GitHub < >

GitHub Enterprise Cloud: Data Leak Prevention Best Practices

Prevention and Mitigation

Preventing private or sensitive data from being exposed is a top priority for all organizations. Whether they are intentional or accidental, data leaks open up substantial risk to the parties involved. GitHub takes all possible measures to help protect you against these incidents, however, it is ultimately a shared responsibility with your organization admins. There are two key components when it comes to defending against data leaks: taking a **proactive approach towards prevention** and **maintaining a mitigation plan** in the case where an incident occurs.

Prevention

Limit & Review Access

- Admins should consider the following settings for tighter controls:
  - **Restrict** repository creation to private or internal
  - **Disable** the ability to fork repositories
  - **Disable** changing repository visiblity
  - **Disable** repository deletion and transfer
- Personal access tokens should be scoped to the *minimum permissions necessary*
- Secure your code by converting public repositories to private whenever appropriate
  - You can alert the repository owners of this change automatically with this GitHub App
- Confirm your organization's identity by verifying your domain and restricting email notifications to only verified email domains
- Ensure your organization has upgraded to the Corporate ToS instead of Standard ToS

Accident Prevention

- Leverage Token Scanning (which is now automatically activated) to prevent leaking of certain secrets and tokens into public repositories
- Avoid accidental commits using these steps

Mitigation

User-Level Mitigation

- If a user commits sensitive information, they can remove sensitive data from a repository on their own
- Remember, it is possible to revert (almost) anything in Git

Org-Level Mitigation

- We recommend reaching out to GitHub Support with the concering commit SHA, as our team can help with mitigation by taking the following additional steps:
  - Delete the PR in staff tools to wipe out any relat ed audit logs, history, etc.
  - Confirm the commit hasn't appeared in other locations since the last check
  - Run garbage collection
  - Invalidate git cache on repo
- In cases where your team is confident about the ownership of the leaked data, fill out a DMCA takedown notice form and alert GitHub Support
- On the other hand, if your repository has been taken down due to a false claim, fill out a DMCA counter notice form and alert GitHub Support

General Approach to Account Security

- Ensure **secure authentication** processes by using **SAML** and **SCIM** integrations, as well as **2FA** whenever possible
- Create **strong passwords** and secure them properly following some of GitHub's recommended password guidelines
- Establish an **internal security policy** within GitHub, so your users know the approriate steps to take and who to contact if an incident occurs