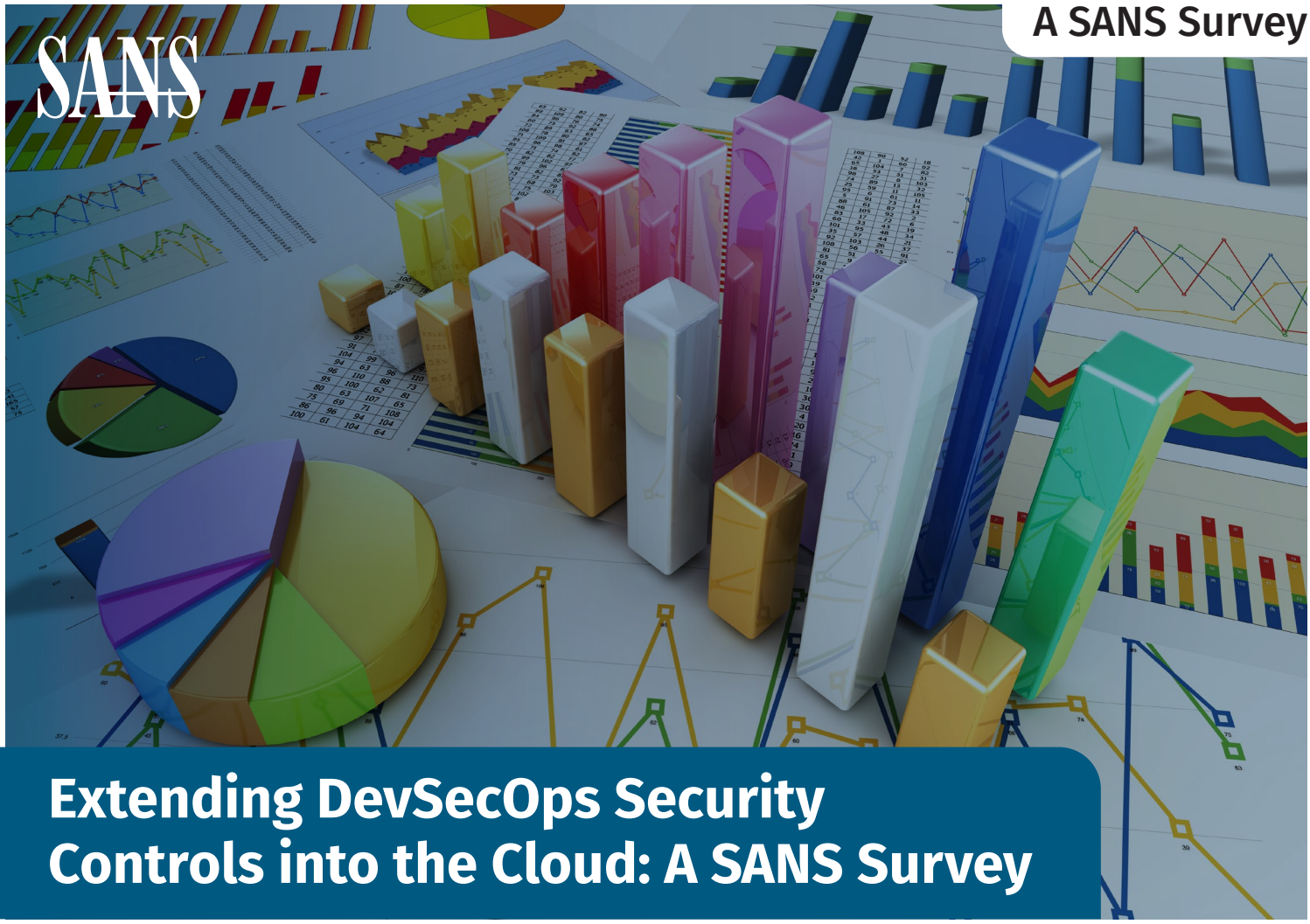
A SANS Survey



Extending DevSecOps Security Controls into the Cloud: A SANS Survey

Written by **Jim Bird** and **Eric Johnson**
Advisor: **Frank Kim**

October 2020

Sponsored by:

orca security

Analyst Program

Executive Summary

By moving work to the cloud, organizations can take advantage of the massive investments in infrastructure-as-a-service (IaaS) and platform-as-a-service (PaaS) engineering that cloud providers have made. Working in the cloud also creates new challenges and opportunities when it comes to managing security and compliance risks.

DevOps—extending Agile development values, techniques and tools from development to operations—has become the de facto model for IT in the cloud. DevOps consists of cross-functional teams of developers and operations engineers sharing responsibilities for building, deploying and running applications; leaning heavily on automated testing and build tooling; and letting the cloud provider do the undifferentiated heavy lifting of running the data center, provisioning infrastructure and operating generic platform services.

Secure DevOps, or *DevSecOps*, integrates security along the path, from requirements to architecture and design, coding, testing, release and deployment. This integration enables teams to get work done quickly while managing security risks in-phase. DevSecOps makes security a first-class problem—and the security team a first-class participant—in development and operations.

This survey, the seventh in an annual series that focuses on application security and DevOps, examines DevSecOps in the cloud to understand:

- How organizations are using the cloud in platforms, runtime architectures and development environments to identify security requirements, risks and opportunities.
- How organizations are building and deploying applications in the cloud—that is, understanding what Continuous Integration (CI) and Continuous Delivery (CD) technologies and practices are in use. The CI/CD pipeline is not just a software delivery mechanism; it also serves as a control plane that security and compliance teams can leverage to inject testing and controls, enforce policies, and build an end-to-end audit trail of changes.
- Whether or not security teams are able to keep up with fast-moving DevOps teams that deploy changes at high velocity. Have organizations been successful in *Shifting Security Left* into development?
- What security tools and practices give DevSecOps teams the most value, whether organizations are investing more in the Dev or Ops security domains, and if organizations are *Shifting Left* or *Shifting Right*.

DevOps: Shifting Left and Shifting Right

DevOps asks teams to shift left in order to introduce testing and reviews as early as possible. This approach reduces the cost and time related to finding and fixing problems, while also minimizing friction and delays.

Instead of waiting until analysis, design and coding have been completed before handing work off to QA and Security for testing and review, DevOps teams work in tight iterative and incremental loops, continuously testing and reviewing changes as the code is checked in. Shifting left puts responsibility and accountability for building high-quality, secure, working software directly onto the people who are writing the code.

Security can *shift left* by:

- Securing management buy-in on security priorities and compliance requirements
- Training developers in secure coding and embedding "security champions" into teams
- Finding security tools that are easy to use and fit naturally into development workflows and automated pipelines
- Helping DevOps teams implement these security tools

There will always be problems and risks that can't be understood and solved up front by developers looking through the tightly focused Dev lens of DevOps. These problems must be discovered and solved through a wider Ops lens. Shifting left isn't enough.

DevOps teams, and Security, also need to *shift right* by:

- Continuously testing and experimenting—evaluating security tools and practices to learn what is useful and what slows teams down and then conducting chaos testing and operational fire drills, penetration testing and security red teaming in live environments
- Continuously monitoring and using telemetry, collecting insight on operational vulnerabilities and attacks that pose real (not theoretical) risks to the organization
- Implementing automated compliance safeguards and runtime protection to enforce security policies and defend against rapidly changing threats

Figure 1 provides a snapshot of the demographics for the respondents to this survey.

