Best practices for preventing data leaks in your organization

In this article

Learn guidance and recommendations to help you avoid private or sensitive data present in your organization from being exposed.

About this guide

As an organization owner, preventing exposure of private or sensitive data should be a top priority. Whether intentional or accidental, data leaks can cause substantial risk to the parties involved. While GitHub takes measures to help protect you against data leaks, you are also responsible for administering your organization to harden security.

There are several key components when it comes to defending against data leaks:

- Taking a proactive approach towards prevention
- Early detection of possible leaks
- Maintaining a mitigation plan when an incident occurs

The best approach will depend on the type of organization you're managing. For example, an organization that focuses on open source development might require looser controls than a fully commercial organization, to allow for external collaboration. This article provide high level guidance on the GitHub features and settings to consider, which you should implement according to your needs.

Secure accounts

Protect your organization's repositories and settings by implementing security best practices, including enabling 2FA and requiring it for all members, and establishing strong password guidelines.

- Enabling secure authentication processes by using SAML and SCIM integrations, as well as 2FA authentication whenever possible. For more information, see "About identity and access management with SAML single sign-on," "About SCIM for organizations," and "Securing your account with two-factor authentication (2FA)."
- Requiring organization members, outside collaborators, and billing managers to enable 2FA for their personal accounts, making it harder for malicious actors to access an organization's repositories and settings. This is one step further from

enabling secure autentication. For more information, see "Requiring two-factor authentication in your organization."

- Encouraging your users to create strong passwords and secure them appropriately, by following GitHub's recommended password guidelines. For more information, see "Creating a strong password."
- Establishing an internal security policy in GitHub, so users know the appropriate steps to take and who to contact if an incident is suspected. For more information, see "Adding a security policy to your repository."
- Encouraging your users to enable push protection for users so that no matter which public repository they push to, they will be protected. For more information, see "Push protection for users."

For more detailed information about securing accounts, see "Best practices for securing accounts."

Prevent data leaks

As an organization owner, you should limit and review access as appropriate for the type of your organization. Consider the following settings for tighter control:

Recommendation	More information
Disable the ability to fork repositories.	"Managing the forking policy for your repository"
Disable changing repository visiblity.	"Restricting repository visibility changes in your organization"
Restrict repository creation to private or internal.	"Restricting repository creation in your organization"
Disable repository deletion and transfer.	"Setting permissions for deleting or transferring repositories"
Scope personal access tokens to the minimum permissions necessary.	None
Secure your code by converting public repositories to private whenever appropriate. You can alert the repository owners of this change automatically using a GitHub App.	Prevent-Public-Repos in GitHub Marketplace
Confirm your organization's identity by verifying your domain and restricting email notifications to only verified email domains.	"Verifying or approving a domain for your organization" and "Restricting email notifications for your organization"
Ensure your organization has upgraded to the GitHub Customer Agreement instead of using the Standard Terms of Service.	"Upgrading to the GitHub Customer Agreement"
Prevent contributors from making accidental commits.	"Removing sensitive data from a repository"

Detect data leaks

No matter how well you tighten your organization to prevent data leaks, some may still occur, and you can respond by using secret scanning, the audit log, and branch protection rules.

Use secret scanning

Secret scanning helps secure code and keep secrets safe across organizations and repositories by scanning and detecting secrets that were accidentally committed over the full Git history of every branch in GitHub repositories. Any strings that match patterns provided by secret scanning partners, by other service providers, or defined by you or your organization, are reported as alerts in the **Security** tab of repositories.

There are two forms of secret scanning available: **Secret scanning alerts for partners** and **Secret scanning alerts for users**.

- Secret scanning alerts for partners—These are enabled by default and automatically run on all public repositories and public npm packages.
- Secret scanning alerts for users—To get additional scanning capabilities for your organization, you need to enable secret scanning alerts for users.

  When enabled, secret scanning alerts for users can be detected on the following types of repository:

  - Public repositories owned by organizations that use GitHub Enterprise Cloud (for free)
  - Private and internal repositorites when you have a license for GitHub Advanced Security

For more information about secret scanning, see "About secret scanning."

You can also enable secret scanning as a push protection for a repository or an organization. When you enable this feature, secret scanning prevents contributors from pushing code with a detected secret. For more information, see "Push protection for repositories and organizations." Finally, you can also extend the detection to include custom secret string structures. For more information, see "Defining custom patterns for secret scanning."

Review the audit log for your organization

You can also proactively secure IP and maintain compliance for your organization by leveraging your organization's audit log, along with the GraphQL Audit Log API. For more information, see "Reviewing the audit log for your organization" and "Interfaces."

Set up branch protection rules

To ensure that all code is properly reviewed prior to being merged into the default branch, you can enable branch protection. By setting branch protection rules, you can enforce certain workflows or requirements before a contributor can push changes. For more information, see "About protected branches."

Mitigate data leaks

If a user pushes sensitive data, ask them to remove it by using the `git filter-repo` tool or the BFG Repo-Cleaner open source tool. For more information, see "Removing sensitive data from a repository." Also, it is possible to revert almost anything in Git. For more information, see the GitHub Blog.

At the organization level, if you're unable to coordinate with the user who pushed the sensitive data to remove it, we recommend you contact GitHub Support with the concerning commit SHA.

If you're unable to coordinate directly with the repository owner to remove data that you're confident you own, you can fill out a DMCA takedown notice form and tell GitHub

Support. For more information, see DMCA takedown notice.

> **Note:** If one of your repositories has been taken down due to a false claim, you should fill out a DMCA counter notice form and alert GitHub Support. For more information, see DMCA counter notice.

Next steps

- "Best practices for securing code in your supply chain"
- "Best practices for securing your build system"