The Complete Playbook to Operationalizing IaC Security

Learn How to Scale Your IaC Security Program and Build a Secure-by-Default Culture

PRISMA® CLOUD | paloalto® NETWORKS

Table of Contents

Introduction: Building a Secure-by-Default Culture

Infrastructure as code (IaC) has transformed how infrastructure is provisioned and configured, enabling organizations to adopt agile development across their entire stack. With IaC, you can easily manage, configure, update, secure, and scale your infrastructure via imperative or declarative configuration files. When leveraged at scale, IaC addresses notorious cloud challenges, such as scalability, availability, configuration consistency, and right-sized infrastructure architecture for your applications. And while an IaC-based approach to cloud-native development brings significant benefits, it can introduce some unique security challenges.

To make your IaC adoption as seamless as possible, it's important to build a holistic and proactive strategy—one that includes security—to get ahead of those challenges. By integrating security best practices into your IaC strategy from day one, you'll be able to proactively mitigate some of the most common issues organizations face.

In this playbook, we'll review some common security challenges organizations encounter when leveraging IaC, highlight the foundations for a prevention-first cloud security strategy, and give you a step-by-step guide for operationalizing IaC security at scale based on your unique needs.

Common Challenges

IaC provides a wide range of benefits and opportunities for development and DevOps teams, but ignoring security when building an IaC strategy can lead to misaligned security feedback and gaps down the line. If you have to shore up your cloud infrastructure security after you've already embedded IaC into your workflows, there are a few common challenges you'll face.

The Misconfiguration Snowball Effect

Because IaC allows you to quickly deploy multiple environments at scale, it provides both significant efficiency gains and the possibility that a single misconfiguration in an IaC template gets deployed across hundreds of resources. The effect on your team of a single IaC misconfiguration cascading into hundreds of alerts—also known as the IaC misconfiguration snowball effect—is less time to spend on high-impact work, alert fatigue, and risk going unaddressed.

