DevSecOps: How to Seamlessly Integrate Security Into DevOps

Published: 30 September 2016 **ID:** G00315283

Analyst(s): Neil MacDonald, Ian Head

Information security architects must integrate security at multiple points into DevOps workflows in a collaborative way that is largely transparent to developers, and preserves the teamwork, agility and speed of DevOps and agile development environments, delivering "DevSecOps."

Key Challenges

- DevOps compliance is a top concern of IT leaders, but information security is seen as an inhibitor to DevOps agility.
- Security infrastructure has lagged in its ability to become "software defined" and programmable, making it difficult to integrate security controls into DevOps-style workflows in an automated, transparent way.
- Modern applications are largely "assembled," not developed, and developers often download and use known vulnerable open-source components and frameworks.

Recommendations

Information security architects should:

- Start with secure development and training, but don't make developers become security experts or switch tools.
- Embrace the concept of people-centric security and empower developers to take personal responsibility for security compensated for with monitoring. Embrace a "trust and verify" mindset.
- Require all information security platforms to expose full functionality via APIs for automatability.
- Use proven version control practices and tools for all application software and, equally as important, for all scripts, templates and blueprints used in DevOps environments.
- Adopt an immutable infrastructure mindset where production systems are locked down and changed via development.

Table of Contents

List of Figures

Strategic Planning Assumptions

By 2019, more than 70% of enterprise DevOps initiatives will have incorporated automated security vulnerability and configuration scanning for open source components and commercial packages, up from less than 10% in 2016.

By 2019, more than 50% of enterprise DevOps initiatives will have incorporated application security testing for custom code, up from less than 10% in 2016.

By 2019, more than 60% of DevOps initiatives will have adopted version control and tight management of infrastructure automation tools, up from less than 5% in 2016.

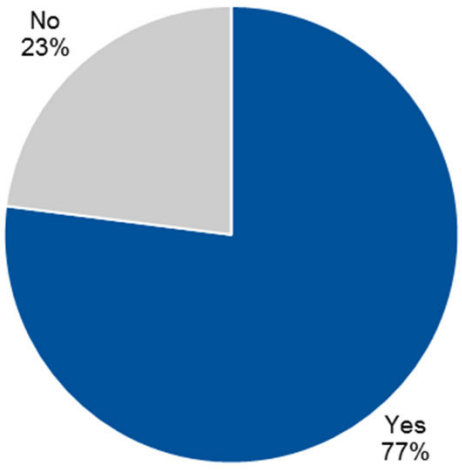
Introduction

In 2012, Gartner introduced the concept of DevSecOps (originally "DevOpsSec"; see Note 1) to the market in "DevOpsSec: Creating the Agile Triangle." In this research note, we identified the need for information security professionals to become actively involved in DevOps initiatives and to remain true to the spirit of DevOps, embracing its philosophy of teamwork, coordination, agility and shared responsibility.

Based on hundreds of discussions with clients, we estimate that fewer than 20% of enterprise security architects have engaged with their DevOps initiatives to actively and systematically incorporate information security into their DevOps initiatives; and fewer still have achieved the high degrees of security automation required to qualify as DevSecOps. Using the best practices outlined in this research, we believe security architects can optimize and improve their overall security posture by designing a set of integrated controls to deliver DevSecOps without undermining the agility and collaborative underpinnings of the DevOps philosophy.

However, surveys at Gartner's data center and information security summits in 2015 indicate that information security is viewed as an inhibitor to the agility and speed required by digital business and DevOps initiatives. Both information security professionals (Figure 1) and IT operations professionals (Figure 2) were surveyed.

