seen developers as the prime culprits who are responsible for introducing software vulnerabilities, due to their perceived tendency to take shortcuts when writing code. For a successful DevSecOps program, these relationships must be repaired.

Development staff should be seen as an extension of information security staff—not a separate silo within the enterprise. The information security team should identify and train security champions on good security practices and automation tools, which can give them early feedback on security weaknesses within their software. These local security experts then become the voice of information security and can solve common security roadblocks quickly, recommend tools for secure source code reviews and automated vulnerabilities scanning, give advice on vulnerabilities, and, when required, escalate matters to the enterprise security team. As developers become more comfortable sharing security information within their team, a culture of awareness and information sharing advances within DevSecOps. A DevSecOps program may include incentives to encourage buy-in to the new culture, such as awards and prizes for the developers who identify and fix the greatest number of security bugs in a software release before it goes into production. A spirit of teamwork can be fostered further by setting up collaboration channels for quick informal feedback chats between the information security and development teams and sharing a security knowledge base to help demystify common security vulnerabilities. Enterprises that are serious about DevSecOps must invest in security training and development as heavily for local security champions as for cybersecurity staff.

Lesson 2—Continuous Integration Needs Continuous Security

In a traditional software release, the information security team is an independent entity that provides objective feedback and runs security tools at various stages in the life cycle to provide feedback on vulnerabilities and insecure code. However, these actions are done manually and must pass multiple security clearance gates before software can be deployed. The traditional tools cannot keep up with the fast pace of DevOps software releases, resulting in missed basic security hygiene checks and vulnerabilities that are pushed into production. For the traditional procedures and tools to be able to work in a CI/CD world, the security team would allow the DevOps team access to the security tools to make the tools an integral part of the DevOps pipeline, similar to the automated build, compilation and testing tools. For example, every new code check-in should immediately trigger a secure code review API to give the development team immediate feedback on any security vulnerabilities. Similarly, a successful software build should trigger an automated API job for application vulnerability scanning and pass/fail the build based on predefined criteria that are agreed on by the teams.
This scenario shows that the traditional way of logging in to a console and initiating security scans/code reviews is not feasible in DevOps; these tasks must be transformed into consumable services that can be called on demand via API calls by the DevSecOps team.

Information security professionals need to ensure that tools are compatible with CI/CD tools and can be used in a DevSecOps environment. The policies, procedures and service level agreements (SLAs) for software security assurance should also be reviewed to ensure that they can be adapted to work in a high-speed CI/CD world.

Lesson 3—Bug Bounty Programs Must Become Business as Usual

Many enterprises use continuous penetration testing as a tool to assess the security of their software products after major changes. Although this tool works well in a traditional environment, manual penetration tests do not scale well in a DevOps environment and need to be supplemented with bug bounty programs to be truly effective. Every daily release of code can contain vulnerabilities that may have been missed by security tools and tests earlier in the pipeline. Bug bounty programs crowdsource and incentivize the discovery of software vulnerabilities and offer cash rewards (bounties) to security researchers who find and report vulnerabilities. Although bug bounty programs are regularly used by large enterprises, including Facebook and Microsoft, bug bounty programs should become the new normal and a standard tool for all enterprises invested in DevSecOps. With a bug bounty program, enterprises get the assurance that every change to their software is subjected to a wide variety of intensive tests by numerous security professionals with vast and cumulative experience. A bug bounty program extends and complements (but does not replace) the enterprise standard vulnerability scanning and penetration testing exercises.

DevOps Is Critical to Evolution

The previously mentioned strategies are a few of the key success factors for a DevSecOps program, but they only are a starting point and not an exhaustive list. The era of high-speed software deployments is here to stay. A quick fix to implement DevSecOps does not exist. Enterprises must invest significant time and effort in changing their culture, tools and staff skill set to adapt and get the best results from DevOps while remaining secure. Information security professionals must realize that, more than any tool or technology, their mind-set must evolve to survive and remain relevant in a DevOps world.

Endnotes