PRISMA® CLOUD | paloalto® NETWORKS

Implementing Cloud Native Security: Shift-Left to Increase Effectiveness

The past few years have witnessed the rise of new cybersecurity trends in cloud native development, including "shift-left security" and "DevSecOps." These terms, which build on concepts rooted in the broader DevOps movement, refer to new security strategies and paradigms that help organizations keep their workloads secure in the present age of cloud-based, scale-out, constantly changing applications and infrastructure.

Many in IT, security, and development probably understand what these concepts mean at a high level. Implementing them in a practical sense, however, may be harder. Not only do you need to gain organization-wide buy-in for the value that modern security strategies provide, but you must also determine the most efficient and effective ways of integrating them into workflows and tool sets that are already in place.

This guide is designed to address that challenge by exploring what modern, shift-left security entails. We start by defining modern security paradigms and explaining their importance. We then walk through practical strategies for putting those security philosophies into practice, with guidance broken down according to the different IT architectural strategies that organizations commonly have in place today.

We can't offer step-by-step instructions for implementing concepts like DevSecOps and shift-left security at your organization because every team and tool set is unique. We can, however, provide the guidance you need to plan an effective security strategy that is tailored to meet the needs of the cloud native, DevOps-dominated era.

DevOps and the Redefinition of Cybersecurity Needs

The first step in planning a strategy for implementing modern security practices at your organization is to understand exactly how and why the security landscape has changed over the past decade.

What Is DevOps?

At the root of this change is DevOps, an approach to software development and deployment that emphasizes automation, agility, and the continuous delivery of application updates. DevOps is driven by a process known as continuous integration/continuous delivery, or CI/CD. CI/CD is a strategy for rapidly delivering software using a variety of automation tools—including CI servers, test automation software, and release automation suites—that allow teams to write, test, and deploy new code on a regular, consistent basis.

Over the past decade, an increasing number of organizations have turned to DevOps as a way to improve their ability to modify software quickly and ensure that developers and IT operations departments collaborate more closely. DevOps also empowers businesses to pivot more quickly by making it easier for them to release new software, and it can increase the overall happiness of IT teams. By the closing years of the 2010s, a strong majority of organizations reported that they were already embracing DevOps or had plans to do so.

What Does DevOps Mean for Security?

DevOps itself focuses on software development and delivery techniques. It does not explicitly address security needs, and it might be tempting to assume that DevOps has few implications for security. In fact, however, DevOps has tremendous consequences for the way applications and infrastructure must be secured. The following are the main reasons for that:

- **Speed**: By increasing the rate at which software is written and deployed, DevOps requires security teams to move more quickly than ever in testing for vulnerabilities pre-deployment and monitoring for breaches post-deployment.
- **Collaboration**: DevOps places a heavy emphasis on collaboration between different types of teams. In order to meet the needs of DevOps, then, security teams must find ways of communicating and collaborating seamlessly with developers and IT engineers—groups with which they traditionally did not engage on a consistent basis. Security teams must also ensure that their priorities and challenges can be understood by other teams that do not specialize in security.
- **Tooling**: Most CI/CD tools are designed to help teams get software out the door as quickly as possible. Security is not their focus. This means that organizations must work to integrate additional, purpose-built security tooling into their workflows to meet security needs in a fast-moving DevOps environment.
- **Cloud native architectures**: In order to meet the demands of DevOps, many organizations have turned to new types of application technologies, such as containers and serverless functions. These technologies pose new security challenges, which we discuss in detail later in this white paper.
- **Environment scale**: The automation and rapid pace of change that DevOps enables has encouraged some organizations to build application environments that are larger than ever. They may sprawl across multiple clouds and involve a range of different deployment technologies and management tools. All of these moving pieces increase the breadth of attack surfaces and add to the complexity that security teams must manage.

In short, the nature and scope of the security challenges that organizations must manage in the age of DevOps have evolved significantly.