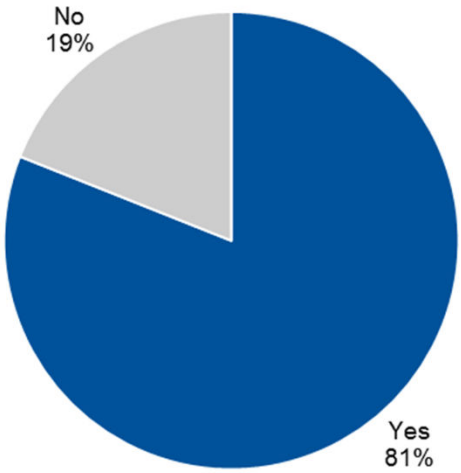
Figure 1. Information Security Professionals: Do You Believe Your Information Security Policies/Teams Are Slowing IT Down?



n = 41

Source: Gartner (September 2016)

Figure 2. IT Operations Professionals: Do You Believe Your Information Security Policies/Teams Are Slowing IT Down?



n = 93

Source: Gartner (September 2016)

As shown in Figures 1 and 2, both information security and IT operations professionals, in nearly identical ratios (approximately 4 to 1), believe information security is slowing down IT's ability to respond to the needs of the business.

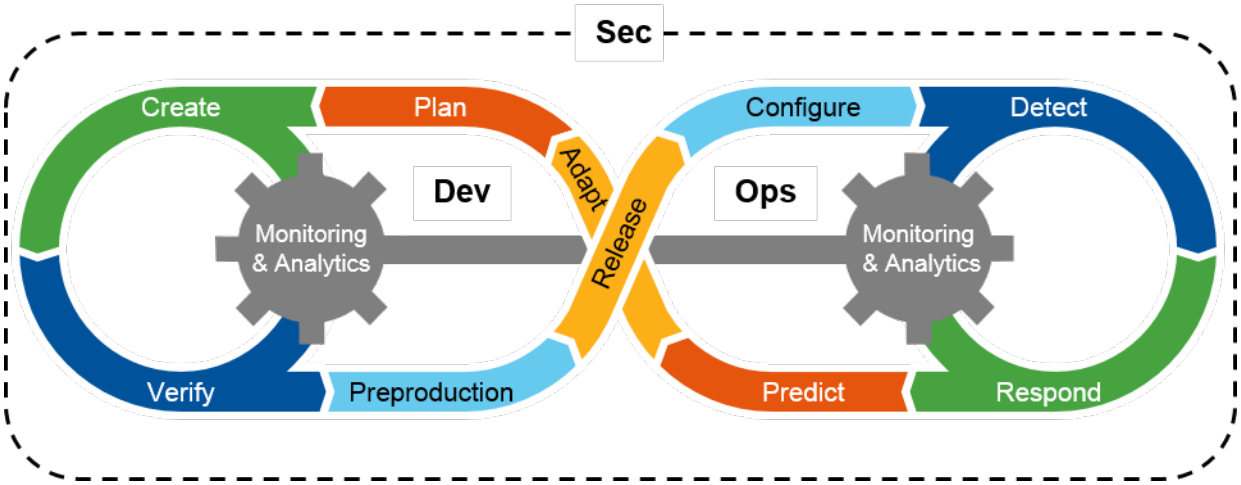
At the time of the original DevOpsSec research in 2012, DevOps was relatively new. However, recent Gartner research[1] indicates that 38% of enterprises are now using DevOps, and 50% will be actively using it by the end of 2016. In the same survey, security and audit tools represented the single highest-rated category of tools in terms of importance to an effective and efficient DevOps implementation, and 82% of respondents indicated that they had to deal with one or more regulations in their DevOps initiatives.[1] The good news is that DevOps teams understand that security and compliance are necessary. Now is the time for security architects to engage these teams and apply the best practices identified in this research.

Analysis

Security Controls Must Be Programmable and Automated Wherever Possible

DevSecOps can be depicted graphically as the rapid and agile iteration from development (the left side of Figure 3) into operations (the right side of Figure 3), with continuous monitoring and analytics at the core.