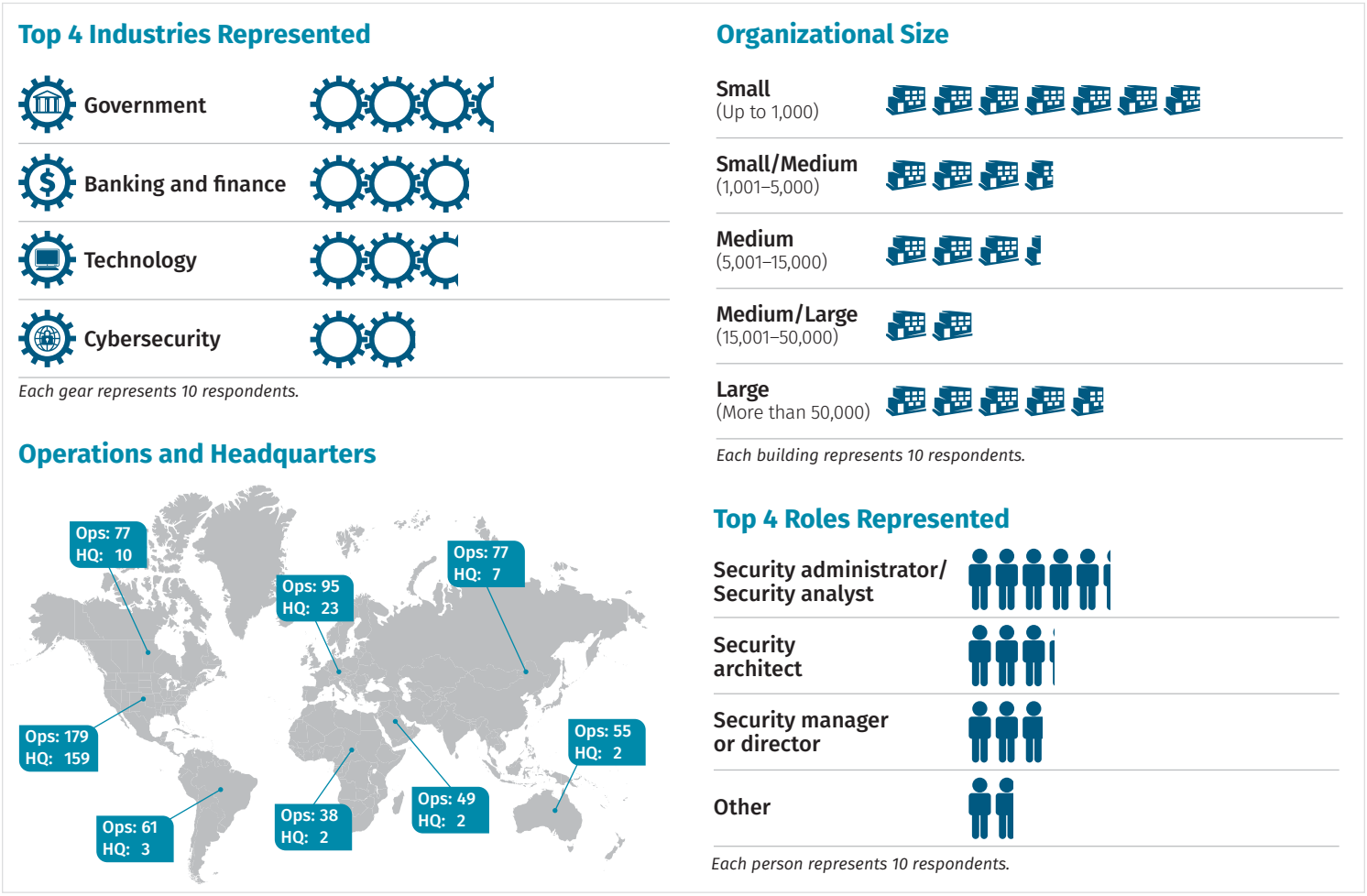


Figure 1. Survey Demographics

By extending DevOps to the cloud, organizations offload many of the responsibilities and risks for operations and scale to the cloud provider. As cloud platform offerings mature, organizations can also shift more responsibilities for security and compliance to the cloud platform itself. This enables organizations to take advantage of the cloud platform's capabilities and available cloud-based third-party services to reduce security risks and costs, as well as simplify their security and compliance programs.

However, this is not a simple lift-and-shift exercise. Organizations must take responsibility for architecting a secure solution, understanding and correctly using the capabilities that cloud providers offer, and identifying and filling in any gaps.

Key Findings

- While on-premises application hosting is still the most common means for delivery, cloud-hosted platforms are gaining traction. Yet many security professionals (36%) are spending less than 25% of their time building a "paved road" for the cloud provider platforms.

- Most organizations, especially large enterprises, need to work with multiple cloud platform providers, which means that they need to understand and manage a larger range of security and compliance risks. The majority of organizations (92%) use at least one public cloud provider, and the average organization has workloads running in 2.33 public cloud providers.
- Agile and DevOps methods are enabling developers to deliver features and changes faster and more cost effectively. The velocity of feature delivery has increased by 14% over the past four years, but the speed of security assessments is not keeping up. Only half of organizations are taking advantage of automated testing, and 27% are not doing any security testing at all.
- Most organizations are struggling to shift security left. Only 40% are including security assessments early in planning and design, where important decisions are made about architecture approach, development tooling and technology platforms—and where mistakes or misunderstandings can be dangerous and expensive.
- Successfully implementing DevSecOps is not a technical problem; it is an organizational problem. Lack of resources, lack of management and developer buy-in, bureaucracy, poor communication across silos and poor prioritization are holding organizations back. But organizations can compensate to some extent for these shortcomings by shifting more work and risk onto cloud providers who have the scale, capabilities and agility to respond.

Understanding the Cloud Landscape

Mapping out the cloud landscape—the extent of cloud services adoption, the cloud platforms' runtime architectures used and the related development environments—helps security understand the potential risks of working in the cloud and how these risks can be managed.

As DevOps teams move their workloads into the cloud, security teams are shifting right and learning how to apply operations, monitoring and runtime security controls across public cloud providers, such as Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP). A majority of security teams (64%) are now spending at least 25% of their time deeply involved in public cloud security and operational responsibilities (see Figure 2).

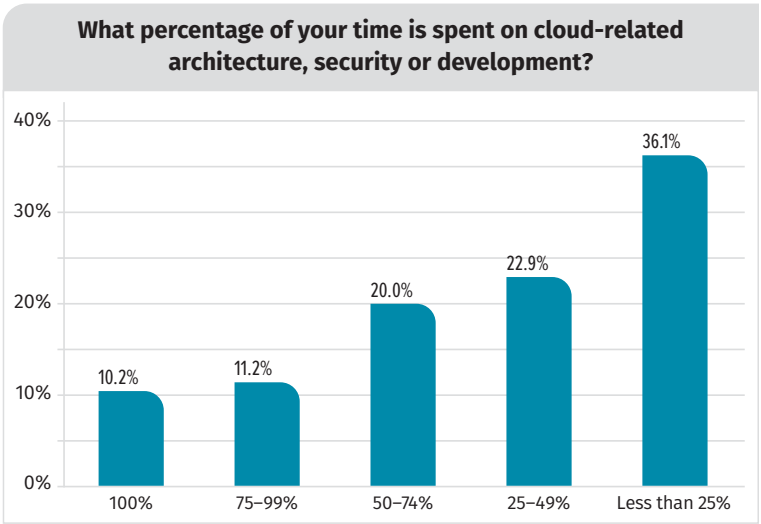


Figure 2. Time Spent on Architecture, Security or Development

Increasing Cloud Adoption

With public cloud adoption rising, organizations are slowly transitioning workloads from on-premises to cloud-hosted virtual machines, cloud-hosted container services and cloud-hosted serverless platforms (see Figure 3).

As organizations move from on-premises infrastructure to cloud-based services, operational complexity and scale, development cost, compliance responsibilities and security risk shift from your organization to the cloud platform:

- **On-premises—**The organization needs to manage everything, from soup to nuts.
- **Cloud virtual machines—**Responsibilities for data center management and infrastructure provisioning shift to the cloud provider.
- **Cloud container service—**Responsibilities for patching the virtual machine and hardening the container runtime shift to the cloud provider. DevOps teams focus on the application and its build/runtime dependencies packaged and shipped in a Docker or CRI-O image.
- **Cloud serverless—**Responsibilities for hardening container images and patching the underlying application runtime shift to the cloud provider. DevOps teams need to worry only about writing and testing application code, the permissions associated with the function's service account and major upgrades to the application runtime environment (e.g., `NodeJS 10.X` to `Node 12.X`).

Using Netflix's paved road metaphor, as organizations shift from on-premises to cloud-managed workloads, and as their responsibilities for operations and security decrease, the road is less paved. The technologies and architectures are more powerful, but less understood and less mature. Organizations need to make a trade-off between reducing development and operational complexity and cost, and taking on the security and compliance risks associated with each managed cloud provider.

