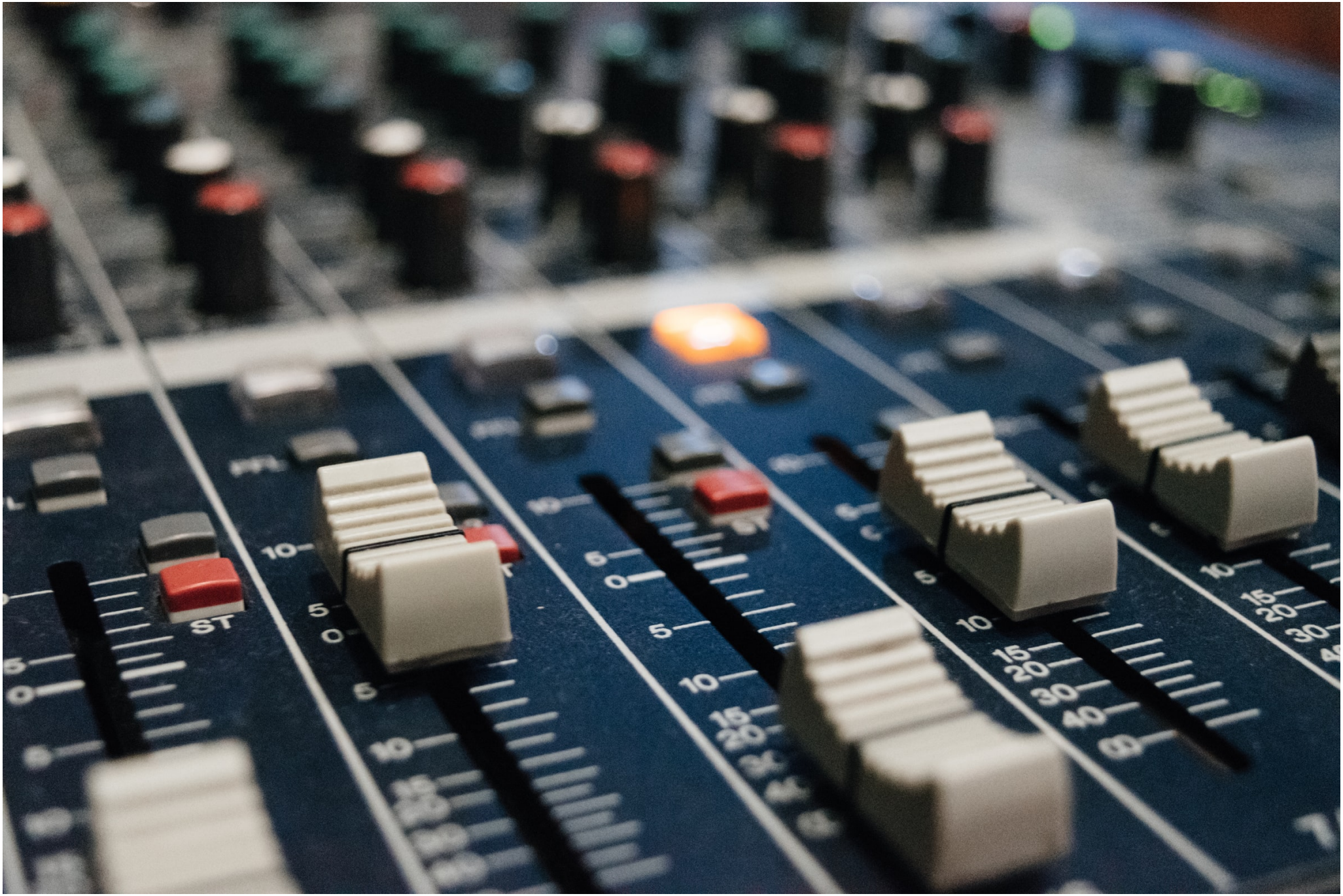
security security research tutorial

GitLab instance: security best practices

Mark Loveless · May 20, 2020 · 6 min read · Leave a comment


ST

GitLab is a feature-rich and powerful collaboration tool that is easy to use, and our self-managed installation is intended to be ready-to-go right out of the box. Exposing *any* service to the internet can create its own challenges from a security perspective, and as a result an administrator might have a bit of head-scratching over how to set things up safely.

