Mandiant

ASM GITHUB INTEGRATION

Mandiant Advantage Attack Surface Management (MA-ASM) provides two GitHub workflows which result in different types of Entities and Issues. Our token-based GitHub integration works with both GitHub organizations and accounts. Depending on which method you choose, the MA-ASM workflow varies.

- **Integrate with a GitHub Organization:** MA-ASM locates all of the users belonging to the integrated organization and creates a `GithubAccount` Entity for each of them.
- **Integrate with a GitHub Account:** MA-ASM creates a `GithubAccount` Entity of the user the token belongs to and `GithubRepository` Entities for all the repositories the token has access to (including private repositories owned by other users).

This integration eliminates the need for manual entry of the same data as Seeds.

Integration with GitHub requires two steps:

1. **Create a GitHub Access Token**
2. **Use GitHub Access Token for MA-ASM Integration**

Create a GitHub Access Token

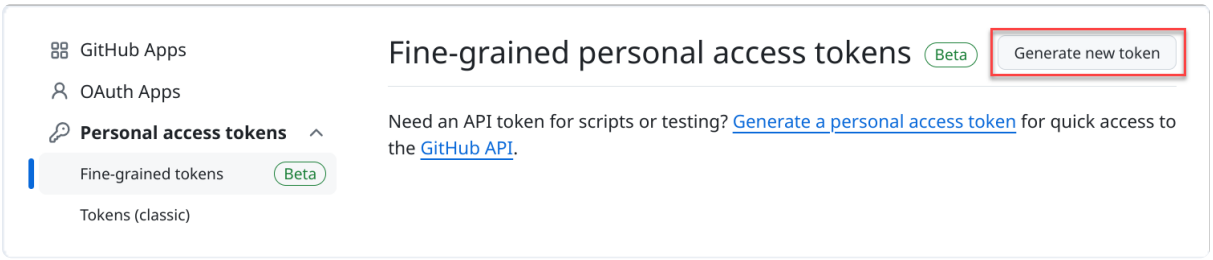
There are two types of access tokens you can create for the GitHub integration: Fine-grained or Classic. To create either type of token, sign in to GitHub and navigate to **https://github.com/settings/tokens**. Then follow the steps for the type of token you want to generate:

- **Fine-grained token (Beta)**: Recommended by GitHub
- **Classic token**: Less secure option

- See **Managing your personal access tokens (https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens)** to determine which token best suits your needs.
- See **https://github.com/settings/tokens** for more information on how to create a token to access the **GitHub API (https://docs.github.com/en/rest)**.

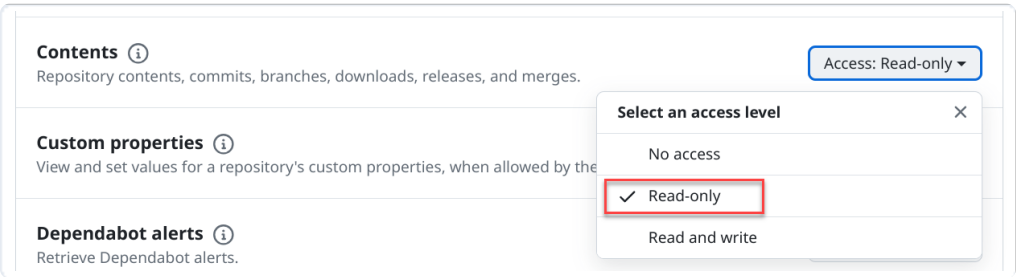
Create a fine-grained token (Beta)

1. Navigate to **Personal Access tokens > Fine-grained tokens**.
2. Click **Generate new token**.

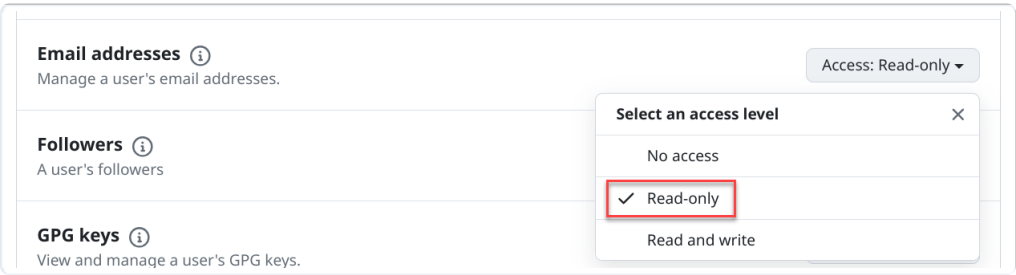


3. Fill in the fields as follows:
   - **Token name**: Enter a name such as *Mandiant-ASM* that's easy to remember.
   - **Expiration**: Set this based on your organizations's requirements and swap out the token when it expires.
   - **Description**
   - **Resource owner**