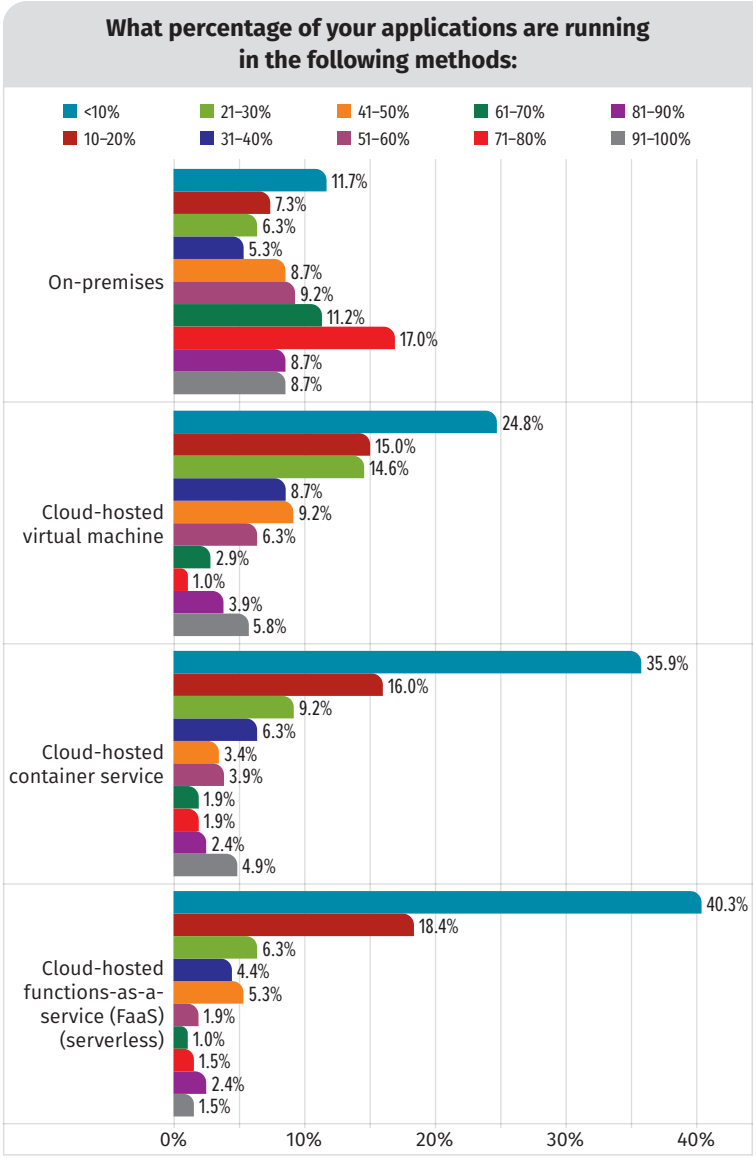


Figure 3. Workload Methods

Paving the Road for Developers

At Netflix, one of the key responsibilities of the security team is to build a "paved road" for developers and operations engineers to use when they are coding, provisioning and deploying online services. The paved road includes secure headers and templates, secure-by-default frameworks and general-purpose services, self-service security testing tools, and automated compliance enforcement for different types of applications and on different platforms.

When developers set off on a paved road, they know that they will be able to move faster and safer at the same time.[1]

[1] For more on Netflix's paved road, see https://medium.com/@NetflixTechBlog/scaling-appsec-at-netflix-6a13d7ab6043

While on-premises application hosting is still the most common means for delivery, cloud-hosted platforms are gaining traction. Yet many security professionals (36%) are spending less than 25% of their time building a paved road for the cloud provider platforms. Security professionals should prioritize and start evaluating the security risks and considerations for the different cloud platforms and architectures:

- **Cloud virtual machines—**This is the simplest transition from on premises, but it pushes more responsibility onto the organization to manage cloud operations, security and automation. Each cloud provider auto-provisions default network and firewall rules for new virtual machines, which can unintentionally expose private resources to the outside world.
- **Cloud container service—**Security needs to be managed all along the container life cycle. There are lots of points where security risks can be introduced, but also lots of tools available to help manage risks, from code linters to container image scanning, secure container repositories and runtime security defense.
- **Cloud serverless—**In this case, security shifts primarily to the application source code. Insecure code and referencing vulnerable open source libraries are the primary attack surface for externally facing functions.[2] Mature continuous integration and delivery is the only way to scale security and compliance with the volume of functions required to power an application or microservice.

Serverless Security Risks

Serverless platforms, also known as *functions-as-a-service (FaaS)*, are compelling for certain kinds of development and cloud security work. Examples include AWS Lambda, Google Cloud Functions and Microsoft Azure Functions.

In serverless environments, the majority of traditional security controls are cloud-managed:

- Patching for virtual machines and container runtimes.
- Minor updates to the application frameworks are automatically applied to the execution environment.
- Managed function service accounts are integrated into the platform.
- Service account credentials are automatically provisioned and rotated.
- Function audit logging and monitoring capabilities are integrated into the platform.

However, depending on the cloud provider, many default security settings are concerning:

- Warm start times (i.e., amount of time before the environment is recycled) vary from 3 to 12 minutes, depending on the provider. This creates an opportunity for attackers to store malware in the execution environment.
- Service account credentials expiration windows range from 30 minutes to 12 hours, depending on the provider.
- Network security controls (e.g., egress firewall rules and network logging) are disabled by default.
- Function service account permissions can be overly permissive. For example, Google Cloud Functions default to the Editor Role, which allows read and write access to existing resources, while AWS and Azure follow a least privilege model. Function permissions require disciplined automated and manual security review.

[2] See OWASP's Serverless Top 10 for common security risks: https://owasp.org/www-project-serverless-top-10/

Cloud Platform Analysis: The Big 3

The Big 3 cloud providers (AWS, Azure and GCP) continue to dominate the public cloud space. Of 211 respondents, only 59 (28%) reported running workloads in an alternative cloud provider. Of those 59 respondents, 30 (51%) indicated that less than 10% of their workloads are running in a cloud provider that is not one of the Big 3. See Figure 4.

Further analysis of the Big 3 providers supports the 2019 Gartner cloud infrastructure as a service report.[3] AWS, the first major player in the public cloud offering, is being used by 85% of the respondents, with 20% of these respondents hosting more than 91% of their applications in AWS.

Microsoft Azure continues to close the gap, with 84% of respondents using Azure. However, only 12% of those respondents depend heavily (more than 91%) on Azure for the running their workloads.

GCP remains a distant third, with only 65% adoption. Perhaps the most telling gap in the numbers is that only 4% of the GCP users depend on GCP for 91% of their workloads.

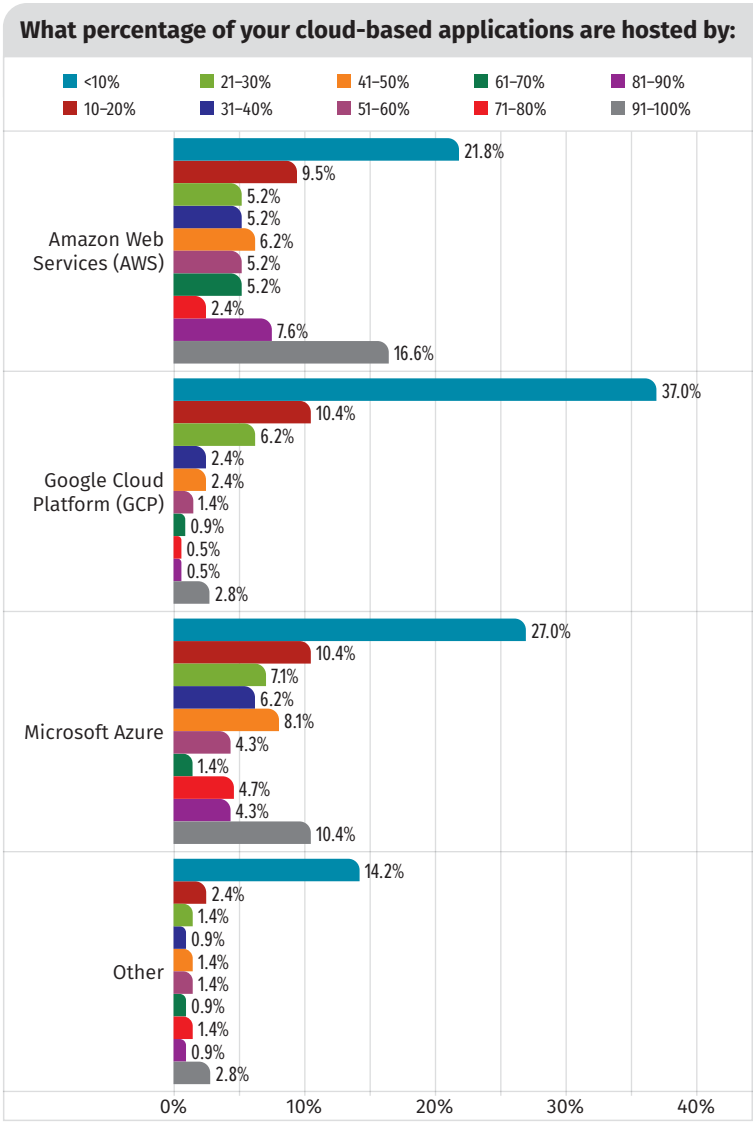


Figure 4. Cloud Hosting Providers

Benefits of Managed Container Services

Orchestrators such as Kubernetes play a critical role in the deployment, operations and *security* of container-based systems, especially at scale. But orchestrators introduce a new set of risks that need to be managed: operational complexity, large attack surfaces, misconfiguration and configuration drift, identity and access management (IAM), and network access management.

Containers (especially Docker) exploded in DevOps teams before cloud services for managing containers were widely available. Heavy use of on-premises orchestrators is a legacy technical debt problem. Organizations that had early success with containers had to put in

Most organizations, especially large enterprises, need to work with multiple cloud platform providers. SANS found that most organizations (92%) use at least one public cloud provider, with slightly more than 60% having workloads running on three or more public cloud providers, including AWS, Azure, GCP and a handful of others.