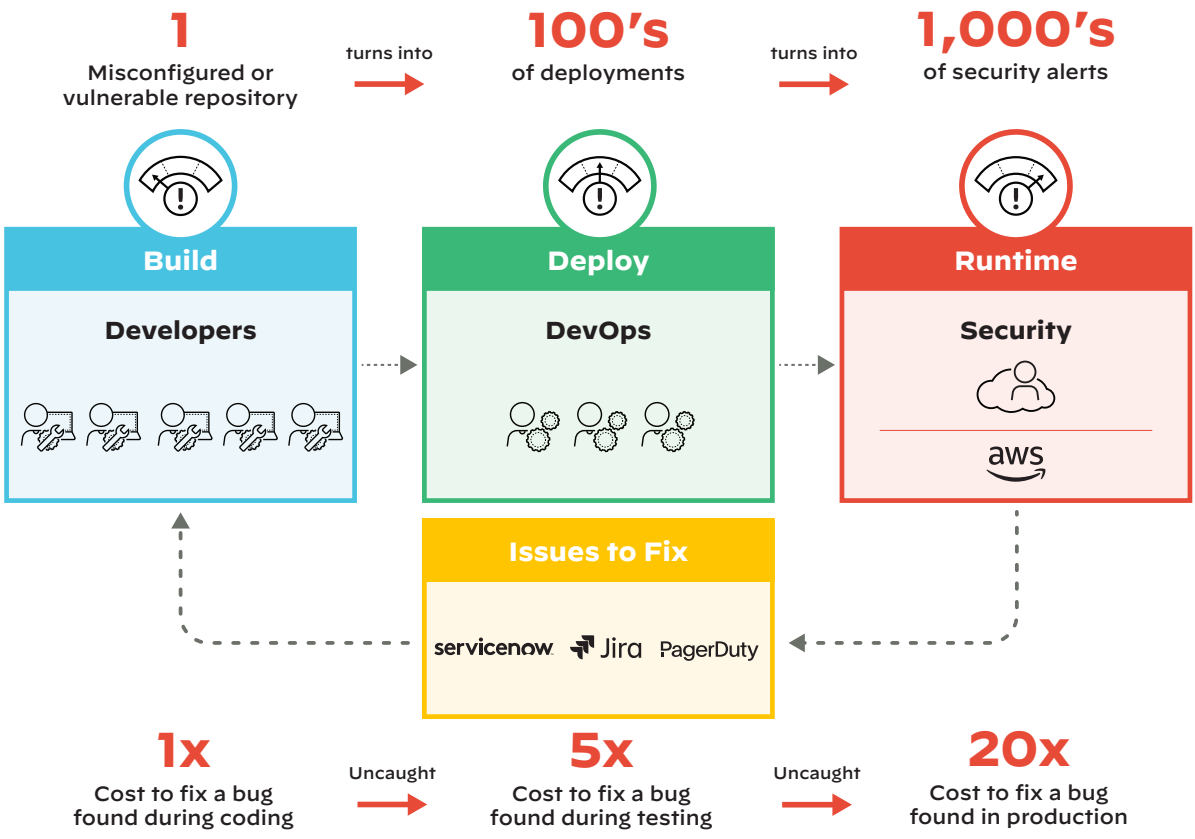

Figure 1: The next big challenge: "shift-left" cloud security

Misconfigured Open Source IaC Components

Many teams don't account for the extent of open source code components used by their IaC. While open source code simplifies and expedites the IaC adoption process, these code components are built to simply complete a task and oftentimes don't consider important security configurations. Research by Unit 42 indicates that about half of open source Terraform templates contain misconfigurations,[1] so you can't assume that open source IaC is secure by default.

Because of the complexity of modern, cloud-native environments, it's nearly impossible to manually keep misconfigurations in open source IaC from being deployed. Additionally, developers typically don't have the security training they need to properly understand and remediate issues. So, without a security-first approach to IaC, it's all too easy for misconfigurations in open source to slip through the cracks, get deployed, and become alerts your security team needs to triage or worse, weaknesses bad actors can exploit.

Cloud Infrastructure Drift

When your team adopts IaC but ignores the GitOps best practice for treating your configuration files as the single source of truth, you run the risk of introducing cloud infrastructure drift. Cloud drift occurs when the configuration of infrastructure in your runtime environment doesn't match the configuration of infrastructure in your build-time environment. Drift most often happens when a change is made to a running cloud resource out-of-band from the IaC file that was originally used to provision it. Your team may make out-of-band changes to quickly fix a misconfiguration or deal with a critical issue, but these changes can also happen because of miscommunication or a lack of understanding of IaC.

Regardless of why drift happens, it reduces your team's visibility into your resources, makes it more challenging to identify misconfigurations, and negates many of the benefits of IaC. For example, if out-of-band changes are made to an IaC template, the next time that template is deployed, those changes will be overwritten.

Foundations for a Secure-by-Default Approach

With the right approach to IaC security, you can get ahead of these challenges. On top of IaC basics, such as keeping all configuration files under source control and ensuring your Git repository is the single source of truth, building in security from the start is key to creating a secure-by-default environment.

Because IaC is code, it can be versioned, stored, tested, and secured just like any other code. In that way, IaC is key to embedding cloud security earlier in the development cycle. IaC security solutions, such as Prisma Cloud's Code Security Module, provide developer tool plugins and integrations to make it easy to surface IaC security issues as part of existing workflows across the development lifecycle: from code and commit to build and deploy. This is especially important for IaC because while addressing issues as early as possible is cheaper and faster, you gain more representation of resources the closer they are to their runtime state.

Figure 2: The application development lifecycle where multiple checkpoints can be added across the development lifecycle

1. *Cloud Threat Report, 2H 2021*, Unit 42, September 28, 2021, https://www.paloaltonetworks.com/prisma/unit42-cloud-threat-research-2h21.html.

