Phase 2: Preparing to enable at scale

In this article

Preparing to enable code scanning

Preparing to enable secret scanning

In this phase you will prepare developers and collect data about your repositories to ensure your teams are ready and you have everything you need for pilot programs and rolling out code scanning and secret scanning.

This article is part of a series on adopting GitHub Advanced Security at scale. For the previous article in this series, see "Phase 1: Align on your rollout strategy and goals."

Preparing to enable code scanning

Code scanning is a feature that you use to analyze the code in a GitHub repository to find security vulnerabilities and coding errors. Any problems identified by the analysis are shown in GitHub Enterprise Server. For more information, see "About code scanning."

Rolling code scanning out across hundreds of repositories can be difficult, especially when done inefficiently. Following these steps will ensure your rollout is both efficient and successful. As part of your preparation, you will work with your teams, use automation to collect data about your repositories, and enable code scanning.

Preparing teams for code scanning

First, prepare your teams to use code scanning. The more teams that use code scanning, the more data you'll have to drive remediation plans and monitor progress on your rollout. During this phase, focus on leveraging APIs and running internal enablement events.

Your core focus should be preparing as many teams to use code scanning as possible. You can also encourage teams to remediate appropriately, but we recommend prioritizing enablement and use of code scanning over fixing issues during this phase.

Collecting information about your repositories

You can programmatically gather information about the different programming languages used in your repositories and use that data to enable code scanning on all repositories that use the same language, using GitHub Enterprise Server's GraphQL API.

Note: To gather this data without manually running the GraphQL queries described in this article, you can use our publicly available tool. For more information, see the "ghas-enablement tool" repository.

If you want to gather information from repositories belonging to multiple organizations in your enterprise, you can use the query below to obtain the names of your organizations

and then feed those into repository query. Replace OCTO-ENTERPRISE with your enterprise name.

```
query {
  enterprise(slug: "OCTO-ENTERPRISE") {
    organizations(first: 100) {
      totalCount
      nodes {
        name
      }
      pageInfo {
        endCursor
        hasNextPage
      }
    }
  }
}
```

You can identify which repositories use which languages by collating repositories by language at the organization level. You can modify the sample GraphQL query below, replacing OCTO-ORG with the organization name.

```
query {
  organization(login: "OCTO-ORG") {
    repositories(first: 100) {
      totalCount
      nodes {
        nameWithOwner
        languages(first: 100) {
          totalCount
          nodes {
            name
          }
        }
      }
      pageInfo {
        endCursor
        hasNextPage
      }
    }
  }
}
```

For more information about running GraphQL queries, see "Forming calls with GraphQL."

Then, convert the data from the GraphQL query into a readable format, such as a table.

Language	Number of Repos	Name of Repos
JavaScript (TypeScript)	4212	org/repo org/repo
Python	2012	org/repo org/repo
Go	983	org/repo org/repo
Java	412	org/repo org/repo
Swift	111	org/repo org/repo

Kotlin	82	org/repo org/repo
C	12	org/repo org/repo

You can filter out the languages that are currently not supported by GitHub Advanced Security from this table.

If you have repositories with multiple languages, you can format the GraphQL results as shown in the table below. Filter out languages that are not supported, but retain all repositories with at least one supported language. You can enable code scanning on these repositories, and all supported languages will be scanned.

Language(s)	Number of Repos	Name of Repos
JavaScript/Python/Go	16	org/repo org/repo
Rust/TypeScript/Python	12	org/repo org/repo

An understanding of which repositories are using which languages will help you identify candidate repositories for pilot programs in phase 3, and prepares you to enable code scanning across all repositories, one language at a time, in phase 5.

Enabling code scanning for your appliance

Before you can proceed with pilot programs and rolling out code scanning across your enterprise, you must first enable code scanning for your appliance. For more information, see "Configuring code scanning for your appliance."

Preparing to enable secret scanning

> **Note:** When secret scanning detects a secret in repositories owned by organizations that use GitHub Enterprise Cloud and have a license for GitHub Advanced Security, GitHub alerts all users with access to security alerts for the repository.

If a project communicates with an external service, it might use a token or private key for authentication. If you check a secret into a repository, anyone who has read access to the repository can use the secret to access the external service with your privileges. Secret scanning will scan your entire Git history on all branches present in your GitHub repositories for secrets and alert you or block the push containing the secret. For more information, see "About secret scanning."

Considerations when enabling secret scanning

GitHub Enterprise Server's secret scanning capability is slightly different from code scanning since it requires no specific configuration per programming language or per repository and less configuration overall to get started. This means enabling secret scanning at the organizational level can be easy but clicking **Enable All** at the organization level and ticking the option **Automatically enable secret scanning for every new repository** has some downstream effects that you should be aware of:

License consumption

Enabling secret scanning for all repositories will consume all your licenses, even if no one

is using code scanning. This is fine unless you plan to increase the number of active developers in your organization. If the number of active developers is likely to increase in the coming months, you may exceed your license limit and then be unable to use GitHub Advanced Security on newly created repositories.

Initial high volume of detected secrets

If you are enabling secret scanning on a large organization, be prepared to see a high number of secrets found. Sometimes this comes as a shock to organizations and the alarm is raised. If you would like to turn on secret scanning across all repositories at once, plan for how you will respond to multiple alerts across the organization.

Secret scanning can be enabled for individual repositories. For more information, see "Configuring secret scanning for your repositories." Secret scanning can also be enabled for all repositories in your organization, as described above. For more information on enabling for all repositories, see "Managing security and analysis settings for your organization."

Custom patterns for secret scanning

Secret scanning detects a large number of default patterns but can also be configured to detect custom patterns, such as secret formats unique to your infrastructure or used by integrators that GitHub Enterprise Server's secret scanning does not currently detect. For more information about supported secrets for partner patterns, see "Secret scanning patterns."

As you audit your repositories and speak to security and developer teams, build a list of the secret types that you will later use to configure custom patterns for secret scanning. For more information, see "Defining custom patterns for secret scanning."

For the next article in this series, see "Phase 3: Pilot programs."