Figure 3. DevSecOps



Source: Gartner (September 2016)

Our goal as information security architects must be to automatically incorporate security controls without manual configuration throughout this cycle in a way that is as transparent as possible to DevOps teams and doesn't impede DevOps agility, but fulfills our legal and regulatory compliance requirements as well as manages risk. Security controls must be capable of automation within DevOps toolchains in order to enable this objective. This is important for two reasons. First, automation reduces the chance of misadministration and mistakes, which are leading causes of operations incidents, unexpected downtime and successful security attacks. Second, high levels of automation eliminate the need for involvement from a security professional to manually configure a security setting using a security console (and thus impacting the agility of DevOps environments). When security platform capabilities — such as identity and access management (IAM), firewalling, vulnerability scanning, application security testing and so on — are exposed programmatically, the integration and automation of these security controls are enabled throughout the DevOps life cycle in automated toolchains. Information security sets the policies, which can then be applied programmatically based on the type of workload. However, many security vendors are behind in their ability to be driven programmatically, and require a trained person to go to their console, or only a portion of their security functionality is exposed via APIs.

Specific best practices:

- Require security and management vendors to:
  - Fully API-enable their platform services and expose 100% of functionality via APIs
  - Provide explicit support for common DevOps toolchain environments, such as Chef, Puppet and similar automation tools
  - Provide explicit support for containers and container orchestration and management systems (which are not necessary for DevSecOps, but help streamline service delivery from development into production).

Use IAM and Role-Based Access Control to Provide Separation of Duties

As new and updated services cycle through the iterative DevSecOps process shown in Figure 3, auditors and security architects will want to have clear separation of who can do what, as well as where and when, in terms of service development and deployment. Even though a single team may be responsible, different people in the team will assume different roles. The scope of their capabilities can be managed by linking with existing IAM systems, and defining different roles for development versus preproduction versus production. The goal is not to tightly lock down what individuals can do. Indeed, these identities and roles will be highly empowered within the phase of DevSecOps for which they are responsible.

Specific best practices:

- Link to existing IAM systems for identities and permissions such as Active Directory or Lightweight Directory Access Protocol (LDAP). Require tooling vendors to integrate with these systems for access control. Security policies are enforced in the tools, and all tooling access and activities are monitored.
- Define and assign different required roles for development versus production. Ideally, no person directly touches the live environment, except via scripts and APIs.
- Mandate that the product team is responsible and auditable for its product changes on a "trust and verify" basis. Verification can be achieved by use of audit logs and configuration repositories, such as Git.

Implement a Simple Risk and Threat Model for All Applications

Basic risk-based threat modeling should be a standard best practice for DevSecOps. Start with a simple questionnaire for developers that can assess the risk of the service at a high level (see Note 2). For example, is sensitive data being handled? What type of sensitive data? Are communications being encrypted? Is data at rest being encrypted? Basic security best practices in coding should be communicated and reinforced with developer training. Examples of best practices include whitelisting and sanitization of all input from users and APIs to ensure correct syntax, encrypting all network communications when communicating externally, and encrypting all data at rest if stored in public clouds.

Specific best practices:

- Train developers in secure coding best practices and how to write resilient code that sanitizes input and blocks common attack patterns, such as buffer overflows, SQL injection and cross-site scripting.
- Develop a simple threat-and-risk model assessment tool and implement it as a part of the planning and design process. Base the level of threat modeling on the risk of the application. Applications handling sensitive data or directly accessing the internet should require deeper threat modeling and collaboratively involve information security.
- Plan to mask, de-identify or synthesize data used in development for testing. Do not use sensitive production data.

Scan Custom Code, Applications and APIs

Custom code should be scanned for security vulnerabilities in development. However, traditional static application security testing (SAST) and dynamic application security testing (DAST) are too heavyweight, complex and need to be run by a security professional (see "Magic Quadrant for Application Security Testing"). This approach won't work and won't scale for DevSecOps.

One best practice is to train developers to adopt a lightweight "spell checker" type scanning tool for quick checks of security within the developer's integrated development environment (IDE) as they develop code. Automated scanning and security test software should be part of the continuous integration test toolchain. Use interactive application security testing (IAST; see "Critical Capabilities for Application Security Testing") if the application development is being performed on a platform that supports instrumentation such as Java, .NET and PHP. IAST is well-suited for the highly automated testing needed for DevSecOps. If IAST isn't possible, use application security testing (AST) tools and services that can be fully automated without requiring a security professional to be involved. For example, DAST tools can be driven automatically using Selenium scripts, or SAST scans can be triggered automatically from scripting tools. AST-testing-as-a-service providers are a possibility, but only if the SLA of the AST services provider meets enterprise SLA requirements (for example, a guaranteed 24-hour turnaround).