Leveraging IaC security solutions to address issues earlier will reduce the number of runtime errors and alerts, thus minimizing time spent on repetitive security tasks. But getting to that ideal state takes time and resources, and the path will differ depending on your team's priorities and workflows. This playbook will walk you through rolling out and operationalizing your IaC security program.

A Practical Guide to Operationalizing IaC Security

You have so many ways you can roll out an IaC security program to best suit your needs, with myriad decisions to make along the way. We know how difficult it can be to navigate that process, since your rollout strategy depends on many factors, such as how mature your IaC program is and what problems you're looking to address with IaC.

Because those factors are unique to each organization, we'll give you a general step-by-step plan that you can customize for your needs as you choose your IaC security path, roll out your program, and iterate along the way.

Before You Start: Pick Your Path and Define Your Goals

Before you can roll out your security program, you need to pick the best path based on your team's existing processes and how much effort you're ready to invest in setting up your program.

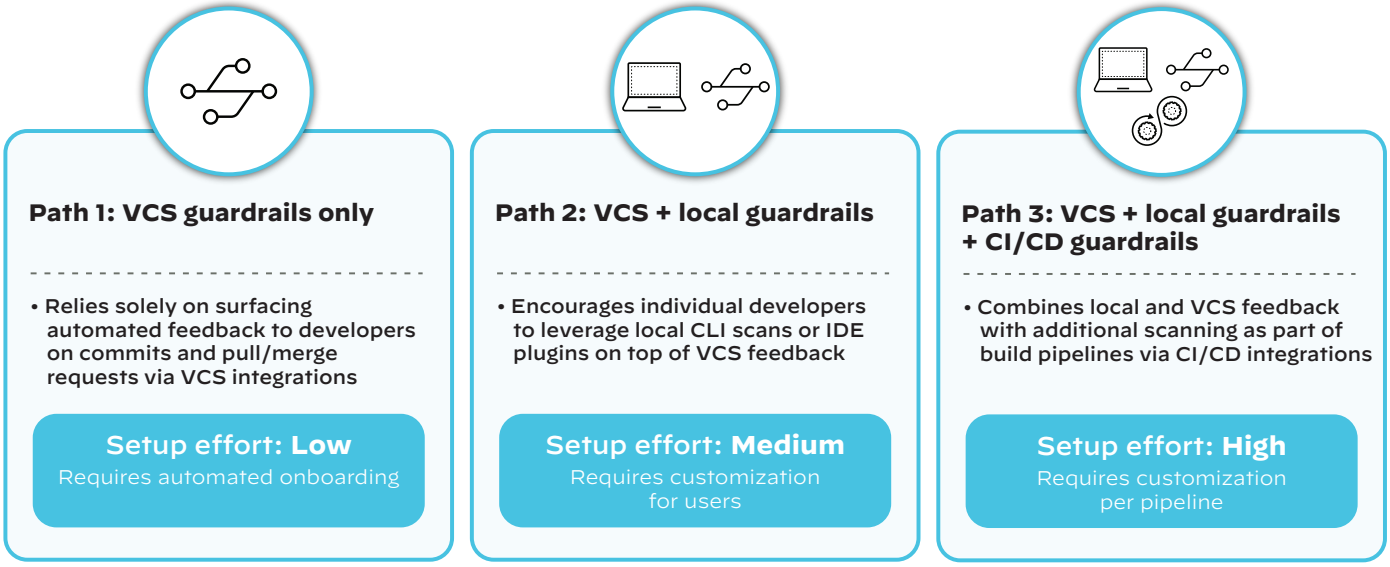

Figure 3: Three paths to roll out your security program

As you think about your direction, we recommend starting with your ideal end state and then iterating on your approach and expanding until you reach your goal. Keep in mind that choosing a path now does not lock your team into that path.

Path 1

By taking this approach, you get the majority of IaC scanning benefits for the least setup effort, like getting easily actionable, collaborative, and automated feedback to developers. With Prisma Cloud, you can set up VCS integrations to provide periodic scanning of repositories and real-time developer feedback within the pull/merge request workflow. Combined with customized enforcement, that feedback allows you to fine-tune when and where security feedback is surfaced and how security policies are enforced.

