CI/CD Security Checklist

Six Best Practices to Proactively Address CI/CD Pipeline Weaknesses

CI/CD pipelines play a critical role in cloud-native software development, serving as the foundation in which developers store, compile, and deploy code. But CI/CD pipelines are frequently the weakest point in the software supply chain, and bad actors have taken note. CI/CD pipeline weaknesses provide an opportunity for bad actors to inject malicious code and leak sensitive data, thereby corrupting both the CI/CD pipeline and the software supply chain.

CI/CD security is often overlooked, and pipelines require special care in how they are configured and how user behavior is controlled. However, with the right approach, it's possible to proactively address CI/CD security and harden your pipelines over time.

Below are six tactical rules—a CI/CD security checklist—for proactively addressing CI/CD pipeline weaknesses so you can prevent issues like pipeline poisoning, secrets exfiltration, and dependency chain abuse.

1. Scan for Code Security Issues

Securing your CI/CD pipelines starts with securing the code stored, compiled, and deployed within those pipelines. This requires that you scan all your code before deployment, but you'll want to pay close attention to scanning your open-source code components because open-source code frequently contains vulnerabilities.

To ensure that the code inside your CI/CD pipeline is secure, you'll want to adopt proactive code security scanning that surfaces feedback into existing developer tools and workflows, like integrated development environments (IDEs), or as pre-commit hooks. It's important to scan everything from your infrastructure as code (IaC) files to your container images to identify all misconfigurations, vulnerabilities, and open-source license compliance issues within your codebase.

Recommended code security scanning actions:

- Adopt IaC scanning that surfaces feedback via IDEs and pre-commit hooks so you're addressing code security issues as early in the software development lifecycle as possible.
- Set up guardrails that automatically block code with severe issues from being added to your repositories and deployed into production.
- Scan your code repositories for known vulnerabilities in open-source packages using a software composition analysis (SCA) solution. Make sure to implement version bump fixes, which mitigate the risk of introducing breaking changes by bumping packages to only the smallest update that remediates the vulnerability.

2. Scan for CI/CD Pipeline Weaknesses

In addition to scanning the code within your CI/CD pipelines, you'll also need to scan the code that makes up the pipelines themselves. Even one misconfiguration in a CI/CD configuration file can result in exposed sensitive information, which bad actors can use to instigate attacks like malicious code injection and data leakage.

To get ahead of this risk, you can leverage the fact that CI/CD pipelines are configured in code to secure them. Take your existing policy-as-code approach to identifying and remediating misconfigurations and vulnerabilities to find and fix weaknesses in your pipelines.

You should adopt the following best practices to secure your CI/CD pipeline configurations:

- Scan your CI/CD configuration files using an IaC scanner and an SCA tool. With these solutions, you'll be able to identify and remediate misconfigurations and vulnerabilities in custom and open-source pipeline configuration files and can harden your pipelines over time.
- Apply your policy-as-code approach to your pipeline configuration files and proactively alert on weaknesses that could become larger issues over time, such as flagging if a developer attempts to add a deprecated command to a pipeline.
- Flag suspicious coding patterns within the CI/CD pipeline code as pull requests.

3. Find and Remove Exposed Credentials

Hard-coded credentials enable application developers to easily access or authenticate the services they need to build and deploy. But this practice introduces significant risk, since bad actors can leverage publicly exposed credentials to gain unauthorized access into your systems. This is true even for secrets that exist within private repos or behind a firewall, since mistakes happen and sometimes files can become public.

Adopting a proactive, developer-friendly secrets security strategy is the best defense against secrets exposure. It's a best practice to implement secrets scanning across your environment, but you'll want to pay special attention to scanning your CI/CD configuration files for exposed credentials. Secrets are oftentimes overlooked in delivery pipelines—which is a major security risk, given the strategic position CI/CD pipelines hold within the software supply chain.

To protect yourself against secrets exposure, adopt these best practices:

- Scan all major file types—IaC templates, CI/CD configuration files, golden images, and Git repos—for secrets across your environment. Prioritize remediating alerts from exposed credentials in CI/CD configuration files.
- Adopt a multidimensional secrets scanner with a fine-tuned entropy model to reduce false positives and noisy alerts.
- Implement thorough and deep secrets scanning. For example, if you're using IaC and CI/CD pipelines to pull in container images that run as application components or testing environments, you'll want to ensure your secrets scanner can identify and scan all images pulled from your IaC and CI/CD configuration files.