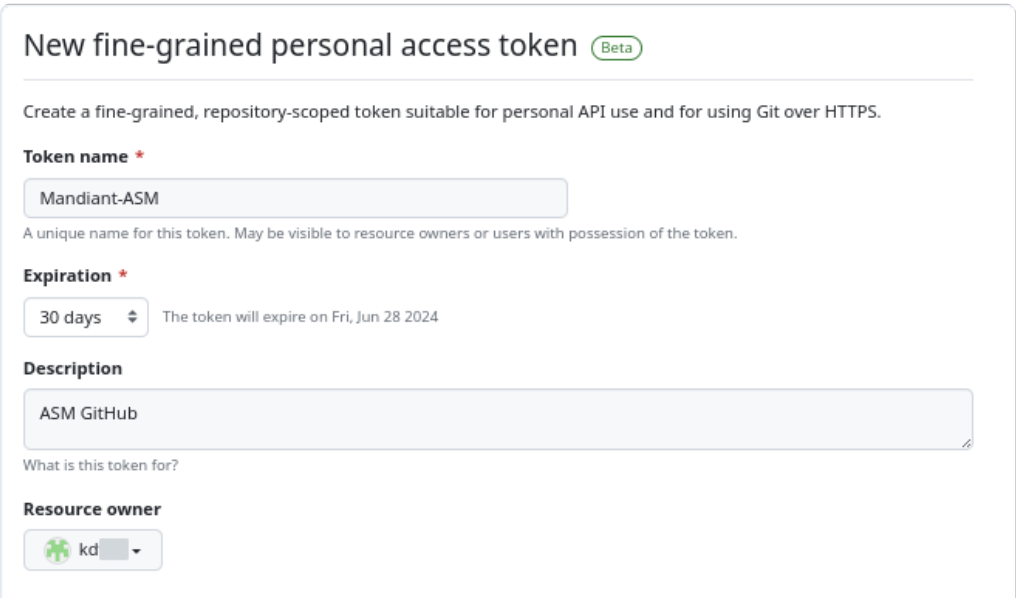
- **Repository access**: Select the choice which best fits your organization's requirements.
  - **Public Repositories (read-only)**
  - **All repositories**: This includes all current and future repositories you own.
  - **Only select repositories**: You must select at least one repository and no more than 50.
- **Permissions**
  - **Repository Permissions**: This subsection only appears if **All repositories** or **Only select repositories** is selected in the **Repository access** section.
    - Navigate to the **Contents** option and update this **Access** to **Read-only**.