Fortunately, we have a large number of security features and options that can be used to help lock things down. In this blog post, we've highlighted a few important features that will certainly help an administrator harden that new GitLab instance - particularly one facing the internet.

Access basics

During the initial GitLab installation, you will be asked to set up a root password. Obviously, we highly recommend a long password, unique to your GitLab instance that is not easily guessable with a mixture of uppercase and lowercase along with numbers and special characters. For a working example, see how we advise GitLab team members to create, store and manage passwords.

To help simplify your installation, consider using environment variables. The root password can also be set this way. For example:

```
GITLAB_ROOT_PASSWORD=hunter2 GITLAB_HOST=https://hunter2.instance apt install gitlab-ee
```

This has the added advantage of kicking off the entire letsencrypt process to ensure up-to-date certificates are used for your instance.

You will also want to ensure that users of your instance are also using strong, unique passwords, and you will want to ensure that the methods they use to access your instance are solid. Again, refer to our documentation on passwords for some ideas to share.

There are some choices you can make to limit access to data and restrict access to authorized users. In **Admin Area > Settings > General** you will want to expand the "visibility and access controls" section and make a few changes.

To help secure SSH access, RSA SSH keys should be allowed, as well as ED25519. Without going *too* deep, the open source crowd seems to prefer ED25519 as everything about it is open source (well-documented, trustworthy elliptical curve parameters), whereas other algorithms do not specify or go into details as to why they chose certain values. DSA also has a theoretical attack that could be used against it, although RSA could in theory fall to the same attack but is more resistant. Ah, but I digress! The main reason to support both RSA and ED25519 is that older systems that will connect may not be set up for ED25519, but will still support RSA, so at least both are recommended. With respect to RSA, encourage your users to use 2048 bits or higher when configuring keys.

We highly recommend using passwordless SSH authentication over password authentication. The communications are more secure (passwordless SSH authentication uses public/private key cryptography), it allows for an easier workflow, and it is one less password to worry about.

For more on SSH keys, see our documentation on ssh keys restrictions, as well as the additional visibility and access control settings that can be configured.

Restricting how and who

There are a few settings we recommend tweaking to help define how users access our instance and who we even allow to have access. You'll want to check out three areas in particular under the **Admin Area > Settings > General** settings.

Sign up restrictions:

- Ensure open sign-up is disabled on your instance. Open registration is disabled by default on self-managed instances with GitLab 13.6 and above installed. If new sign-up is enabled and your instance is open to the internet, anyone can sign up and access data. Administrators who would like to further restrict access on their instance can follow our documentation on how to configure user access.
- Make sure that Send confirmation email on sign-up" is checked. This adds a level of assurance that the user is in fact a real user.
- If you want to restrict access to a sub-group such as the users in your organization, consider configuring a whitelist for your organization's domain, (e.g., "example.com") which will allow them to sign up.
- Minimum password length: 12. For users that are allowed access, make sure they will be using longer passwords. See our password length limits documentation for details.
- For more detailed information, see our documentation around sign up restrictions

Sign in restrictions:

- Make sure that Require 2FA is enabled. Multifactor authentication is the more secure method of protecting authentication to a user's account, and is strongly encouraged.
- Disable "password authentication enabled for Git over HTTP(S)" if for some reason you can't require MFA. This will require users to use a personal access token, further securing the user accounts.
- For more detailed information, check our documentation around sign in restrictions.

Visibility and privacy: Ensure project visibility is set to "Private" on existing projects and by default for *new* projects. Private projects can only be cloned, downloaded, or viewed by project members, newly registered users will not be able to access these projects.

Improving performance and network tweaks

There are a few settings that will allow you to help protect your system from various network usage spikes, making your system a lot more stable and accessible for users.

User and IP rate limits

Going to **Admin Area > Network > User and IP rate limits** allows you to make a few adjustments. Specifically you will want all three items checked:

- "Enable unauthenticated request rate limit"
- "Enable authenticated API request rate limit"
- "Enable authenticated web request rate limit"

The default values associated with those items should be fine under most conditions. For more information, see our documentation around user and IP rate limits.