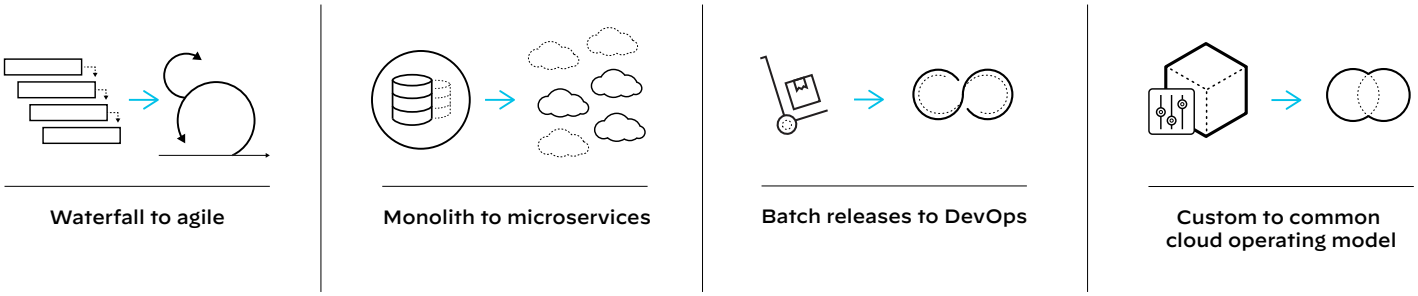

Figure 1: Cloud is modernizing the software development lifecycle

Addressing DevOps Security Needs: Shift-Left Security

Two main strategies have emerged to address these new security challenges: shift-left security and DevSecOps. Shift-left security refers to methodologies to include security earlier in the development process, and DevSecOps refers to organizational or cultural changes that support secure development.

Shift-Left Security

Shift-left security emphasizes the integration of security testing and vulnerability scanning into the early stages of the software development lifecycle—in other words, to the "left" side of the process, if you imagine a linear workflow that starts with coding and ends with deployment and management.

Whereas security strategies in the past focused largely on detecting threats in software once it had been deployed, shift-left security prioritizes the identification of vulnerabilities as part of the software design and development phase of the CI/CD pipeline. It helps teams identify security problems before code is pushed into production so that they can be addressed before they impact end users. This doesn't mean that security monitoring post-deployment is not important —it is—but rather that the security process expands in scope to devote more resources to pre-deployment scanning than it did traditionally.

Shift-left security builds on related DevOps concepts, such as shift-left testing, which moves software testing of all types to the left in the CI/CD pipeline.

DevSecOps

The second key concept for meeting DevOps security needs is DevSecOps. Put simply, a DevSecOps mindset means that developers, security engineers, and IT operations engineers communicate and collaborate closely. Each group must understand the others' needs and have clear visibility into how the initiatives launched by one set of stakeholders will impact the priorities of others.

For example, the security team should understand how a new feature that developers are implementing, or a new deployment tool that the IT team is rolling out, will impact security. Likewise, if the security team plans on migrating to a new security platform, developers and IT engineers should be plugged into the process so that they can adjust their workflows as necessary.

In essence, DevSecOps extends the core goal of DevOps—close collaboration between developers and IT engineers—to include security engineers as well.

Why Shift-Left Security?

Adopting shift-left security is important not because developers and IT teams are blind to security needs, but because they often lack the time and deep experience necessary to address modern security problems.

Developers care about writing good code, and good code means secure code. For their part, IT engineers care about following security best practices when deploying and managing software. Yet these groups may not always know what the latest best practices are when it comes to security, especially in a DevOps environment where tools and processes are changing constantly.

By embracing shift-left security as a way of plugging the security team into development and IT processes—and to check for security risks early and often—organizations place themselves in a stronger position to meet the new security challenges that DevOps has introduced.