Multicloud is unavoidable:

- Organizations often choose the best service available, regardless of the provider (e.g., AWS Lambda, Microsoft Azure Active Directory, or Google Kubernetes Engine).
- Enterprises need to manage systemic risk by spreading work across multiple platform providers.
- Corporate mergers and acquisitions are a contributing factor to the rapidly increasing cloud inventory.

This presents a major challenge for security professionals. As DevOps teams shift right into multiple cloud providers, security teams must learn the security models, risks and configuration options for each provider's key service platforms.[4]

[3] "Magic Quadrant for Cloud Infrastructure as a Service, Worldwide," www.gartner.com/doc/reprints?id=1-1CMAPXNO&ct=190709&st=sb

[4] For more on multicloud security considerations, see www.sans.org/reading-room/whitepapers/cloud/top-5-considerations-multicloud-security-39505 [Registration required.]

place orchestration solutions as the use of containers scaled up. Docker Swarm was the easiest entry point for smaller teams and smaller systems, but the future of Swarm is uncertain.[5] For bigger teams and bigger systems, Kubernetes became the de facto standard.

Installing, configuring, hardening and managing an on-premises Kubernetes cluster is a heavy lift. For managing and orchestrating containers, it makes much more sense to shift responsibilities right, to a proven cloud provider.

On-premises hosted Docker, Swarm and Kubernetes installations require organizations to manage the underlying servers, container orchestrator, container runtime and container life cycle. Security teams are responsible for understanding and implementing the entire set of CIS Benchmarks for Kubernetes and Docker, each of which are more than 200 pages. For this reason, organizations are starting to shift right. Respondents reported less on-premises use of Kubernetes (32%) and Docker (35%) installations compared to the cloud-hosted options (42% and 43%, respectively). See Figure 5.

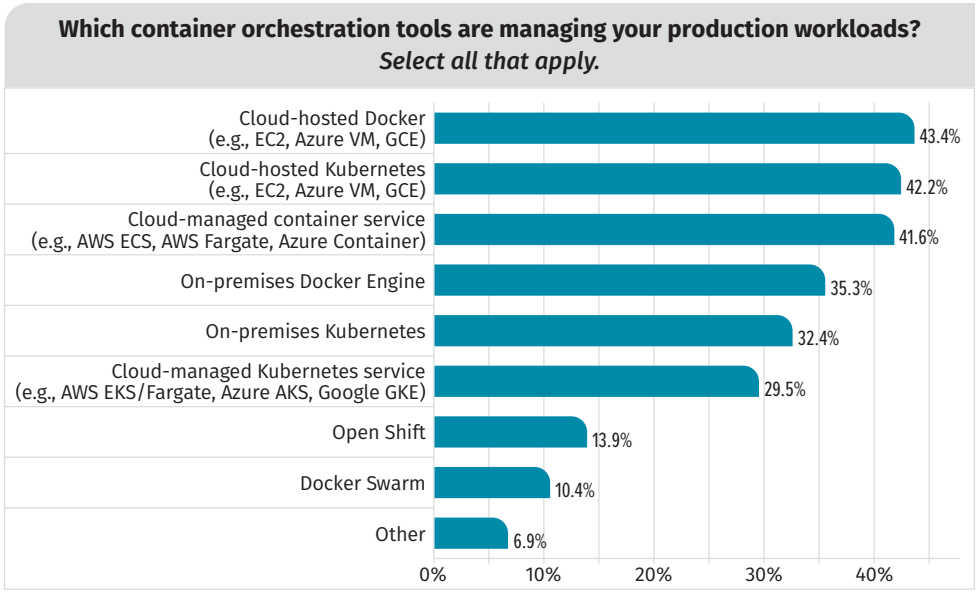


Figure 5. Container Orchestration Tools in Use

More organizations (42%) are avoiding the complexity and overhead involved with installing, managing and hardening Docker and Kubernetes services altogether. As an alternative, they are running in cloud managed container services such as AWS Fargate, Amazon Elastic Container Service (AWS ECS) and Microsoft Azure Container Service to reduce operational costs, time and security risks:

- Kubernetes services removes the Kubernetes administration control plane, such as server administration and hardening, from the organization's responsibility model. Once you provide the cluster's YAML definition, you can focus on the cluster config, execution role, image scanning, and setting up an admission controller.
- AWS Fargate is serverless container orchestration. There is no central server to manage and patch. Just provide a task definition with the image ID and the number of desired containers to manage workloads, and it manages the cluster.

Takeaway

When shifting right, security's job gets simpler and more focused. Let the cloud provider manage the technical complexity, operational risk, and lockdown and hardening of the containerized environment so that the DevOps teams can focus on delivering business features.

[5] Following its acquisition of Docker Enterprise and the enterprise customer base, Mirantis originally announced a sunset date for Swarm in 2021, but now says that it will continue to maintain it for the foreseeable future. www.mirantis.com/blog/mirantis-will-continue-to-support-and-develop-docker-swarm

Programming Environments and Risks

As teams move to different platforms, their use of development tools, languages and frameworks change to take advantage of new capabilities and conveniences. At the same time, this introduces new security risks. See Figure 6.

Security practices and tooling must be adapted to the needs of development teams as well as the development environments, languages and frameworks that these teams are using. While Java, PHP and C/C++ continue to be major sources of security risk based on their legacy use, common cloud platform languages such as JavaScript/Node.js and Python are increasing in use—and risk.[6]

Which languages and platforms in your application portfolio have been the greatest source of risk or exposure to your organization? ***Select your top three.***

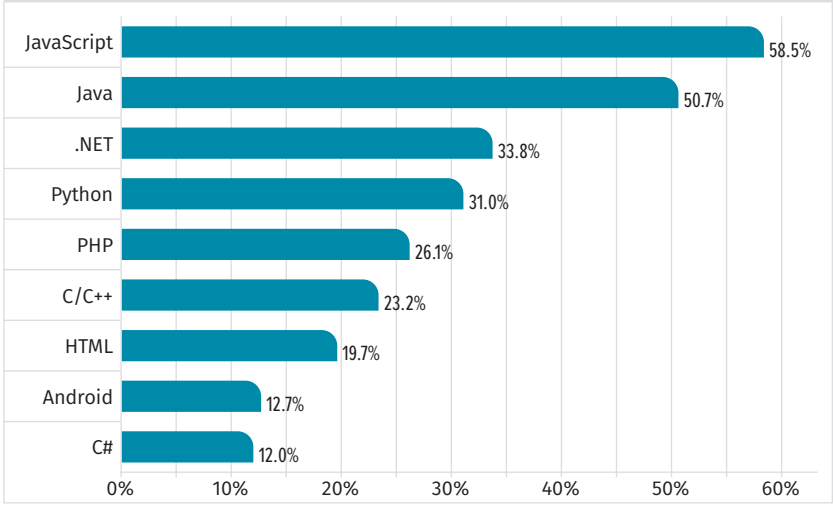


Figure 6. Riskiest Languages and Platforms

Security at Velocity

The increased velocity of delivery is key to DevOps success. Top-performing DevOps organizations deploy changes across their infrastructure dozens or hundreds or even thousands of times per day.

But working at high velocity forces organizations to make compromises—and introduces risks:

- **Agility vs. compliance—**Hand-offs to security specialists, compliance reviews and other manual checks waste time and disrupt flow. As teams speed up, security testing must become self-service, automated as much as possible, and integrated into engineering environments and workflows. Give engineering teams security training, clear and measurable goals, and testable compliance requirements. Let engineers choose when and where to enforce checks (e.g., in the IDE, during pull requests or in CI/CD as part of the build pipeline). Implement automated checks to enforce policies in development, testing and production.
- **Speed vs. coverage—**If teams are delivering changes several times a day, testing (including security testing) needs to happen in minutes, not hours or days. Teams cannot test the entire code base on every change and should not need to. Most changes in iterative and incremental development should be small and reasonably isolated. Focus reviews and testing on the areas of code recently changed, relying on incremental scanning back-stopped by automated smoke tests on critical functions and automated safety checks to catch common configuration errors.
- **Accuracy vs. completeness—**Tests must be accurate and consistently provide clear, actionable feedback to developers (e.g., where the bug is, how to fix the bug and why). Developers won't have the time or skills to assess false positive findings from scanning or testing.

[6] Security risks with languages track popularity of use. See GitHub's "State of the Octoverse" for tracking of development language use: https://octoverse.github.com

To understand how organizations are making these risk trade-offs, we asked a series of questions to learn:

- How often organizations are delivering changes to production
- How they are delivering these changes—that is, what technical practices and toolchains they are using
- How often they are testing or assessing security
- What security controls they are relying on to manage security risks

We describe the survey results and our findings related to these questions in the following sections.

