Best practices for securing code in your supply chain

In this article

Guidance on how to protect the center of your supply chain—the code you write and the code you depend on.

About this guide

This guide describes the highest impact changes you can make to improve the security of your code. Each section outlines a change you can make to your processes to improve security. The highest impact changes are listed first.

What's the risk?

Key risks in the development process include:

- Using dependencies with security vulnerabilities that an attacker could exploit.
- Leaking authentication credentials or a token that an attacker could use to access your resources.
- Introducing a vulnerability to your own code that an attacker could exploit.

These risks open your resources and projects to attack and those risks are passed directly on to anyone who uses a package that you create. The following sections explain how you can protect yourself and your users from these risks.

Create a vulnerability management program for dependencies

You can secure the code you depend on by creating a vulnerability management program for dependencies. At a high level this should include processes to ensure that you:

1. Create an inventory of your dependencies.
2. Know when there is a security vulnerability in a dependency.
3. Enforce dependency reviews on your pull requests.

4. Assess the impact of that vulnerability on your code and decide what action to take.

Automatic inventory generation

As a first step, you want to make a complete inventory of your dependencies. The dependency graph for a repository shows you dependencies for supported ecosystems. If you check in your dependencies, or use other ecosystems, you will need to supplement this with data from 3rd party tools or by listing dependencies manually. If you have at least read access to the repository, you can export the dependency graph for the repository as an SPDX-compatible, Software Bill of Materials (SBOM), via the GitHub UI or GitHub REST API. For more information, see "Exporting a software bill of materials for your repository."

Automatic detection of vulnerabilities in dependencies

Dependabot can help you by monitoring your dependencies and notifying you when they contain a known vulnerability. You can even enable Dependabot to automatically raise pull requests that update the dependency to a secure version. For more information, see "About Dependabot alerts" and "About Dependabot security updates".

Automatic detection of vulnerabilities in pull requests

The dependency review action enforces a dependency review on your pull requests, making it easy for you to see if a pull request will introduce a vulnerable version of a dependency to your repository. When a vulnerability is detected, the dependency review action can block the pull request from merging. For more information, see "About dependency review."

Assessment of exposure to risk from a vulnerable dependency

When you discover you are using a vulnerable dependency, for example, a library or a framework, you must assess your project's level of exposure and determine what action to take. Vulnerabilities are usually reported with a severity score to show how severe their impact could be. The severity score is a useful guide but cannot tell you the full impact of the vulnerability on your code.

To assess the impact of a vulnerability on your code, you also need to consider how you use the library and determine how much risk that actually poses to your system. Maybe the vulnerability is part of a feature that you don't use, and you can update the affected library and continue with your normal release cycle. Or maybe your code is badly exposed to risk, and you need to update the affected library and ship an updated build right away. This decision depends on how you're using the library in your system, and is a decision that only you have the knowledge to make.

Secure your communication tokens

Code often needs to communicate with other systems over a network, and requires secrets (like a password, or an API key) to authenticate. Your system needs access to those secrets to run, but it's best practice to not include them in your source code. This is especially important for repositories to which many people might have access and critical for public repositories.

Automatic detection of secrets committed to a repository

Note: Secret scanning is available for organization-owned repositories in GitHub Enterprise Server if your enterprise has a license for GitHub Advanced Security. For more information, see "About secret scanning" and "About GitHub Advanced Security."

Note: Your site administrator must enable secret scanning for your GitHub Enterprise Server instance before you can use this feature. For more information, see "Configuring secret scanning for your appliance."

You may not be able to enable or disable secret scanning, if an enterprise owner has set a policy at the enterprise level. For more information, see "Enforcing policies for code security and analysis for your enterprise."

You can configure secret scanning to check for secrets issued by many service providers and to notify you when any are detected. You can also define custom patterns to detect additional secrets at the repository, organization, or enterprise level. For more information, see "About secret scanning" and "Secret scanning patterns."

Secure storage of secrets you use in GitHub Enterprise Server

Besides your code, you probably need to use secrets in other places. For example, to allow GitHub Actions workflows or Dependabot to communicate with other systems. For more information on how to securely store and use secrets, see "Using secrets in GitHub Actions" and "Configuring access to private registries for Dependabot."

Keep vulnerable coding patterns out of your repository

Note: Code scanning is available for organization-owned repositories in GitHub Enterprise Server. This feature requires a license for GitHub Advanced Security. For more information, see "About GitHub Advanced Security."

Note: Your site administrator must enable code scanning for your GitHub Enterprise Server instance before you can use this feature. For more information, see "Configuring code scanning for your appliance."

You may not be able to enable or disable code scanning if an enterprise owner has set a GitHub Advanced Security (GHAS) policy at the enterprise level. For more information, see "Enforcing policies for code security and analysis for your enterprise."

Create a pull request review process

You can improve the quality and security of your code by ensuring that all pull requests are reviewed and tested before they are merged. GitHub has many features you can use to control the review and merge process. To get started, see "About protected branches."

Scan your code for vulnerable patterns

Insecure code patterns are often difficult for reviewers to spot unaided. In addition to scanning your code for secrets, you can check it for patterns that are associated with security vulnerabilities. For example, a function that isn't memory-safe, or failing to escaping user input that could lead to an injection vulnerability. GitHub offers several different ways to approach both how and when you scan your code. To get started, see "About code scanning."

Next steps

- "Securing your end-to-end supply chain"
- "Best practices for securing accounts"
- "Best practices for securing your build system"