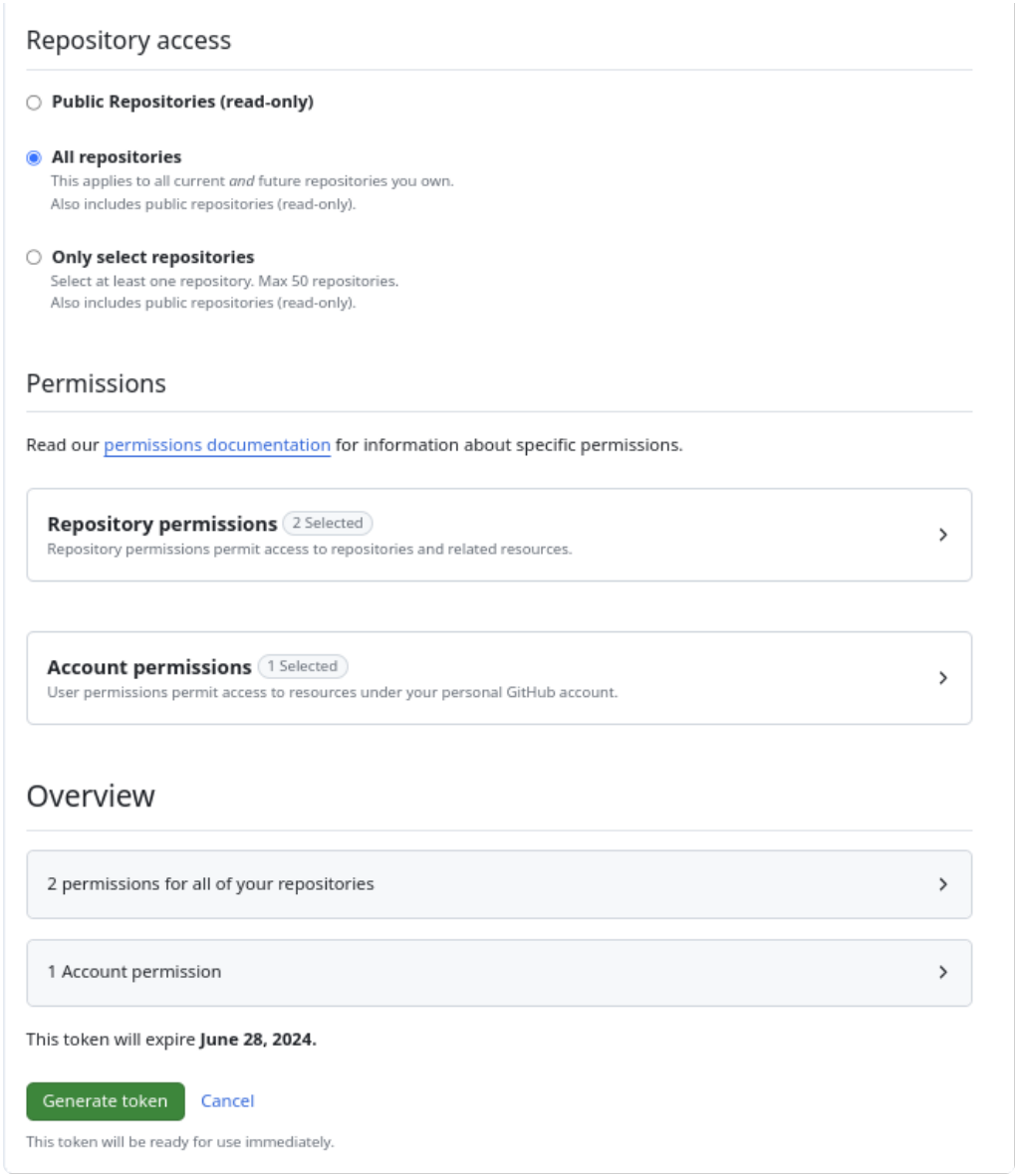


  - **Account Permissions**:
    - Optional: Navigate to the **Email addresses** option and update this **Access** to **Read-only**.



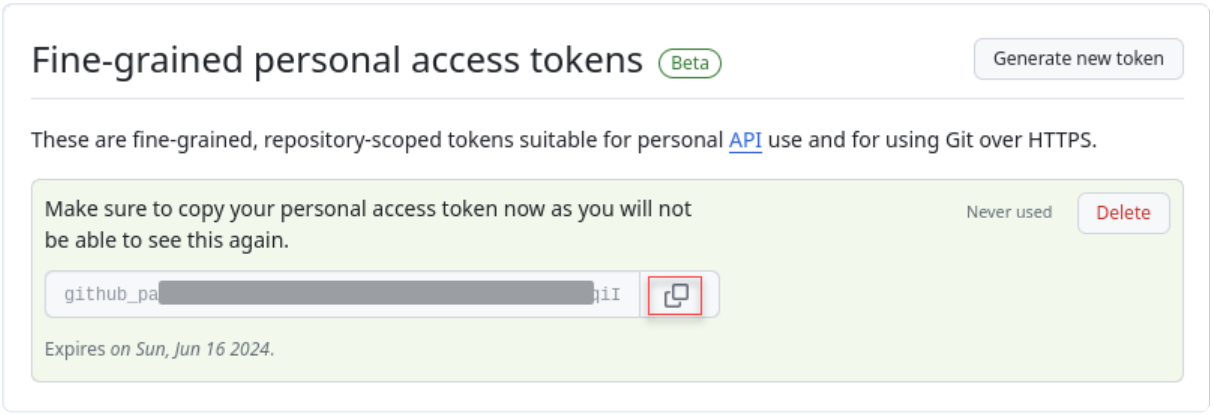
> Setting the Email addresses option to Read-only is recommended as this allows MA-ASM to display the email with which the access token is associated. This makes it easier to differentiate between many GitHub integrations.