Delivery Velocity Is Accelerating

The velocity at which organizations are delivering IT changes continues to accelerate, as shown in Table 1. This table provides a year-over-year comparison of how often respondents deploy system changes to production applications.

Table 1. Delivery Velocity

Frequency of Delivery to Production	2017	2018	2020
Continuously (several times per day)	5.3%	10.0%	**11.1%**
Daily	12.0%	7.0%	**12.1%**
Weekly	25.4%	24.0%	**31.4%**
More than once per month	17.7%	25.0%	**19.8%**
Monthly	18.7%	15.0%	11.1%
Quarterly	13.4%	11.0%	6.3%
More than once per year	3.8%	4.0%	3.9%
Annually	1.9%	1.0%	1.0%
Other (ad hoc/more than once per year)	1.9%	3.0%	3.9%

In the early days of Agile development, Scrum teams and XP teams delivered working software every month, which was considered radical at the time. Today, almost three-quarters (74%) of organizations are delivering changes more than once per month, an increase in velocity of 14% over the past four years.

To deliver changes at high velocity, DevOps teams follow a core set of common technical practices and automation technologies. Security and compliance can build on and leverage these practices and tools—if these practices and tools are in place and working effectively. See Figure 7.

More than half of organizations are following iterative, incremental design and development using core Agile and DevOps technical practices, such as CI and automated builds.

However, almost half of organizations are not using automated testing. This implies that they are still relying on manual testing (which doesn't scale and can't keep up with rapid, iterative development and delivery) or, worse, they aren't testing at all.

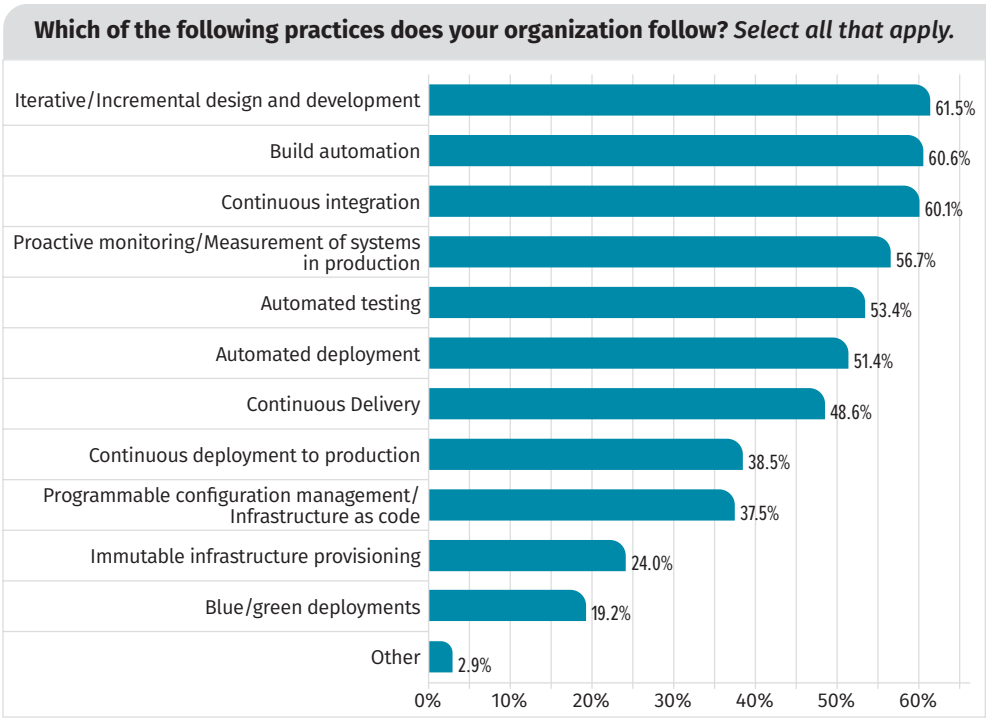


Figure 7. Technical Practices and Automation Technologies in Use

And only 38% are provisioning and configuring infrastructure using modern programmatic tools, such as Chef, Terraform or AWS CloudFormation, instead of point-and-click admin console interfaces. Examples include the following:

- Configuring through code, rather than manually, is a prerequisite for leveraging cloud services at scale. Code-based changes are versioned, testable, auditable and repeatable.
- Configuring cloud infrastructure in code creates a control structure for security and compliance. Configuration changes can be checked into version control, and automatically scanned and tested using CI/CD build pipelines to catch configuration errors and enforce hardening policies.

However, there is a high learning curve associated with understanding cloud services and cloud infrastructure coding languages, practices and tools at the same time. DevOps maturity plays a part in this. If the engineering culture is code-driven already, it lends perfectly into programmatic cloud infrastructure.

Automated build and delivery, using CI/CD tooling, is key not only to speed but also reducing risk. Automation is predictable, scalable, reproducible and auditable.

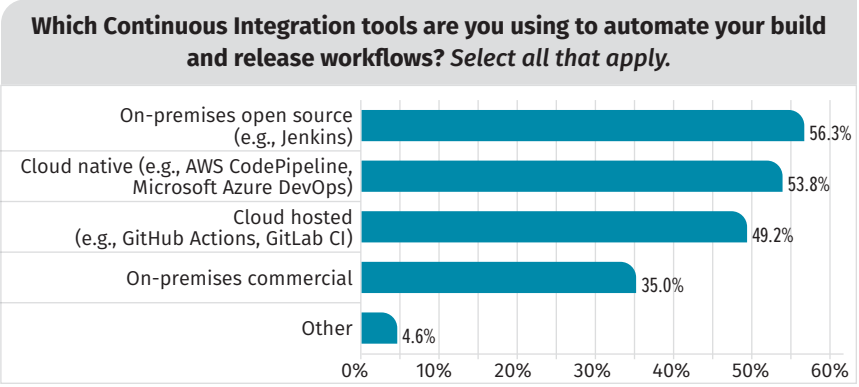


Figure 8. Continuous Integration Tools in Use

Most organizations (56%) continue to rely on open source, on-premises tools, such as Jenkins, to automatically build and deploy code changes. But more modern cloud-hosted solutions (54%) and cloud-native tools are being widely adopted, at 49% (see Figure 8). These cloud platforms enable DevOps teams to shift right—offloading the responsibilities and costs for provisioning, configuring, hardening, monitoring and managing these tool sets onto the cloud service provider—reducing operational complexity and risk. DevOps teams can focus on their builds, designing and optimizing their workflows, instead of the undifferentiated heavy lifting of infrastructure.

GitHub and GitLab are rapidly releasing security features in their automated CI/CD offerings. Examples of these features include automated static analysis, source code component analysis and dynamic application security testing (DAST). Security and development teams can easily leverage these features to shift security left and keep up with the velocity of change.

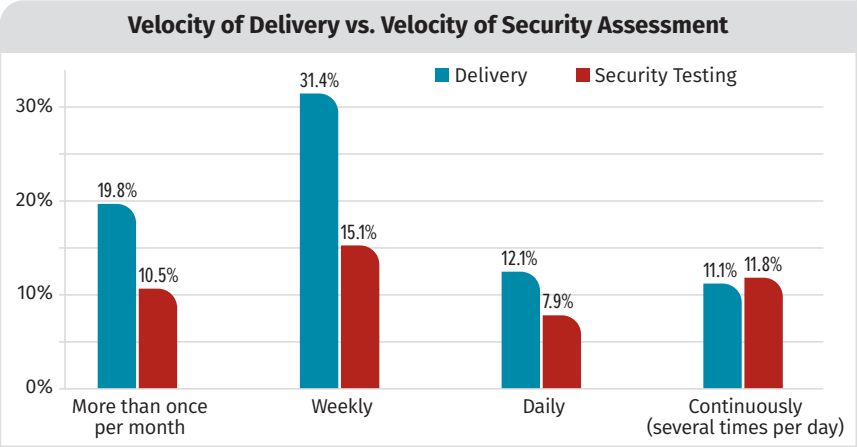


Figure 9. Velocity of Delivery vs. Velocity of Security Assessment

Is Security Keeping Up with the Velocity of Change?

The faster that engineering and development teams deliver changes, the faster security teams need to identify and assess risks. See Figure 9.

By comparing the velocity of delivery to the velocity of security testing, as shown in Figure 9, we can see that most organizations are unable to keep up with the pace of delivery:

- Although a small number of practice leaders are delivering and testing continuously, in all other cases the frequency of security testing significantly trails delivery.
- A significant number (39%) of organizations are still relying on point-in-time or ad hoc security testing, which leaves them without a clear picture of security risk or the ability to manage these risks.
- What is more concerning is that *27% of organizations do not perform security assessments at all.*

When Is Security Testing Performed in DevOps?