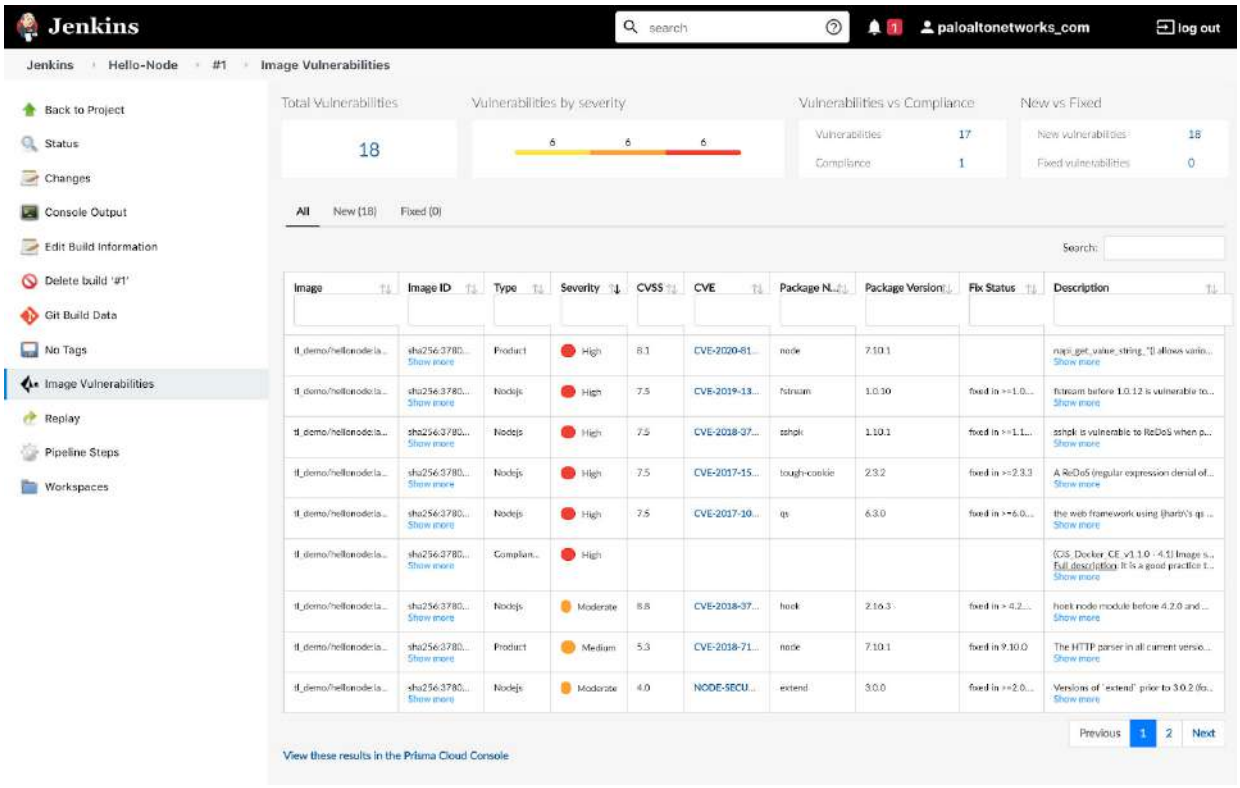

Figure 2: Vulnerability scanning integrated with Jenkins, a CI server

Addressing Technical Needs

Now, let's discuss how to implement these methodologies in a practical sense, starting with technical requirements. When selecting tools that enable shift-left security, there are three main traits to look for: DevOps tool integration, multiple architecture and configuration support, and security automation.

Integration with DevOps Tools

Because DevOps prioritizes integrations between teams and workflows, it's critical to select security tools that are designed to integrate with other DevOps tools. For example, vulnerability scanners should be able to integrate with CI servers so that they can scan code directly and continuously, as developers integrate it. Likewise, tools designed to detect security risks in infrastructure configurations should integrate with infrastructure as code (IaC) tools, which DevOps teams commonly use to automate infrastructure setup.

The point here is not just about basic integration, such as the ability for two tools to share data with each other. The ultimate goal in selecting security tools should be to choose solutions that empower all stakeholders—developers, IT engineers, and security engineers—to gain visibility into and share ownership of security responsibilities. Security tools that are so complex that only security specialists can understand them do not meet DevOps' security needs.

Support for Multiple Architectures and Configurations

As noted in the intro, the infrastructure and application deployment technologies associated with DevOps tend to be highly complex. They are also very diverse; the architecture used by one organization may look nothing like that of another. You can use virtual machines running on-premises to support DevOps workflows, or you can use a hybrid cloud framework where applications are containerized and distributed across cloud and on-premises servers.

To do their job effectively, then, security tools must be able to work with any and all configurations and architectures. Avoid tools that only work with the public cloud, for example, or that only support containers without supporting bare-metal servers and virtual machines.

Security Automation

Automation, which is essential for enabling the fast release cycles of CI/CD pipelines, is part and parcel of DevOps. It should also be central to DevOps security tools.

Look for security tools that can automatically configure themselves to work with your environment, and that come pre-populated with the security policies they need to understand common threats. Likewise, strive to adopt tools that automate as many security workflows as possible. Tools that can automatically scan your infrastructure and application configurations for potential vulnerabilities, and even automatically remediate security risks where feasible, will allow your security tools to integrate seamlessly into broader DevOps workflows.

Meeting Organizational Needs

Tools are only one facet of an effective DevOps security strategy. It is equally important to have the right organizational structure and processes in place to achieve the collaboration between different stakeholders that is at the core of DevSecOps and shift-left security.

Although every organization is unique, the following strategies can help to build the organizational structures and workflows that effectively address DevOps security needs.