Specific best practices:

- Evaluate and adopt IAST for applications that support it, and favor solutions using self-inducers for automated testing.
- Plan to fully automate any traditional static or dynamic tools or services that are used. For example, DevOps toolchain scripting tools can invoke automated testing. Do not make developers leave their native environment and toolchains.
- If SAST and DAST solutions are used, require vendors to support differential scans that test only the modified code and downstream-impacted modules.
- Acknowledge and accept that having zero vulnerabilities isn't possible. Reduce false positives (albeit with a risk of higher false negatives) and trim the output of AST tools and services to focus developers first on the highest severity, highest confidence vulnerabilities. Favor AST scanning tools and services that use machine learning and collective intelligence to trim results to only the highest confidence results.
- By policy, don't allow custom code with known critical vulnerabilities to enter production. Accept that vulnerabilities that represent lower levels of risk may or may not be addressed in future iterations. Approaches that identify and accept manageable risk are necessary.
- Work with DevOps managers to measure and motivate development teams to produce code with fewer vulnerabilities. Make security metrics a part of code quality metrics and hold development teams accountable.

Scan for OSS Issues in Development

Modern applications are more accurately described as "assembled" versus "developed from scratch." Many developers download code from open-source software (OSS) repositories such as Maven and GitHub. The issue is that often developers (knowingly or unknowingly) download known vulnerable OSS components and frameworks for use in their applications. Sonatype estimates 6% of downloads from Maven are of known vulnerable components and, in an earlier study, found that 71% of production applications contained at least one OSS component with known security flaws classified as "severe" or "critical."[2] Gartner refers to this category of source code scanning tools and services as "software composition analysis" (SCA; see "Hype Cycle for Application Security, 2016" and Note 3).

Specific best practices:

- Prioritize OSS software module identification and vulnerability scanning in development in 2016 and 2017.[3]
- Scan all applications, system images, virtual machines and containers in development for unknown, embedded or vulnerable OSS components in the operating system, application platform and in the application itself.
- Implement an "OSS firewall"[4] to proactively prevent developers from downloading known vulnerable code from Maven, GitHub and other OSS code repositories by policy.

Scan for Vulnerabilities and Correct Configuration in Development

The need to scan for known vulnerabilities goes beyond custom code and OSS as discussed in the previous sections. As packages are created and integrated, the entire content of all images (virtual machines [VMs], Amazon Machine Images, containers and similar constructs) should be scanned in development for vulnerabilities at the OS, application platform and commercial off-the-shelf software layers. The scan should also include correct configuration of the settings of the OS and application platform according to industry standard best practices for secure configuration and hardening guidelines. Gartner estimates that through 2020, 99% of vulnerabilities exploited will continue to be ones known by security and IT professionals for at least one year (see "Predicts 2016: Threat and Vulnerability Management"). Most "advanced" attacks come from attacks on known vulnerabilities. Preventing vulnerable systems from being released into production addresses the issue at the source.

Specific best practices:

- Architect DevOps processes to automatically scan the contents of all system images, including the base OS, application platform and all containers for known vulnerabilities and configuration issues as part of the continuous integration process. By policy, don't allow systems to leave development with known critical vulnerabilities.
- Require developers to remove unnecessary modules and harden all systems to industry standard best practices.

- Integrate with anti-malware scanners (such as VirusTotal), network sandboxing and algorithmic malware detection (such as Cylance) to scan systems to ensure malicious code hasn't been introduced to the image during the development process.