As organizations shift left, they need to add security testing and reviews into development and DevOps team workflows, as part of analysis, design, coding and testing. As shown in Figure 10, security testing *is being done* at multiple points in most organizations, from upfront requirements to design, through coding and developer/QA testing workflows, and into the pre-deployment and deployment stages.

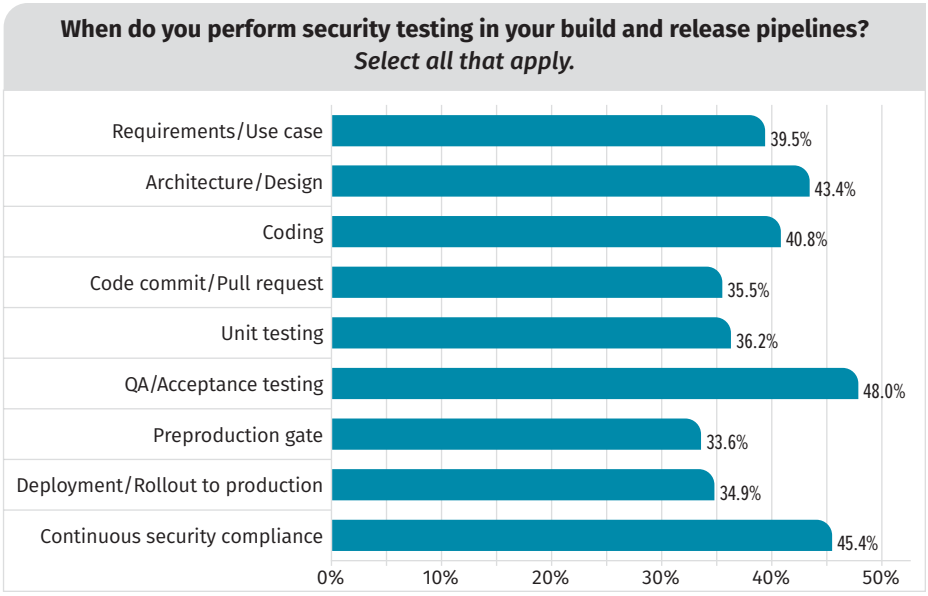


Figure 10. Timing of Security Testing in Development/Deployment

However, fewer than half of organizations (40%) have shifted security testing and reviews left into upfront requirements and design, where McGraw's Law tells us that roughly half of security problems (design flaws) are introduced.[7] This is when important—and expensive—fundamental decisions need to be made about architectural approach, platform choice, development environments, languages and frameworks. These decisions can have serious operational, security and compliance consequences.

Mistakes and misunderstandings made at these early points will need to be caught later—in QA or acceptance testing, pre-deployment reviews or after the code is already in production—where the costs and consequences of dealing with these problems are much higher.

[7] The fundamental differences between security bugs (mistakes in coding) and flaws (mistakes in requirements and design) was first examined in Dr. Gary McGraw's book *Software Security: Building Security In*, www.swsec.com

DevSecOps Tools and Practices: Shift Left or Shift Right?

Organizations can use a wide range of security tools and practices to understand and manage security risks, both early in development and later in operations.[8] We asked respondents to rank security tools and practices from most useful to least useful (with 1 being most useful) and then looked at this list—from planning through to deployment and operations—through the shift left (Dev) and shift right (Ops) lenses. (See Table 2 and our findings as described in the feature "Shift Left vs. Shift Right Analysis" on the next page).

Table 2. Usefulness of Security Testing Practices/Tools and Position in Development Cycle

	Planning	Analysis	Design	Coding	Testing	Deployment	Operations
1							Configuration security monitoring
2	Upfront risk assessments						
3			Threat modeling				
4	Security training for developers						
5							Periodic vulnerability scanning
6					Container scanning		
7							WAF
8							Third-party penetration testing
9							NDR/NTA
10				Manual code review			
11						Continuous vulnerability scanning	
12							Third-party compliance reviews
13							Internal penetration testing
14							NGAF/NGWAF
15		Security stories					
16			Dependency analysis SCA				
17							File integrity monitoring/HIDS
18					DAST		
19							Virtual patching
20					IAST		
21							Container runtime security
22							RASP
23							Bug bounties

[8] SANS has mapped security practices and open source tools in a secure DevOps toolchain, found at www.sans.org/security-resources/posters/cloud-security-devsecops-practices/200/download [Registration required.]

Shift Left vs. Shift Right Analysis

- Looking at the top 10 practices shown in Table 2 on the previous page, there is roughly an even split between shift left (Dev) and shift right (Ops).
- Configuration security monitoring (CSM) tools ranked highest on the list. According to OWASP, security misconfiguration in applications and cloud deployment is the most common security issue in modern online applications.[9] CSM tools—automatically verifying configuration settings and enforcing security hardening and compliance policies—continuously validate and audit security controls and prevent configuration drift. These tools can catch common but dangerous mistakes, such as insecure defaults, missing credentials, confidential data exposed on publicly accessible storage, inadequate encryption and logging, and so on. As cloud providers offer more services, there is more configuration work to do—making CSM even more important.
- Although most security tools and practices are targeted to operations (not development), it is clear that organizations recognize the value of shifting security left into early stages of planning, requirements and design. Upfront risk assessments and threat modeling were, respectively, the second and third most useful practices identified. However, as we saw in Figure 9 earlier, *fewer than half of organizations actually conduct security testing or reviews during requirements analysis and design*.
- Security training for developers continues to be one of the most valuable practices and is key to shifting left. Organizations cannot expect developers to take more responsibility for security if they don't have the skills to do so.
- It is becoming a mainstream practice to automate container/image scanning to catch vulnerabilities in image layers and container configuration. These scans can be added into CI/CD pipelines, as an admission control step in image registry pulls, or included in continuous vulnerability scanning.
- Network detection and response (NDR) and network traffic analysis (NTA) capabilities are important for understanding, managing and securing east–west network communications in hybrid cloud architectures and container-based microservices. They provide deep visibility into network traffic and using machine learning to identify threats and attacks as they occur. Public cloud provider traffic-mirroring services have made it easy to deploy these technologies as part of a shift right approach to operational security.
- Manual code reviews are a valuable security control for shifting left. As security teams continue to apply DevOps practices to security workflows, manual code reviews become easier to enforce with branch protections and mandatory pull requests.
- Security teams still lean heavily on point-in-time assessments (e.g., periodic vulnerability scanning, third-party penetration testing), even though these practices can't keep up with rapid, continuous change. As delivery speeds up, security will need to shift to continuous vulnerability scanning and rapid, in-phase testing approaches (e.g., IAST).
- As we saw in our previous survey from 2018,[10] while bug bounties (ranked last in the list of security practices in both surveys) receive a lot of media attention, they are difficult to set up and run in practice.

[9] https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A6-Security_Misconfiguration

[10] "2018 Secure DevOps: Fact or Fiction?" November 2018, www.sans.org/reading-room/whitepapers/cloud/paper/38690 [Registration required.]

Next, we wanted to understand how organizations are managing and conducting their security programs. Is security in organizations still siloed or are responsibilities for security shifting left, from security and compliance teams to business unit owners (who determine priorities and hold the budgets) and development or DevOps teams? See Table 3.

Table 3. Testing and Remediation Roles vs. Responsibility

Role	Managing	Conducting	Accepting	Corrective Actions
Business unit owner	39.3%	12.6%	52.6%	26.8%
Commercial application vendors	17.4%	23.4%	19.4%	33.3%
Cross-functional teams in DevOps or DevSecOps	28.8%	38.1%	29.4%	39.2%
Development team	28.8%	38.1%	36.8%	63.3%
External security consultants	8.5%	42.8%	13.2%	9.2%
Internal security team	50.1%	65.4%	34.1%	22.6%
Quality assurance	17.9%	33.4%	28.8%	18.1%
Security architect	42.1%	38.8%	28.8%	24.1%
Security-as-a-service (cloud) providers	13.4%	29.5%	17.3%	22.0%
System architect	33.9%	35.9%	31.5%	36.0%
Other	4.6%	2.2%	2.6%	2.7%

Takeaway

Security teams (internal and/or external consultants or service providers) still primarily own responsibilities for conducting security testing in most organizations. To scale, organizations need to shift more responsibilities for security testing onto development or DevOps teams. Although there are some technology problems to solve here, such as integrating testing tools and practices into automated workflows (as we will see later in this analysis), the major barriers to shifting security testing to developers are not technical; they are organizational.

Now, let's shift right and look at how quickly and effectively organizations deal with security issues in operations.

How Are Security Vulnerabilities Being Managed in Operations?

Figure 11 indicates that less than half of organizations are repairing all, or almost all, critical vulnerabilities satisfactorily and in a timely manner. This has serious implications for the security of production systems, especially those that are mission critical. When serious vulnerabilities are found in production, teams must be able to respond and patch quickly in order to close the *window of exposure*[11] and outrace attackers, and stay in compliance.

