Phase 6: Rollout and scale secret scanning

In this article

1. Focus on newly committed secrets
2. Remediate previously committed secrets, starting with the most critical
3. Expand the program to include more secret types and custom patterns

---

For the final phase, you will focus on the rollout of secret scanning. Secret scanning is a more straightforward tool to rollout than code scanning, as it involves less configuration, but it's critical to have a strategy for handling new and old results.

> This article is part of a series on adopting GitHub Advanced Security at scale. For the previous article in this series, see "Phase 5: Rollout and scale code scanning."

You can enable secret scanning for individual repositories or for all repositories in an organization. For more information, see "Managing security and analysis settings for your repository" or "Managing security and analysis settings for your organization."

This article explains a high-level process focusing on enabling secret scanning for all repositories in an organization. The principles described in this article can still be applied even if you take a more staggered approach of enabling secret scanning for individual repositories.

1. Focus on newly committed secrets

When you enable secret scanning, you should focus on remediating any newly committed credentials detected by secret scanning. If you focus on cleaning up committed credentials, developers could continue to accidentally push new credentials, which means your total secret count will stay around the same level, not decrease as intended. This is why it is essential to stop new credentials being leaked before focusing on revoking any current secrets.

There are a few approaches for tackling newly committed credentials, but one example approach would be:

1. Notify: Use webhooks to ensure that any new secret alerts are seen by the right teams as quickly as possible. A webhook fires when a secret alert is either created, resolved, or reopened. You can then parse the webhook payload, and integrate it into any tools you and your team use such Slack, Teams, Splunk, or email. For more information, see "About webhooks" and "Webhook events and payloads."
2. Follow Up: Create a high-level remediation process that works for all secret types. For example, you could contact the developer who committed the secret and their technical lead on that project, highlighting the dangers of committing secrets to GitHub, and asking the them to revoke, and update the detected secret.

> **Note:** You can automate this step. For large enterprises and organizations with hundreds of repositories, manually following up is unsustainable. You could incorporate automation into the webhook process defined in the first step. The webhook payload contains repository and organization information about the leaked secret. Using this information, you can contact the current maintainers on the repository and create an email/message to the responsible people or open an issue.

3. **Educate**: Create an internal training document assigned to the developer who committed the secret. Within this training document, you can explain the risks created by committing secrets and direct them to your best practice information about using secrets securely in development. If the a developer doesn't learn from the experience and continues to commit secrets, you could create an escalation process, but education usually works well.

Repeat the last two steps for any new secrets leaked. This process encourages developers to take responsibility for managing the secrets used in their code securely, and allows you to measure the reduction in newly committed secrets.

> **Note:** More advanced organizations may want to perform auto-remediation of certain types of secrets. There is an open-source initiative called GitHub Secret Scanner Auto Remediator which you can deploy into your AWS, Azure, or GCP environment and tailor to automatically revoke certain types of secrets based on what you define as the most critical. This is also an excellent way to react to new secrets being committed with a more automated approach.

2. Remediate previously committed secrets, starting with the most critical

After you have established a process to monitor, notify and remediate newly published secrets, you can start work on secrets committed before GitHub Advanced Security was introduced.

How you define your most critical secrets will depend on your organization's processes and integrations. For example, a company likely isn’t worried about a Slack Incoming Webhook secret if they don’t use Slack. You may find it useful to start by focusing on the top five most critical credential types for your organization.

Once you have decided on the secret types, you can do the following:

1. Define a process for remediating each type of secret. The actual procedure for each secret type is often drastically different. Write down the process for each type of secret in a document or internal knowledge base.

   > **Note:** When you create the process for revoking secrets, try and give the responsibility for revoking secrets to the team maintaining the repository instead of a central team. One of the principles of GHAS is developers taking ownership of security and having the responsibility of fixing security issues, especially if they have created them.

2. When you have created the process that teams will follow for revoking credentials, you can collate information about the types of secrets and other metadata associated with the leaked secrets so you can discern who to communicate the new process to.

   You can use security overview to collect this information. For more information about using security overview, see "Filtering alerts in security overview."

   Some information you may want to collect includes:

- Organization
- Repository
- Secret type
- Secret value
- Maintainers on repository to contact

> **Note:** Use the UI if you have few secrets leaked of that type. If you have hundreds of leaked secrets, use the API to collect information. For more information, see "Secret scanning."

3. After you collect information about leaked secrets, create a targeted communication plan for the users who maintain the repositories affected by each secret type. You could use email, messaging, or even create GitHub issues in the affected repositories. If you can use APIs provided by these tools to send out the communications in an automated manner, this will make it easier for you to scale across multiple secret types.

3. Expand the program to include more secret types and custom patterns

You can now expand beyond the five most critical secret types into a more comprehensive list, with an additional focus on education. You can repeat the previous step, remediating previously committed secrets, for the different secret types you have targeted.

You can also include more of the custom patterns collated in the earlier phases and invite security teams and developer teams to submit more patterns, establishing a process for submitting new patterns as new secret types are created. For more information, see "Defining custom patterns for secret scanning."

You can also enable push protection with secret scanning. Once enabled, secret scanning checks pushes for high-confidence secrets and blocks the push. For more information, see "Push protection for repositories and organizations."

As you continue to build your remediation processes for other secret types, start to create proactive training material that can be shared with all developers of GitHub in your organization. Until this point, a lot of the focus has been reactive. It is an excellent idea to shift focus to being proactive and encourage developers not to push credentials to GitHub in the first place. This can be achieved in multiple ways but creating a short document explaining the risks and reasons would be a great place to start.

> This is the final article of a series on adopting GitHub Advanced Security at scale. If you have questions or need support, see the section on GitHub Support and Professional Services in "Introduction to adopting GitHub Advanced Security at scale."