Path 2

This approach builds on top of IaC security automation in build-time by shifting IaC security feedback further left. By empowering developers to address misconfigurations and exposed secrets earlier, before code is committed, feedback cycles are faster and cheaper. It also helps with general security awareness for developers who may be less familiar with cloud security best practices.

This path does rely on individuals to configure scanning in their local environments, making it difficult to enforce total adoption. However, because these local scans can also include custom configurations from the Prisma Cloud platform, users will quickly see that scanning locally before they push code saves them time later on.

Path 3

This approach enables the most complete adoption of IaC security. While it involves customization for each pipeline and takes the most implementation effort, it will enable you to take full advantage of IaC security features and automation capabilities at scale. Because CI/CD pipelines are the heartbeats of software development and deployment, embedding centralized and universal guardrails is the best way to get final security assurance before cloud resources are provisioned.

This is especially important for frameworks like Terraform, wherein code resources become more representative of their runtime states the further along the pipeline they get. The CI/CD pipeline may be the first time security tools have access to dynamically imported values or Terraform plan files to identify the misconfigurations that would be created with an apply command.

How to Start Implementing Your IaC Security Program

Once you have an end goal in mind for your IaC security program, you're ready to start rolling it out. To help make your program implementation as frictionless as possible, we recommend taking a crawl-walk-run approach.

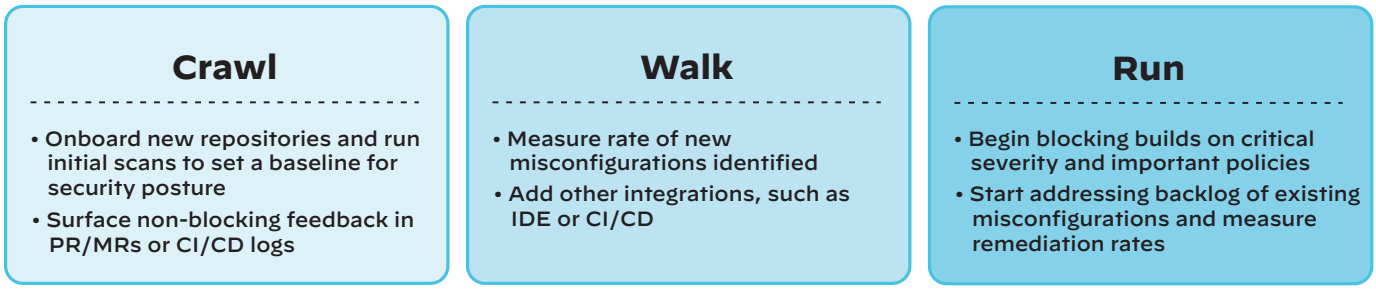

Figure 4: A crawl-walk-run approach operationalizes IaC security across the organization

Crawl: Onboard and Set Baseline

To begin, you should onboard no more than 50 repos so that you can get comfortable with the new security program, start to experiment with more customizable features, and tailor the program to your workflows.

Set Expectations

Before you actually enable any new integrations, it's important to set expectations with your team about how IaC security findings will be surfaced and enforced.

Start with Soft Fail Checks

Because hundreds of policies are being scanned and each organization has its own standards, we recommend starting with just non-blocking feedback via soft fail checks. Keep in mind that all violations in IaC and secrets are set to hard fail by default.

When you enable soft fail checks, regardless of your path, you are merely surfacing informational feedback. When time permits, developers can act on that feedback or move feedback into a backlog to work through over time, providing a powerful educational tool for developers to see specifics related to code risks. Flagging misconfigurations as informational also allows you to suppress policies that aren't applicable to specific teams.