4. Implement Least-Privileged Access

Least-privileged access involves giving a user or group the minimum level of permissions needed to perform their assigned tasks and plays a major role in protecting organizations from risks, such as lateral movement and data exfiltration. Given the sheer number of identities that span development to deployment, it's critical that you're able to establish and maintain least-privileged access to any resource in your CI/CD pipeline to protect yourself from a widespread software supply chain attack.

Adopting the identity and access management (IAM) framework is a great first step to maintaining least-privileged access across your organization. You can leverage IAM policies to define user or group identities within your environment, and then associate permissions with those different identities.

To prevent permissions sprawl in your organization, you'll want to implement some best practices:

- Establish governance controls around all third-party services that require access to your CI/CD pipelines. Ensure that each service has least-privileged access, regularly audit permissions to remove any unused or duplicative permissions, and ensure that you have adequate visibility into each service's integration details.
- Avoid using shared accounts. Instead, create dedicated accounts for each context and implement least-privileged access for each of those accounts.
- Audit permissions to check that both machine and human identities have only enough permissions to do the specific tasks they're assigned.
- Revoke all stale identities—including inactive employees and systems that no longer require access to specific CI/CD pipelines.

5. Establish Proper Flow Control Mechanisms

CI/CD pipelines improve agility and release velocity, enabling new code to be written and deployed in minutes. While this speed—facilitated by automation—is critical to remaining agile, it also introduces the possibility of attack. Without a process of strict reviews and approvals—also known as flow control mechanisms—bad actors can easily push malicious code through the CI/CD pipeline and instigate a large-scale attack.

Protecting against this kind of attack requires you to set up flow control mechanisms that prevent any one person or machine from single-handedly pushing code or artifacts through the CI/CD pipeline without prior external verification or validation.

Here are some best practices to follow as you establish flow control mechanisms:

- Apply branch protection rules that prevent collaborators from forcing a push to the branch or deleting the branch entirely. For additional security, you can set finely tuned branch protection rules, such as adding detailed requirements the checks code must meet before being pushed to the branch.
- Require accounts to get external review or approval before triggering production build and/or deployment pipelines.
- Implement a drift detection solution to identify inconsistencies between runtime and build-time resources. Ensure that your solution provides fixes in code to help speed up remediation time.

6. Build Processes to Maintain Robust Logging and Visibility

To rapidly respond to an attack and protect critical systems, you need readily accessible information and as much advanced warning as possible. The right logging and monitoring measures enable this rapid response. Without these measures, though, it's possible that you may entirely miss the breach and encounter major obstacles in your efforts to investigate and remediate the issues that led to the attack.

A robust approach to logging and visibility into your CI/CD pipeline requires getting visibility into human and programmatic access, since programmatic access is a key vector in CI/CD pipeline attacks. Your approach should include generating both audit logs—human logs, which contain information such as user access and modifications to permissions—and application logs, which record events like artifact uploads, executions of builds, and pushes to a repository.

To build out logging and monitoring in your team, make sure to implement these best practices:

- Create an inventory of all systems that your team uses. For each system in your inventory, check that all relevant logs are enabled. Many systems do not enable logs by default.
- Store your logins in a central place so your security team can easily analyze logs across systems. This will help them detect and investigate security incidents faster.
- Enable system alerts to flag any unusual behavior for manual review.

Get Started

Get started with Code Security by Prisma Cloud to:

- Find and fix misconfigurations in cloud resources and IaC.
- Enforce hundreds of built-in policies across security and compliance benchmarks.
- Embed guardrails via IDE plugins, pre-commit hooks, and native VCS and CI/CD integrations.

About Prisma Cloud

Prisma® Cloud is a comprehensive Cloud-Native Security Platform with the industry's broadest security and compliance coverage—for applications, data, and the entire cloud-native technology stack—throughout the development lifecycle and across multicloud and hybrid deployments. Prisma Cloud's integrated approach enables security operations and DevOps teams to stay agile, collaborate effectively, and accelerate cloud-native application development and deployment securely. For more information, visit www.paloaltonetworks.com/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