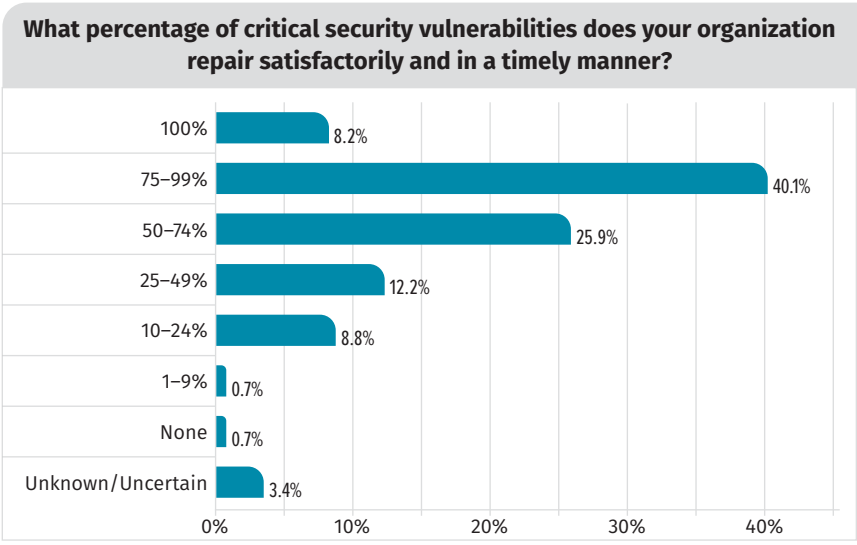


Figure 11. Percentage of Security Vulnerabilities Repaired in a Timely Manner

[11] Security researcher and thought leader Bruce Schneier coined this term: www.schneier.com/crypto-gram/archives/2000/0915.html

As Figure 12 shows, more than two-thirds of organizations are patching critical vulnerabilities within 30 days (a common cutoff for regulatory compliance). As organizations continue to adopt DevOps practices, we expect this to improve as organizations leverage investments made in CI/CD tooling, automated testing and automated deployment to reduce risks and costs of patching.

The time needed to fix critical security vulnerabilities is a key metric for determining the success of a security program, and the organization's ability to successfully shift right, by:

- Leveraging the cloud platform's managed infrastructure controls to clearly identify what needs to be patched
- Leveraging DevOps technical practices and automated build and deployment (CI/CD) toolchains and technologies, such as continuous delivery and containers, to minimize the work and risks involved in building, testing and releasing patches
- Relying on cloud platform providers to automatically patch managed infrastructure services and underlying hosts

As shown in Figure 13, the "Number of security issues discovered after deployment" (measuring the vulnerability escape rate to production) and "Post-audit remediation steps required" are important in evaluating the *effectiveness* of the organization's shift left security controls. They are also important for assessing changes or improvements made to the Dev side of the security program. Organizations can use this to identify weaknesses and gaps in training, reviews or testing, and build a case for upfront investments in secure development practices.

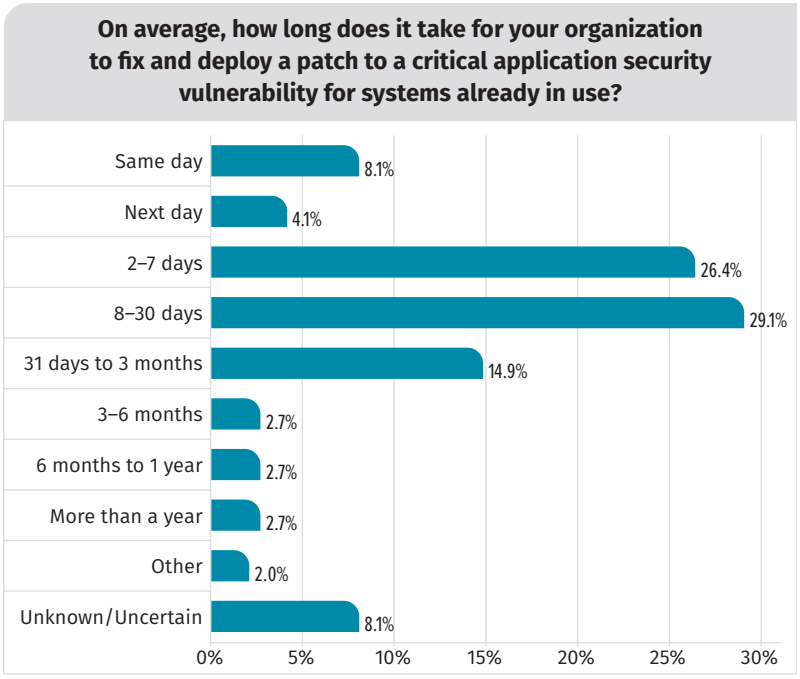


Figure 12. Time to Fix and Patch a Critical Vulnerability

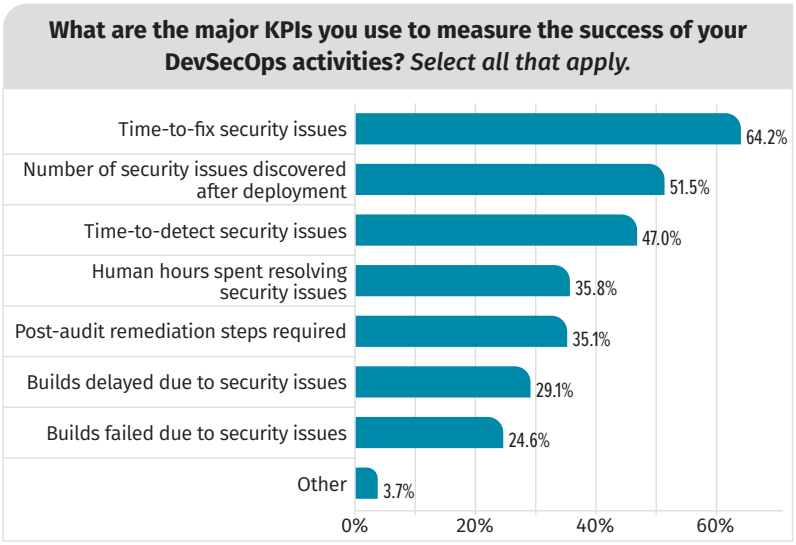


Figure 13. KPIs in Use to Measure DevSecOps

The other measurements, including failed or delayed builds, and the time spent to detect or resolve security issues, relate to the *efficiency* of security controls. They help in identifying opportunities for automation and other optimizations in security workflows.

Takeaway

Shift left (i.e., implementing security reviews and testing as part of development) is not keeping up with the velocity of delivery. This increases the risks of critical security vulnerabilities making it to production.

To compensate, security has to shift right (i.e., be prepared to identify and respond to problems immediately) through:

- Monitoring telemetry and feedback loops to detect attacks and exploits
- Operational incident response capabilities to escalate problems and risks
- CI/CD build pipelines and programmable infrastructure to get patches out quickly
- Root cause analysis to improve systems and organizational capabilities
- Compliance and monitoring solutions that not only evaluate cloud-based desired state configuration, but also provide event triggers for automating notifications, creating tickets and even remediating noncompliant resources

What It Takes to Successfully Shift Left and Shift Right

Both technical and organizational factors are important to the success of DevSecOps programs.

The major challenges to implementing DevSecOps in organizations are fundamentally organizational, from insufficient budget and shortage of security skills and security training, to organizational silos, poor prioritization, and lack of buy-in by management and by development teams. See Figure 14.