Identify Benchmark KPIs

At this phase of your IaC security rollout, you should also start thinking about KPIs and set benchmarks, such as your baseline number of violations, to get the complete picture of how your program is performing so far.

a. First, assess your current runtime issues. Your ultimate goal is to reduce the number of runtime issues your team faces.
b. It's also important to measure leading indicators of runtime issues—the posture of your code, for example. To establish your baseline code posture, break down your violations by severity level and identify the average number of new issues you see each month.
c. Set goals to work towards, such as reducing new monthly violations by a specific percentage, or just track your progress over time.

Indicators of Success and Next Steps

If you're starting to see the rate of new misconfigurations and vulnerabilities decrease, that's a great indicator that your team is learning some cloud security basics, misconfigurations are being addressed earlier, and your policy library is fine-tuned to your needs. Before moving on to the next stage, reaffirm expectations with your team members, adjust KPIs if needed, and ensure you're gathering early adopter feedback and incorporating it into your approach as you expand the rollout across the rest of your organization.

Walk: Get Started with IaC Security Features to Customize and Automate Your Program

The next phase of your IaC security program is two-fold: expand coverage across repos and teams and start leveraging more mature IaC security features. By incorporating what you learned from your initial repo scans and feedback from your internal security champions, you'll be ready for the next phase.

Start Onboarding More Repos and Teams

Start with specific repos that aren't too widely used and more sophisticated dev teams that are confident implementing security controls.

Begin Adding Custom Policies

While built-in policies monitor and enforce a wide range of cloud configuration specifications, custom policies add a layer of customization that help if you need specialized configurations unique to your organization. And as early adopting teams get comfortable with the rollout, you can also start adding other integrations, such as an IDE or CI/CD pipelines. It's important to note that you should start introducing these integrations slowly in this step, and have them more fully embedded into workflows once you get to the run phase.

Explore Advanced Features

For Prisma Cloud customers who are shifting left from runtime security and onboarding developers, this is the point where more advanced features like Smart Fixes may become more valuable. Smart Fixes augment out-of-the-box platform fixes by leveraging your team's past actions when addressing misconfigurations. As you onboard more teams and start scanning more of your IaC, these Smart Fixes will get better at recommending misconfiguration remediations with specifications tailored to your organization. The more you can do to make it easier to fix identified issues, the better. Again, the goal is to see the number of new misconfigurations flagged go down.

Introduce Hard Fails

Now that you've expanded across repos and teams, you can introduce more stringent guardrails, like hard fails. We recommend setting custom enforcement rules for each code category—IaC, secrets, container images, and open source packages—by severity (working closely with the various teams to determine the right rules). Generally, it's a best practice to start by hard failing more critical misconfigurations and vulnerabilities and expanding from there. Once you've introduced these enforceable guardrails, you can scale up the process and further customize how and where you're surfacing feedback.

Run: Leverage More Advanced Security Features and Iterate on Your Rollout Strategy

At this point, you should start comparing your KPIs—especially remediation rates—across teams to evaluate your program as you accelerate scale. How are you performing against the KPIs you set earlier? Re-evaluate your performance and address areas that may need improvement as you start to expand the program to other teams.

Empower Your DevSecOps Evangelists

This is a great time to further push DevSecOps evangelists to educate and inspire others in the organization to become security champions. Highlighting key metrics, such as issues resolved preproduction or mean reduction in time to fix alerts, will help demonstrate the effectiveness of implementations to the broader organization. If you're able to bring security champions together, you can also foster cross-team knowledge sharing, which should help identify areas that still need improvement.

Start Addressing Your Backlog

Since many teams are now fully onboarded and starting to take advantage of the more advanced security functionality and automation capabilities with your IaC security program, this is also the phase when you should start addressing the existing backlog of issues. Each team will benefit from past learnings and customizations and will, undoubtedly, contribute new learnings. Ultimately, you will be able to start improving the remediation rate and mean time to remediate (MTTR).

Remeasure Performance and Fine-Tune Processes

This stage of the rollout also presents another opportunity to measure your performance, reflect on the rollout process, and build a continuous assessment and improvement process to help you reduce friction as you expand and maintain usage across the rest of your organization. Up until now, you've customized policies and suppressions across teams, provided automated feedback with PR comments or CI/CD jobs, and started measuring remediation rates. Now you can gather your performance metrics and take the feedback you've gathered from that experience to address common obstacles and improve your rollout strategy going forward.

