Best practices for securing accounts

In this article

Guidance on how to protect accounts with access to your software supply chain.

About this guide

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

What's the risk?

Account security is fundamental to the security of your supply chain. If an attacker can take over your account on GitHub Enterprise Cloud, they can then make malicious changes to your code or build process. So your first goal should be to make it difficult for someone to take over your account and the accounts of other members of your organization or enterprise.

Centralize authentication

If you're an enterprise or organization owner, you can configure centralized authentication with SAML. While you can add or remove members manually, it's simpler and more secure to set up single sign-on (SSO) and SCIM between GitHub Enterprise Cloud and your SAML identity provider (IdP). This also simplifies the authentication process for all members of your enterprise.

You can configure SAML authentication for an enterprise or organization account. With SAML, you can grant access to the personal accounts of members of your enterprise or organization on GitHub.com through your IdP, or you can create and control the accounts that belong to your enterprise by using Enterprise Managed Users. For more information, see "About authentication for your enterprise."

After you configure SAML authentication, when members request access to your resources, they'll be directed to your SSO flow to ensure they are still recognized by your IdP. If they are unrecognized, their request is declined.

Some IdPs support a protocol called SCIM, which can automatically provision or deprovision access on GitHub Enterprise Cloud when you make changes on your IdP. With SCIM, you can simplify administration as your team grows, and you can quickly revoke access to accounts. SCIM is available for individual organizations on GitHub Enterprise Cloud, or for enterprises that use Enterprise Managed Users. For more information, see "About SCIM for organizations."

Configure two-factor authentication

Note: Starting in March 2023 and through the end of 2023, GitHub will gradually begin to require all users who contribute code on GitHub.com to enable one or more forms of two-factor authentication (2FA). If you are in an eligible group, you will receive a notification email when that group is selected for enrollment, marking the beginning of a 45-day 2FA enrollment period, and you will see banners asking you to enroll in 2FA on GitHub.com. If you don't receive a notification, then you are not part of a group required to enable 2FA, though we strongly recommend it.

For more information about the 2FA enrollment rollout, see this blog post.

The best way to improve the security of your accounts is to configure two-factor authentication (2FA). Passwords by themselves can be compromised by being guessable, by being reused on another site that's been compromised, or by social engineering, like phishing. 2FA makes it much more difficult for your accounts to be compromised, even if an attacker has your password.

As a best practice, to ensure both security and reliable access to your account, you should always have at least two second-factor credentials registered on your account. Extra credentials ensures that even if you lose access to one credential, you won't be locked out of your account.

Additionally, you should prefer passkeys and security keys over authenticator apps (called TOTP apps) and avoid use of SMS whenever possible. Both SMS-based 2FA and TOTP apps are vulnerable to phishing, and do not provide the same level of protection as passkeys and security keys. SMS is no longer recommended under the NIST 800-63B digital identity guidelines.

If service accounts in your organization have been selected for 2FA enrollment by GitHub, their tokens and keys will continue to work after the deadline without interruption. Only access to GitHub through the website UI will be blocked until the account has enabled 2FA. We recommend setting up TOTP as the second factor for service accounts, and storing the TOTP secret exposed during setup in your company's shared password manager, with access to the secrets controlled through SSO.

If you're an enterprise owner, you may be able to configure a policy to require 2FA for all organizations owned by your enterprise.

If you're an organization owner, then you may be able to require that all members of the organization enable 2FA.

To learn more about enabling 2FA on your own account, see "Configuring two-factor authentication." To learn more about requiring 2FA in your organization, see "Requiring two-factor authentication in your organization."

Configure your enterprise account

Enterprise owners may be able to require 2FA for all members of the enterprise. The availability of 2FA policies on GitHub Enterprise Cloud depends on how members authenticate to access your enterprise's resources.

If your enterprise uses Enterprise Managed Users or SAML authentication is enforced for your enterprise, you cannot configure 2FA on GitHub Enterprise Cloud. Someone with administrative access to your IdP must configure 2FA for the IdP.

For more information, see "About SAML for enterprise IAM" and "Enforcing policies for security settings in your enterprise."

Configure your personal account

> **Note**: Depending on the authentication method that an enterprise owner has configured for your enterprise on GitHub.com, you may not be able to enable 2FA for your personal account.

GitHub Enterprise Cloud supports several options for 2FA, and while any of them is better than nothing, the most secure option is a WebAuthn credential. WebAuthn requires an authenticator such as a FIDO2 hardware security key, a platform authenticator like Windows Hello, an Apple or Google phone, or a password manager. It's possible, although difficult, to phish other forms of 2FA (for example, someone asking you to read them your 6 digit one-time password). However WebAuthn is much more resistant to phishing, because domain scoping is built into the protocol, which prevents credentials from a website impersonating the login page from being used on GitHub Enterprise Cloud.

When you set up 2FA, you should always download the recovery codes and set up more than one 2FA credential. This ensures that access to your account doesn't depend on a single device. For more information, see "Configuring two-factor authentication" and "Configuring two-factor authentication recovery methods."

Configure your organization account

> **Note**: Depending on the authentication method that an enterprise owner has configured for your enterprise on GitHub.com, you may not be able to require 2FA for your organization.

If you're an organization owner, you can see which users don't have 2FA enabled, help them get set up, and then require 2FA for your organization. To guide you through that process, see:

1. "Viewing whether users in your organization have 2FA enabled"
2. "Preparing to require two-factor authentication in your organization"
3. "Requiring two-factor authentication in your organization"

Connect to GitHub Enterprise Cloud using SSH keys

There are other ways to interact with GitHub Enterprise Cloud beyond signing into the website. Many people authorize the code they push to GitHub with an SSH private key. For more information, see "About SSH."

Just like your account password, if an attacker were able to get your SSH private key, they could impersonate you and push malicious code to any repository you have write access for. If you store your SSH private key on a disk drive, it's a good idea to protect it with a passphrase. For more information, see "Working with SSH key passphrases."

Another option is to generate SSH keys on a hardware security key. You could use the same key you're using for 2FA. Hardware security keys are very difficult to compromise remotely, because the private SSH key remains on the hardware, and is not directly accessible from software. For more information, see "Generating a new SSH key and adding it to the ssh-agent."

Hardware-backed SSH keys are quite secure, but the hardware requirement might not work for some organizations. An alternative approach is to use SSH keys that are only valid for a short period of time, so even if the private key is compromised it can't be exploited for very long. This is the concept behind running your own SSH certificate authority. While this approach gives you a lot of control over how users authenticate, it also comes with the responsibility of maintaining an SSH certificate authority yourself. For more information, see "About SSH certificate authorities."

Next steps

- "Securing your end-to-end supply chain"
- "Best practices for securing code in your supply chain"
- "Best practices for securing your build system"
- "Best practices for preventing data leaks in your organization"