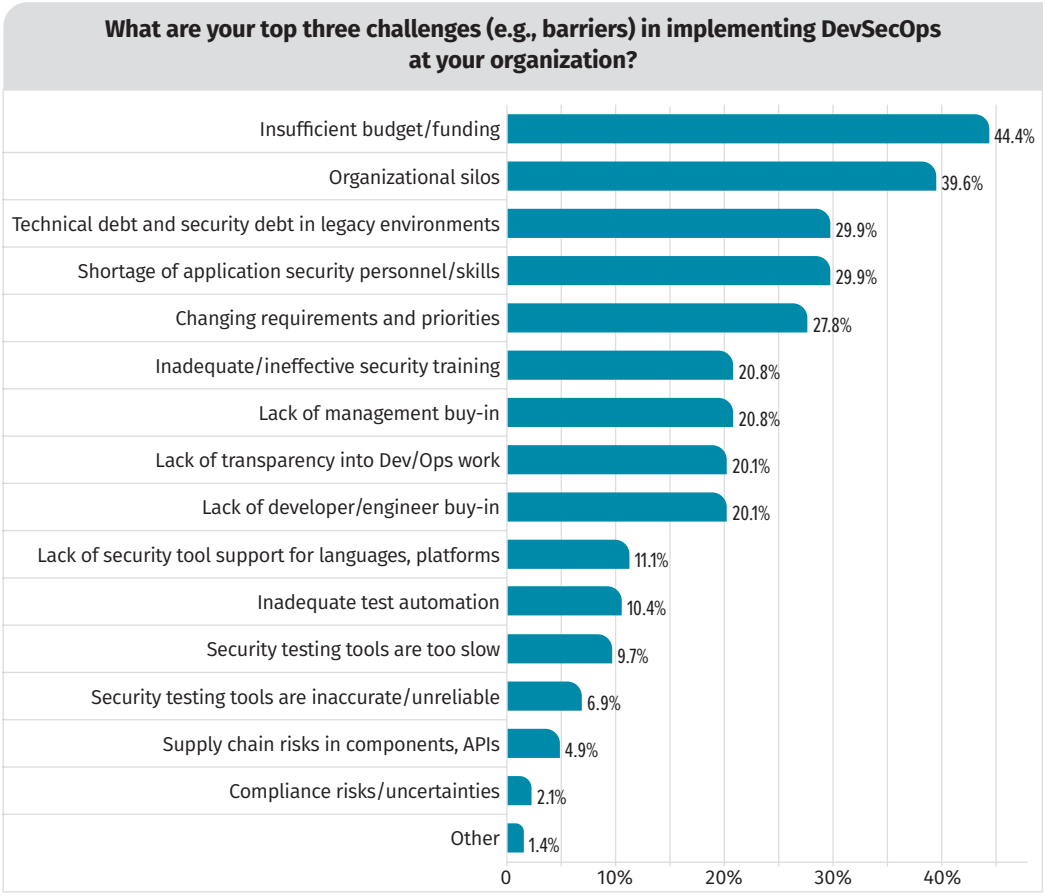


Figure 14. Top Challenges in Implementing DevSecOps

As we explored in the 2018 DevOps survey, lack of tools or poor technical practices are not holding up the success of DevSecOps programs as much as management factors. Organizations can overcome internal management barriers, cultural inertia and legacy debt by shifting more operational responsibility into the cloud, and in the process, become more agile and more secure.

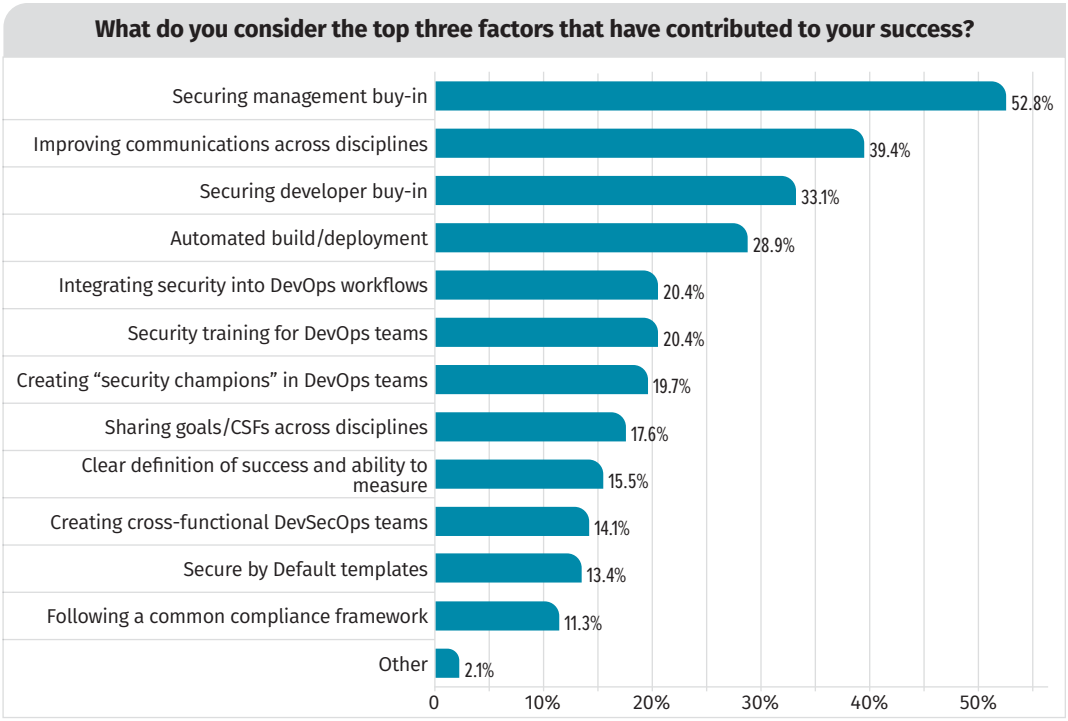


Figure 15. DevSecOps Success Factors

Management factors are not only barriers, but also key enablers of success. Securing buy-in from managers and developers, and improving communications across disciplines (e.g., development, operations and security) are the most important factors for the success of security programs. See Figure 15.

Automating build and deployment steps, and integrating security testing into DevOps workflows, are also prerequisites to the success of security programs. Security and compliance teams need technical fundamentals, such as build and test automation, CI/CD pipelines, and programmatic infrastructure, in place from DevOps teams to successfully shift security left.

Conclusions/Moving Forward

Most organizations need to make fundamental changes to make secure DevOps a reality. To make the changes required, there are two approaches that DevSecOps teams can follow:

- **Shift left—**Initiate change from the bottom up, building on technical disciplines and practices and especially on automation. Foundational technical practices—such as automated build and delivery pipelines (CI/CD), test automation (test-driven development [TDD], behavior-driven development [BDD]), pair programming and pull request reviews, and configuring infrastructure in code—all enable DevOps teams to move faster and deliver working software. These practices also provide a control plane where teams can insert security checks and tests. And they support and reinforce consistency, transparency and accountability, helping to encourage and sustain longer-term, deeper changes in the ways that people think and work across the organization.

- **Shift right—**Offload operational, security and compliance risks and obligations onto the cloud provider. This approach takes advantage of the cloud provider's scale, resources and agility to solve problems and compensate for the organization's weaknesses. This frees up scarce security and development resources to focus on important priorities and risks.

You need to shift right, offloading operational security and compliance responsibilities to the cloud platform, in order to successfully shift left. Shifting right reduces complexity and cost, and shrinks the security problems and risks that you need to manage, so that you can focus on making your security training, testing and reviews more targeted and realistic, which will increase your chances of success.

Shift Right in Order to Shift Left

Shift left is not about throwing security and compliance problems over to developers. Shift left involves:

- Enabling development teams, giving developers the training, tools and time to do a good job of designing, building and delivering high-quality, reliable and secure services
- Leveraging DevOps investments in automation and tooling to build a control plane for security and compliance
- Helping DevOps teams understand security risks and compliance requirements and how to manage them in the most effective and efficient ways possible.

Shift right is not about leaving security risks until it's too late. Shift right involves:

- Taking advantage of cloud platforms, getting maximum leverage from the strengths and capabilities that the platforms offer for resilience, scale, security and compliance.
- Shifting responsibility to managed cloud platforms, rather than re-inventing the wheel, to help security and compliance move at the speed of DevOps
- Collecting attack data, threat intelligence to identify real risks and attack vectors that need to be defended—learning where to focus training, testing and reviews
- Using data about vulnerabilities escaping to production to understand weaknesses and gaps in understanding, in testing and controls, and shifting left again to improve your processes' design and tooling

KPIs such as *time to fix security problems* and *escape rate to production* help you determine whether to shift left and try to prevent/catch problems up front through better training and better tooling in the development workflow, or to shift right and add protection at deployment and runtime.

About the Authoring Team

Jim Bird, SANS analyst and co-author of SEC540: Cloud Security & DevOps Automation, is an active contributor to the Open Web Application Security Project (OWASP), and an author of books on Agile Security and Secure DevOps. He has worked at major technology organizations and financial institutions around the world in management, software development, operations, and IT and application security.

Eric Johnson is a Principal Security Engineer at Puma Security and SANS Senior Instructor focusing on cloud security, DevSecOps automation, and building static analysis tools. His experience includes application security automation, cloud security reviews, static source code analysis, web and mobile application penetration testing, secure development lifecycle consulting, and secure code review assessments.

Frank Kim leads the management and software security curricula for SANS, developing courses on strategic planning, leadership and application security. He is also a SANS Faculty Fellow, helping to shape, develop and support the next generation of security leaders. Previously, Frank served as CISO at the SANS Institute, leading its information risk function, and executive director of cybersecurity at Kaiser Permanente, where he built an innovative security program to serve one of the nation's largest not-for-profit health plans and integrated healthcare provider. Currently, as founder of ThinkSec, a security consulting and CISO advisory firm, Frank helps leaders develop business-driven security programs.

Sponsor

SANS would like to thank this paper's sponsor:

orca security