Treat Scripts/Recipes/Templates/Layers as Sensitive Code

In highly agile, DevOps-style deployments and in software-defined data centers, "infrastructure is code" — meaning, infrastructure is programmable and capable of being deployed and configured using automation. Security infrastructure is also becoming programmable (see the very first best practice of this research note). If infrastructure is becoming code, then secure coding principles must also apply to the templates, scripts, recipes and blueprints that drive the automatic configuration, and the repositories that hold the code that controls the infrastructure must be secured. Earlier we discussed how high levels of automation can reduce the chance of a mistake. However, a poorly written or mismanaged script can magnify a mistake if released into production. Configuration files and scripts, like source code, should be scanned for mistakes, vulnerabilities and excessive risk. Any item of infrastructure that is configurable using text files can potentially have its configuration files centrally held in a repository such as Git. This then allows all changes to those configurations to be recorded, including what change was made, when and by whom. Eventually, all the infrastructure configuration files may be treated like application source code, with full version control and rollback, as well as full auditing, logging and alerting on their usage. It takes time to roll out such practices to encompass the entire infrastructure. However, when implemented, this practice means that no changes to the infrastructure can be made that are not fully recorded and auditable. This is key to ensuring that internal and external auditors acquiesce in the expansion of DevOps across the application and infrastructure estate.

Specific best practices:

- Ensure that DevOps teams have implemented good version control practices and tools to maintain clear accountability and traceability for all the application software that is deployed into the live environment.
- Extend the scope of the version control and automated deployment tools to the configuration, infrastructure setup and monitoring configuration.
- Use automation scripts to deploy to the staging environment for final tests (may be an automated test in advanced DevOps environments).
- Scan scripts for errors and embedded risk, such as embedded credentials, encryption keys, API keys and so on, that represent significant and avoidable risk.

Measure System Integrity and Ensure Correct Configuration at Load

As we shift our discussion of DevSecOps best practices into production (the right side of Figure 3), the first priority must be to ensure that the system and services we are loading and running are indeed what we expect them to be, and that they are configured at runtime correctly. A smart attacker might attempt to tamper with images or layers in development. Anticipating this, set a goal to measure all system elements possible, including the hardware and virtualization layer (on systems

you own), VMs, OS images, and containers. This should include validation of container assembly layers used in container management systems.

Specific best practices:

- Implement system integrity measurement on systems as they are booted, including hardware-based root of trust measurements of the basic input/output system, bootloader, hypervisor and OS on systems you own.
- Store VMs at rest encrypted and hashed, if VMs are used in the DevSecOps workflow. Verify against tampering at boot.
- Use a container management system (if containers are used) that supports hashing or other techniques to measure and verify system integrity when loaded.

Use Whitelisting on Production Systems, Including Container-Based Implementations

To prevent breaches, one of the most powerful information security controls for a running workload is whitelisting and enforcement of all of its system interactions (see "Market Guide for Cloud Workload Protection Platforms"). The use of whitelisting to control what executables are run on a server provides a powerful security protection strategy. All malware that manifests itself as a file to be executed is blocked by default. However, whitelisting can extend well beyond just what executables are allowed to launch on a system. Examples include whitelisting of network connectivity, user access, administrative access, file system access, middleware/PaaS access, and processes. Historically, full whitelisting has been difficult to implement. However, the automated runtime whitelisting of workloads and services is straightforward, due to the declarative nature of DevOps templates, recipes, scripts and container manifests, and several vendors are taking advantage of this to lock down DevSecOps workloads.

Specific best practices:

- Disable runtime-signature-based anti-malware scanning and implement a whitelisting model on server workloads. Antivirus scanning provides little or no value on well-managed servers, and is a waste of resources in a DevSecOps environment.
- Automatically configure whitelists from the declarative sources of DevOps tool chains and containers.
- Require vendors to support whitelisting approaches for containers, if containers are used.[5]

Assume Compromise; Monitor Everything; Architect for Rapid Detection and Response

In a world of advanced and targeted attacks, perfect prevention isn't possible (see "Prevention Is Futile in 2020: Protect Information Via Pervasive Monitoring and Collective Intelligence"). Workloads and services must be continuously monitored for indications of unusual behavior that would be

indicative of an active breach. Increasingly, these approaches use advanced analytics and machine learning to identify patterns of interest.