Rolling Out Your Program Across Your Organization

Your program rollout will look slightly different for each team in your organization. Key learnings while onboarding the first teams will influence later teams' rollout strategies, and your teams may decide to pick different paths.

For example, your first team may only need to surface feedback on commits and pull/merge requests, so they would pick path 1 and implement VCS guardrails. But other teams may decide they need more complete IaC security by implementing CI/CD guardrails and get both local and VCS feedback, which will require they take path 3.

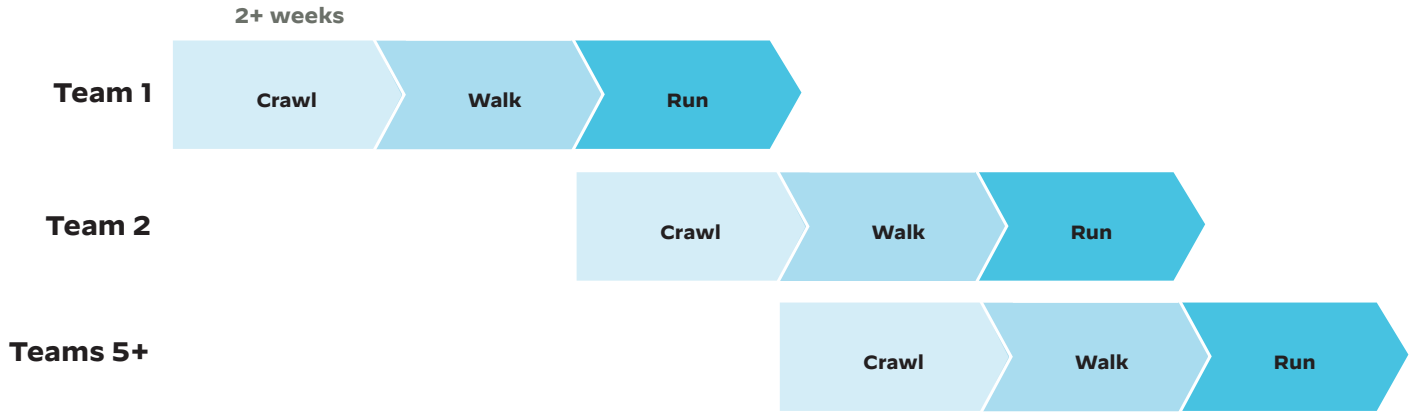

Figure 5: As you implement your IaC security program and stagger the rollout across teams, your organization's overall progress could look something like this

Next Steps—Expanding Code Security Use Cases

Implementing a robust IaC security program gives you the foundation you need to build and maintain a proactive, secure-by-default approach to IaC security. But you shouldn't stop there.

Leverage your experience rolling out IaC security to expand into other areas of code security, such as supply chain security and software composition analysis (SCA). Your organization will have some momentum on the heels of your IaC program rollout, so it's a great time to address your software supply chain security by implementing guardrails, such as automated enforcement of VCS and CI/CD policies. Given the ubiquity of open source in cloud-native codebases, it's a natural next step to start scanning for open source vulnerabilities and license compliance issues via a SCA solution.

About Prisma Cloud

Prisma® Cloud is the industry's most comprehensive cloud-native application protection platform (CNAPP) with the broadest security and compliance coverage—for applications, data, and the entire cloud-native technology stack—throughout the development lifecycle and across hybrid and multicloud environments. Our integrated approach enables security operations and DevOps teams to stay agile, collaborate effectively, and accelerate secure cloud-native application development. To learn more and see Prisma Cloud in action, request a free trial.

PRISMA® CLOUD | paloalto NETWORKS

3000 Tannery Way
Santa Clara, CA 95054

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

www.paloaltonetworks.com

prisma_complete-playbook-to-operationalizing-iac-security_010423