3 Security Best Practices for CI/CD

CI/CD pipelines are at the core of daily operations for many businesses today. These processes, when set up correctly, help to keep the delivery process consistent by automating many manual tasks and providing visibility into how the software is being worked on.

CI/CD is also the place in your technology stack where your infrastructure has access to many different resources, from development and production environment to analytics keys and code signing credentials. The more resources your pipelines have access to (secure secrets, proprietary code, databases, etc.), the more important it is to keep your CI/CD system secure. Here we'll briefly cover 3 categories of security practices and how to address them.

1 Secure pipeline configuration

It is possible to use your CI/CD pipeline configuration to make security issues less likely to happen.

Safely store secrets that you use in your pipelines for connecting to databases and third-party services. On CircleCI, you have the option to use encrypted-at-rest environment variables, or to use contexts to restrict access to specific security group member. Never check these into your repository in plain text, even if the repository is private.

For sensitive files like code signing keys add an additional layer of isolation between your encrypted files and your repository for added security. See an example here. Keep them encrypted and only decrypt them inside your CI/CD jobs when they are needed.

Never leave your CI/CD environment running without monitoring it. Make sure that the containers and VMs used for your pipelines are destroyed after jobs that require sensitive information finish running. On CircleCI, this happens automatically.

By addressing these aspects you can significantly reduce the risk of security incidents due to your CI/CD pipeline configuration.

2 Code and Git history analysis

Git provides a comprehensive history of changes accessed through a CLI. This also means that any sensitive information that makes it into the Git history can be accessible to attackers if a Git repository is exposed, even if the most recent state of the repository doesn't contain the secrets anymore.

Trufflehog and GitLeaks are two tools that can help you identify secrets that have been committed to the codebase so that you can deactivate and replace them. Once your Git history is clear of secrets, ensure that your current revision doesn't contain any vulnerable dependencies.

Static Application Security Testing (SAST) techniques can look through the application in your commit and analyze its dependencies. If they contain any issues or known security vulnerabilities, your commit will be marked as insecure and won't be allowed to proceed to deployment.

Dynamic Application Security Testing (DAST) techniques spin up a copy of your production environment inside your CI job in order to scan the resulting containers and executables. The dynamic aspect helps the system catch dependencies which won't be caught by SAST.

3 Security policy enforcement

Some security aspects can't be statically checked based on known vulnerabilities, but rather are specific to your particular company, and therefore need to be codified as policies. These can take the form of automated or manual compliance checks.

Certain tasks like reviewing the list of all accounts that have access to your repositories, or making sure your onboarding and offboarding processes are in sync, will be manual. We recommend that you set reminders to perform these regularly.

Some tasks actually can be automated quite easily. For example, third-party services can provide a convenient way to codify a set of rules that will be matched against in your CI pipeline, e.g., proving compliance with the regulations that govern your data. If there is a mismatch, the build will fail.

To learn more about using your CI/CD pipeline to enforce security policy, see Vulnerability management with Docker and CI/CD.

To learn more about building security into your CI/CD pipeline, read our Ultimate Guide to CI/CD Security and DevSecOps or visit circleci.com.