Software Supply Chain Security Checklist

Seven Rules for Protecting Your Components and Pipelines from Attack

Manufacturing supply chains require strict quality control and security to ensure that parts are delivered safely, assembled properly, and end products are reliably shipped to the customer. Similarly, software supply chains are crucial to securely assembling and delivering applications from development to production.

As technology evolves and the software components and delivery pipelines that comprise supply chains get more complex, so do the requirements for securing them. Approaching software supply chain security requires securing both the individual components and the underlying delivery pipelines.

Below are seven tactical rules—a software supply chain security checklist—for securing software supply chains to identify, prioritize, and address risks faster to prevent supply chain security attacks.

1. Scan Infrastructure as Code (IaC) for Misconfigurations

IaC templates such as Terraform®, CloudFormation®, Azure® Resource Manager (ARM), and Kubernetes® manifests simplify provisioning and managing infrastructure deployments. However, they can be insecure by default.

To ensure your IaC is following infrastructure security best practices and set a baseline posture for the infrastructure that the application runs in, IaC security is crucial. Finding and fixing misconfigurations in IaC makes it more difficult to attack an application and lowers the blast radius for successful attacks. In addition, IaC templates might contain sensitive data such as private or privileged-access corporate data and information, and access to cloud environments. Recommended supply chain security actions:

- Implement IaC scanning via existing developer tools and workflows like integrated development environments (IDEs) or as pre-commit hooks to surface feedback as early as possible alongside other software testing.
- Embed IaC policy enforcement into version control systems (VCS) and continuous integration/continuous deployment (CI/CD) pipelines to ensure code is secure before deployment.
- Ensure every IaC deployment is triggered from a verified user on a known machine with a monitored network.

2. Scan Open Source Packages for Known Vulnerabilities

Modern applications consist of up to 98% open source components, most of which contain known vulnerabilities. To keep track of these vulnerabilities, databases such as Common Vulnerabilities and Exposures (CVE) provide all of the history, metadata, and background on known vulnerabilities.

Discovering those vulnerabilities within your code libraries and routinely upgrading or patching them is a foundational part of securing applications. Recommended supply chain security actions:

- Scan your local and hosted code repositories for known vulnerabilities, bumping packages when a vulnerability is identified.
- Continuously scan repositories to stay on top of newly released CVEs.
- Leverage risk scores such as CVSS and business criticality to prioritize updating packages when 100% patching isn't possible.

3. Scan Images Across the Development Lifecycle

Containers add another layer of abstraction that help build software once to run anywhere. But when it comes to security, the container abstraction can contain both misconfigurations and vulnerabilities.

For example, an application can use a vulnerable open-source package alongside a configuration exposing port 22 or root as the default user. Together, those issues make over-privileged access and, ultimately, complete control of a container far easier to attain.

Similarly, a Dockerfile could be fully locked down, but a vulnerability that enables remote code execution in the REST API of a web service could allow attackers to easily inject malicious code.

This is compounded by the fact that images are typically stored in and integrated from registries that may also be vulnerable to poisoning. Recommended supply chain security actions:

- Images should be scanned continuously at each stage—locally, at the build phase of a CI/CD pipeline, from the registry, and at runtime—to identify vulnerabilities, malware and misconfigurations that are continuously evolving.
- If an image is from an unknown source, it's also important to test that image in an image sandbox scanner to expose malware. This should be performed locally, in repositories after building images, in registries, and continuously at runtime.
- Check the trustworthiness of an image by only allowing trusted images from trusted sources to avoid image poisoning attacks.

4. Comply with VCS Security Best Practices

Your supply chain is only as secure as the system that stores and manages it. For most cloud native applications, that means using a VCS. Without the right access controls and branch protec-

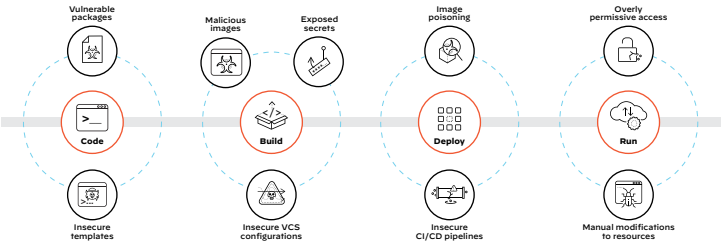

Figure 1: Software supply chain vulnerabilities

tions in place, however, a misconfigured VCS can be the perfect entrance for a supply chain attack. Recommended supply chain security actions:

- When configuring your VCS organization settings, ensure that you have two-factor authentication (2FA) and SSO enabled, and define an IP allow list for git users to prevent unauthorized access.
- Using branch protection rules, verify that commits have come from a trusted source and not from an impersonating identity since they can be easily forged.
- Ensure that repositories do not allow code to be committed without a human review.

5. Ensure Your CI/CD Pipelines Are Securely Configured

For cloud native organizations, CI/CD pipelines play a vital role in delivering and deploying code and thus also need to be hardened. Unvetted changes to workflow files can lead to exposing credentials if a bad actor checks in an update to a pipeline that exfiltrates the secret. It also can lead to code tampering. Recommended supply chain security actions:

- Scan CI/CD workflow files to prevent shell injections sourced in build triggers.
- Prevent the execution of deprecated and insecure commands in pipelines.
- Map secrets to environments and restrict access to pipeline execution on sensitive environments.
- Vet infrastructure that pipelines run on to ensure only the minimal access is allowed to run tests and deploy updates, with a separation of duties between those two tasks.

6. Avoid Secrets Exposure and Unauthorized Access

In addition to identifying weaknesses within each of these layers, it's also important to avoid unintentional exposure of secrets such as login credentials and access keys and tokens within them. Secrets can accidentally be included in IaC templates or CI/CD pipelines, leading to unauthorized access to your mission-critical infrastructure and services.

Recommended supply chain security actions:

- Store all credentials in a purpose-built vault such as HashiCorp Vault, AWS Parameter Store, and Azure Key Vault to keep secrets out of your code, VCS, or CI/CD configuration files.
- Rotate your keys periodically and always have a plan to disable, revoke, and create new credentials in the event that they do get exposed. It's also a best practice to have logging and auditing enabled across services to be able to ascertain whether any nefarious activities occurred with exposed secrets.
- Verify if usage of keys can be avoided using authentication solutions like OpenID connect, eliminating the need for keys in the first place.

7. Understand Risk with End-to-End Visibility

Securing the individual components of your supply chain is just as important as understanding how they connect and interact with each other. Being able to threat model how an attacker might gain access, pivot, and move laterally through chained weaknesses is an important component of supply chain security. Recommended supply chain security actions:

- Map all of your supply chain components with SBOMs and make sure you have complete code-to-cloud visibility so that you can prioritize and address security issues based on the real-world risk they present.
- When a supply chain attack does inevitably occur, it's also important to have that visibility so that you can quickly patch the right vulnerabilities, revoke the right access tokens, and lock down the right VMs to minimize the blast radius.

One Platform for Supply Chain Security

Prisma Cloud's latest enhancements bring overall visibility into the posture of your software supply chain. This, in combination with Prisma Cloud's existing compliance and vulnerability checks, ensure that the components that make up your applications and the pipeline that assembles them are secure.

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