Webhooks

Webhooks are a useful feature with a lot of power. Unless there is a legitimate need to allow webhooks to communicate with internal services, they should be restricted to services that are publicly reachable, which you can verify in **Admin Area > Network > Outbound Requests**. While the "allow requests to the local network from web hooks and services" is disabled by default, you should also uncheck "allow requests to the local network from system hooks" as well. For more detail, including some of the dangers inherent in webhooks, see our webhooks documentation.

Protected paths

In **Admin Area > Network > Protected Paths** ensure that "Enable protected paths rate limit" has been checked. Default values should be more than sufficient. For details, check out our protected paths documentation.

Customize your configuration, harden your instance

We understand with security there is always a balance between protection and agility. In the cases of customers with internet-facing GitLab instances, there are often choices driven by a combination of different business drivers and needs. However, with the help of a few configuration tweaks you can harden your instance and better protect your organization, while still remaining open to the internet.

Additional settings, including those with security implications, can be found in the Admin Area. You'll want to explore those to really fine-tune your setup and make it your own. For some of you, these will have their own security implications that may be unique to your organization. Have fun exploring and securing your instance!

Cover image by Alexey Ruban on Unsplash

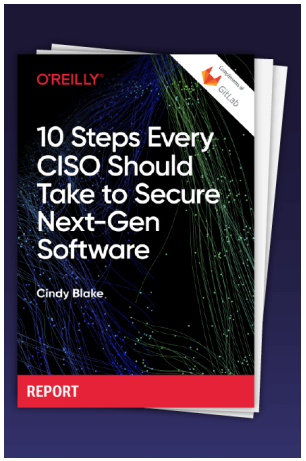
“Default settings are meant to save you time. When it comes to hardening your @GitLab instances, some extra #security configurations are time well spent” – Mark Loveless

Click to tweet

10 Steps Every CISO Should Take to Secure Next-Gen Software

Understand three software shifts impacting security, and the steps CISOs can take to protect their business.

Get the eBook



Tags: security security research tutorial

More to explore

SECURITY

Meet Package Hunter: A tool for detecting malicious code in your dependencies →


Dennis Appelt

SECURITY

How we're creating a threat model framework that works for GitLab →


Mark Loveless

SECURITY

A brief look at Gitpod, two bugs, and a quick fix →


Joern Schneeweisz

All Blog Posts →

Try all GitLab features - free for 30 days

GitLab is more than just source code management or CI/CD. It is a full software development lifecycle & DevOps tool in a single application.

Try GitLab Free (/free-trial/)

Be the first to comment.

Subscribe Add Disqus to your siteAdd DisqusAdd Do Not Sell My Data

Sign up for GitLab's twice-monthly newsletter

Your email address

I would like to receive the bi-weekly newsletter via email

(https://www.linkedin.com/company/gitlab-
(https://www.youtube.com/channel/UCnMGQ8QHMAnVIsI

Why GitLab?

Resources

Community

Customers (/customers/)

Product (/stages-devops-lifecycle/)

Solutions (/solutions/)

Services (/services/)

DevOps tools (/devops-tools/)

Is it any good? (/is-it-any-good/)

Releases (/releases/)

Pricing (/pricing/)

Get started (/get-started/)

All resources (/resources/)

All-Remote (/company/culture/all-remote/)

Blog (/blog/)

Newsletter (/company/contact/)

Events (/events/)

Webcasts (/webcast/)

Topics (/topics/)

Training (/learn/)

Docs (https://docs.gitlab.com/)

Install (/install/)

Contribute (/community/contribute/)

Community Programs (/community/)

Direction (/direction/)

Technology Partners (/partners/technology-partners/)

Channel Partners (/partners/)

Open Source Partners (/solutions/open-source/partners/)

GitLab for Open Source (/solutions/open-source/)

GitLab for Education (/solutions/education/)

GitLab for Startups (/solutions/startups/)

Shop (https://shop.gitlab.com)

Community Forum (https://forum.gitlab.com/)

Support

Get help (/get-help/)

Contact Sales (/sales/)

Company

About (/company/)