Consistent Communication Across Teams

First and foremost, strive to achieve open, consistent, and continuous communication between different teams—especially the development, IT operations, and security teams.

At a basic level, consistent communication means having shared channels that these teams can use to collaborate. Tools like Slack go a long way toward meeting this need. Communication is about more than chat groups though. It also requires shared visibility into security-related workflows and data. Instead of allowing only the security team to access logs or tickets related to security issues, empower all stakeholders to access that information where relevant.

Likewise, the security team should have visibility into the systems and data that developers and IT engineers rely on to do their work. Even if that data does not always seem relevant to security, allowing visibility into it can help security teams recognize vulnerabilities that may not be obvious to other team members.

New Team Structures

To achieve even tighter collaboration between the security team and other teams, consider embedding security engineers in development and IT teams. Alternatively, use a matrixed team structure wherein security specialists report to multiple team leaders, allowing them to share their expertise across the IT organization. If resources allow, you may even consider building a DevSecOps team composed of engineers with equal measures of experience in security, IT engineering, and development.

These changes to team structures are not strictly necessary. In smaller organizations, in particular, it does not always make sense to restructure teams because they are already small enough that they can collaborate effectively around security without having security specialists embedded in them. In larger organizations, however, rethinking the way personnel are structured and managed can be critical for reinforcing DevOps security goals.

Invest in Training

No matter how you choose to structure your teams, make sure that different types of engineers understand how to work with the tools used by the other types. In other words, train security engineers in DevOps and IT tools, and train IT engineers and developers to use security tools.

This is not to say that all team members need to become experts in using all tools—security engineers don't need to suddenly become expert coders, and vice versa. That would be unrealistic. Instead, the goal is to ensure that all stakeholders have the basic competence required to understand how different tools fit into security workflows and how to access the data associated with those tools when troubleshooting security issues.

Shared Metrics

Establish shared metrics to measure the effectiveness of all team members in achieving security goals. Examples of such metrics include:

- **Vulnerability discovery rates in new code written by developers**. This metric provides a sense of how

well developers are doing at writing secure code and how effectively security engineers are guiding them in this process.

- **The rate of security issues discovered in production**, which tracks vulnerabilities that security engineers and IT engineers fail to detect before deployment.
- **The rate of deployments that are canceled due to failed security tests**. No one on your team wants these failures to occur because they delay the CI/CD process (which reflects poorly on developers and IT engineers) while also undercutting the security team's core mission of mitigating total vulnerabilities.

Process Changes for Shift-Left Security

Now that we've discussed how to manage tools and people to meet DevOps security needs, let's address the final key piece of the puzzle: processes.

Because the CI/CD process is at the heart of DevOps, integrating security processes into CI/CD should be the core goal when implementing shift-left security. That said, CI/CD processes can take different forms and be powered by different technologies. The approach that organizations take to integrating security into them should reflect the makeup of the CI/CD process.

The following are common CI/CD setups with tips on integrating security into each.

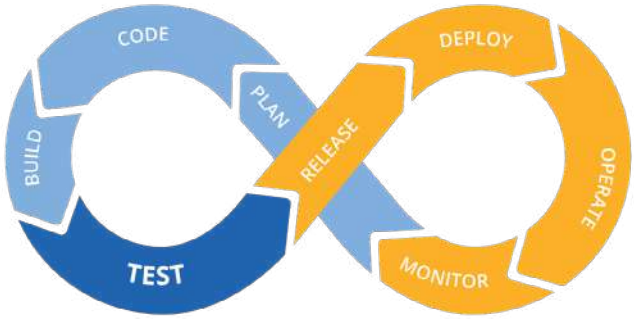

Figure 3: The CI/CD process

For Workloads on PaaS

When your applications run on platform as a service (PaaS), which combines software development tools with a deployment solution, you have relatively little control over the host environment and infrastructure. That is typically managed by the PaaS provider. However, there are still multiple junctures at which you can address security needs:

- **Runtime protection**: Be sure to scan for vulnerabilities that may exist in your production environment. Equally important is setting up active defenses, like firewalls and workload isolation, that can mitigate the risk and impact of a breach.
- **Vulnerability scanning**: Code that you write should be scanned for vulnerabilities before it is deployed. If your PaaS doesn't provide native scanning features or integrations, scan the code outside the PaaS using an external tool before deploying it.
- **Configuration auditing**: Audit your PaaS configurations with security tools that can detect potential risks caused by poorly managed configurations.
- **Network protection**: Scan for network-borne threats on both internal networks that your applications use to share data with each other and public-facing networks that connect your PaaS environment to external users.