Specific best practices:

- Design for pervasive monitoring of critical applications — user logins/logouts, transactions, interactions, network activity and system activity.
- Use the monitoring data to establish baselines of "normal" for the application in order to detect meaningful deviations. Share monitoring data across DevOps or product teams, platform teams and security operations center teams, as unusual activity may be caused by a hardware failure, software failure, bug, insider threat or attack.
- Deploy deception and decoy services automatically to more easily identify attackers as these technologies mature over the next several years.

Lock Down Production Infrastructure and Services

Security architects should work with IT operations to lock down servers and infrastructure so that automated tools are the only way to make changes to workloads in production, if and when a response is needed. This should be your target standard mode of operations, as rapid iteration and addressing vulnerabilities in development will improve your overall security posture (see "How to Make Cloud IaaS Workloads More Secure Than Your Own Data Center"). While it will take time for most organizations to implement across the entire infrastructure, this approach offers a higher level of protection of workloads in production than "separation of duties" alone (see Note 4). Where DevSecOps practices are employed, the role of the change advisory board in product releases evolves to that of a scheduling function, rather than a body that grants permission to deploy.

Specific best practices:

- Information security architects should collaborate with DevOps teams to:
  - Restrict changes to only being made via automated tools and scripts. Disable remote administration via Secure Shell (SSH) and Remote Desktop Protocol (RDP) to force access via APIs and scripts.
  - Adopt an immutable infrastructure mindset (where possible) and automate all changes to the environment using DevSecOps-style workflows. Out-of-date workloads should simply be replaced with newer images in an automated, systematic way (see "How to Make IaaS Workloads More Secure Than Your Own Data Center").
  - Require privileged access management systems to manage credentialed access (see "Market Guide for Privileged Access Management") in the rare cases when direct administrative access is needed.

If Containers Are Used, Acknowledge and Address the Security Limitations

Containers are not required for DevOps, but they are extremely popular in DevOps environments because of the consistency and streamlining they provide from development into production for the developer. However, containers introduce several security issues that must at least be acknowledged. Containers share a common OS, so it is the OS that is providing isolation, not the hypervisor. Without the use of additional tools, network traffic is visible to all hosted containers sharing the same OS. A successful attack on the OS kernel layer exposes all containers. This is why we recommend, as a best practice, using containers on only workloads of similar trust levels; further, we recommend using hypervisors or physical separation when stronger isolation is needed. A full discussion of container security is outside scope of this research, and is discussed in "Virtual Machines and Containers Solve Different Problems" and "How to Secure Docker Containers in Operation."

Bottom Line

DevSecOps is an objective where security checks and controls are applied automatically and transparently throughout the development and delivery of IT-enabled services in rapid-development DevOps environments. Simply layering on standard security tools and processes won't work. Secure service delivery starts in development, and the most effective DevSecOps programs start at the earliest points in the development process and follow the workload throughout its life cycle, as shown in Figure 3. Even if you aren't actively using DevOps, the best practices identified in this research note will apply to any security architect looking to accelerate the development and delivery of IT-enabled services using agile development, test-driven development or other methodologies.

If you haven't already, get involved in DevOps initiatives and start pressuring all security vendors for full programmability of their services for automatability. For 2016, begin the immediate scanning of services in development for vulnerabilities, and make OSS software module identification, configuration and vulnerability scanning a priority in 2016. Make custom code scanning a priority in 2017. Longer term, automate security controls wherever possible to reduce the chance for misconfiguration, mistakes and mismanagement.

Above all, successful DevSecOps initiatives must remain true to the original DevOps philosophy: teamwork and transparency, and continual improvement through continual learning.

Gartner Recommended Reading

Some documents may not be available as part of your current Gartner subscription.

"Embrace DevOps Product Teams to Turbocharge Your I&O Organization and Control Costs"

"Avoid Failure by Developing a Toolchain That Enables DevOps"

"Hype Cycle for Application Security, 2016"

"DevOpsSec: Creating the Agile Triangle"

"Security in a DevOps World"

"Leveraging the DevOps Toolchain to Automate and Secure Virtualization, Private Cloud and Public Cloud Environments"

Evidence

[1] Gartner Enterprise DevOps Survey Study: This research was conducted via an online survey from 9 May to 13 May 2016 among Gartner Research Circle Members — a Gartner-managed panel composed of IT and business leaders.

Objectives: To learn how organizations are adopting DevOps as a means to accelerate enablement; to go faster while improving quality; additionally, to inform on topics, including starting a DevOps approach, pitfalls to avoid, scaling efforts, integrating information security, pursuing this in a regulated environment and quantifying benefits.

In all, 252 IT and business leaders participated, with 95 members qualified by indicating they are already using DevOps.

[2] "2016 Report on the State of the Software Supply Chain," Sonatype; F. Rashid, "71 Percent of Applications Use Components With Severe or Critical Security Flaws: Report," Security Week, 30 April 2013.

[3] Vendors offering OSS scanning capabilities: Black Duck; HPE (partners with Sonatype); IBM (partners with Black Duck); OpenLogic; Palamida; Sonatype; Synopsys (acquired Protecode); Veracode

[4] Nexus Firewall: OSS firewall example from Sonatype

[5] Whitelisting workloads in DevSecOps: Aqua Security; Apcera; Twistlock

Note 1 DevOpsSec Versus DevSecOps

When Gartner originally introduced the concept of "DevOpsSec" in 2012, it was quickly pointed out by the security community that the acronym "DOS" was more commonly associated with "denial of service." The term was therefore adjusted to "DevSecOps" in subsequent research and Hype Cycles. There are several terms used in the industry describing the need to integrated information security into DevOps processes: see DevSecOps.org. and Rugged DevOps.

The practices described in this document are best evangelized across the product teams by having a security community that encourages standards and common practices across all the DevOps product teams. How this community aligns within the larger organization is described in "Embrace DevOps Product Teams to Turbocharge Your I&O Organization and Control Costs."

Note 2 Threat-Modeling Tools

One Gartner client developed their own simple "adaptive" risk assessment questionnaire in-house. It was adaptive in the sense that if specific risky aspects of a service were identified, the questionnaire would ask more detailed and specific questions (such as the how sensitive data was being protected). With this assessment, in its simplest form for routine applications, the developer would answer about 25 questions. In an extremely risky case, with an application handling sensitive data and exposed to the public internet, there were nearly 200 questions.

The Open Web Application Security Project offers threat-risk-modeling guidance. In addition, several third-party threat modeling tools are available. Vendors offering threat-modeling tools include:

- Microsoft: Threat Modeling Tool 2016
- Microsoft: interactive game
- Security Compass
- MyAppSecurity ThreatModeler

Note 3 Software Composition Analysis

Multiple vendors offer this capability, and some AST vendors are bundling this capability into their offering. There are additional benefits to using SCA. One is to understand if any of the OSS components used fall under OSS licensing terms that are incompatible with the enterprise's legal requirements. Another significant benefit is that when an OSS component is later discovered to have a vulnerability, the enterprise security team can quickly identify which production applications are affected and prioritize their remediation. This avoids chaotic scrambles to scan every system when a critical vulnerability in a widely used component occurs (consider Heartbleed OpenSSL vulnerability).

Note 4 Role of ITIL in DevSecOps

Traditional ITIL-inspired change control techniques are still applicable to the underlying infrastructure and platform provided to the product team (for example, server hardware failure), and also where changes may impact a wider audience of products and stakeholders.

GARTNER HEADQUARTERS

Corporate Headquarters
56 Top Gallant Road
Stamford, CT 06902-7700
USA
+1 203 964 0096

Regional Headquarters
AUSTRALIA
BRAZIL
JAPAN
UNITED KINGDOM

For a complete list of worldwide locations,
visit http://www.gartner.com/technology/about.jsp