For Workloads on Containers and Kubernetes

Applications that are deployed using containers simplify security operations in some respects, but they also create novel security risks that arise from the lack of strict isolation between containerized workloads. Address those security issues through the following strategies as part of the CI/CD process:

- **Security testing across the application lifecycle**: Containers present an opportunity to security teams because containers keep most environment variables consistent pre- and post-deployment. In that way, they enable more reliable security testing by allowing teams to test applications prior to deployment within the same environment that will host them post-deployment. Take advantage of this opportunity by testing rigorously.
- **Integration of scanning into container registries**: Container registries, which host container images before they are deployed, allow images to be scanned automatically and continuously.
- **Container runtime protection**: A containerized application can be compromised by vulnerabilities, not just in the application code itself or the host operating system but also in the container runtime (which is the system process that runs the container). Be sure to check your runtime for known vulnerabilities and update it accordingly.

- **Container-to-container microsegmentation**: By default, containers share resources with each other and the host operating system, which presents security risks. These can be mitigated by using resource quotas and (on Kubernetes) pod security policies to restrict the extent to which containers can interact with each other.
- **Underlying operating systems**: Containers and Kubernetes® run on top of a host server (or, more likely, multiple host servers). Security teams must secure the host with both runtime protection and continuous vulnerability and compliance management within the host operating system.

For Workloads on Virtual Machines and Bare Metal

CI/CD pipelines that run on more conventional infrastructure, including virtual machines or bare-metal servers, can integrate security through the following approaches:

- **Automated auditing and monitoring of configurations**: Infrastructure configurations should be automatically scanned and audited on a continuous basis to detect potential security risks.
- **Identity access management tools**: Identity and access management (IAM) tools provided by IaaS platforms should be deployed to restrict access. In addition, IAM configurations must be scanned to detect potential vulnerabilities.
- **Microsegmentation**: Although virtual machines and bare-metal servers are isolated from each other in most respects by default, they can still share networking, storage, and other resources. Segment high-priority workloads from shared resource pools in order to mitigate the risk of threats to those systems.
- **Vulnerability management**: Security teams must take steps to mitigate vulnerabilities that may exist on the host server, such as a lack of isolation between production applications and ancillary operating system components.
- **Compliance**: By auditing the configurations of virtual machines and bare metal, teams can identify setups that deviate from best practices and governance requirements.
- **Runtime protection**: The runtime layer that executes applications—whether it is a container runtime or a language interpreter for non-containerized applications—must be scanned for vulnerabilities and insecure configurations, in addition to being protected against potential breaches.

Many organizations, of course, use multiple approaches to building CI/CD pipelines. They might deploy some applications using containers, for example, while using virtual machines for others. A multifaceted approach to CI/CD is fine, provided that teams implement the security processes that are necessary to protect each workload deployment method they rely on.

Conclusion

Even though DevOps itself is not concerned primarily with security, DevOps has fundamentally altered the security landscape for many organizations by leading them to adopt new types of tooling, increasing the complexity of their infrastructures and applications as well as placing tremendous pressure on them to move quickly. All of these changes make it more difficult or even impossible to secure workloads by relying on traditional security tools and processes.

Instead, organizations that embrace DevOps must also embrace the related concepts of DevSecOps and shift-left security. By implementing tools, organizational structures, and processes designed to support these security methodologies, IT teams can effectively meet the novel security challenges that have arisen in the age of DevOps, while positioning themselves to handle whichever security needs may arise in the future.

No matter which cloud architecture or workloads you deploy, Prisma Cloud from Palo Alto Networks can provide deep visibility across complex stacks. Prisma® Cloud is the industry's most comprehensive Cloud Native Security Platform (CNSP), addressing security needs across all stages of the cloud software development lifecycle, no matter which cloud services you use or how they are managed.

With Prisma Cloud, teams can automatically detect vulnerabilities that are difficult for even the best-trained security engineer to notice manually. It can also enforce best practices to help ensure that all members of your team operate in a secure fashion when deploying workloads to the cloud.

To learn more, request a personalized demo or view a recorded demo at your convenience.

paloalto® NETWORKS | Cybersecurity Partner of Choice

3000 Tannery Way
Santa Clara, CA 95054
www.paloaltonetworks.com

Main: +1.408.753.4000
Sales: +1.866.320.4788
Support: +1.866.898.9087

prisma-implementing-cloud-native-security-wp-080221