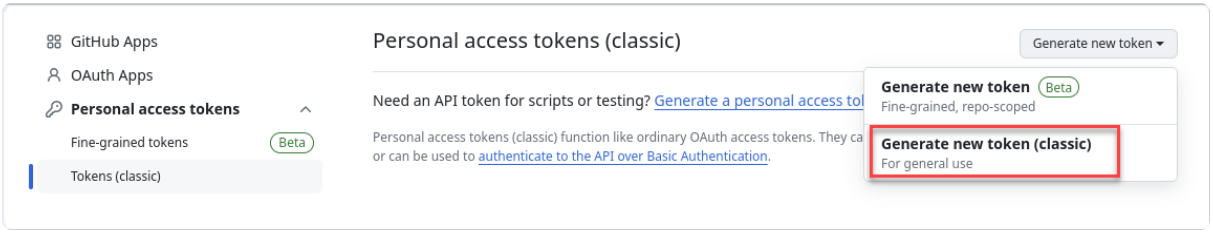
4. Click **Generate token**.
5. Finally, copy the token for use in MA-ASM when establishing the integration.

> This token is only available until you navigate away from this page.



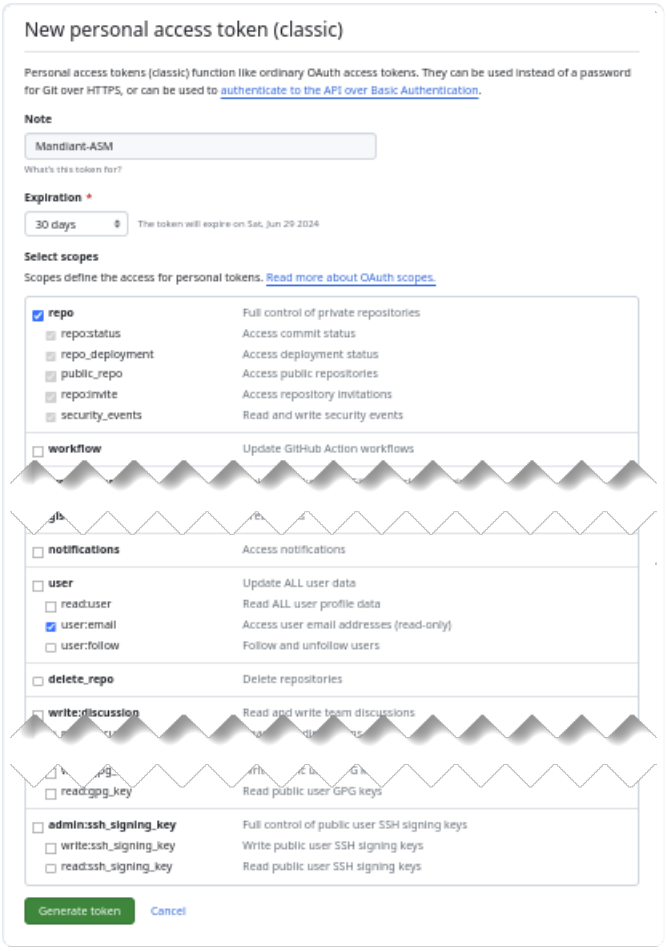
Create a classic token

1. Navigate to **Personal Access tokens > Tokens (classic)**.
2. Click **Generate new token > Generate new token (classic)**.



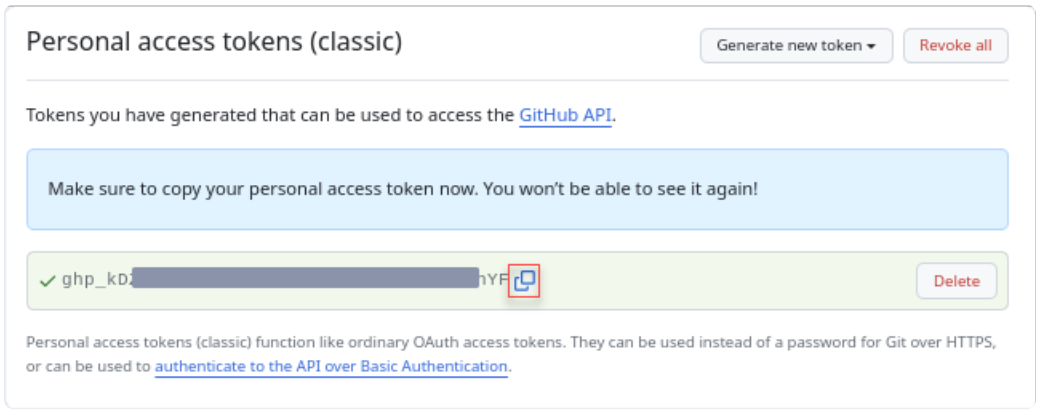
3. Fill out the mandatory fields as follows:
   - **Note**: Enter a name such as *Mandiant-ASM* that's easy to remember.
   - **Expiration**: Set this based on your organizations's requirements and swap out the token when it expires.
   - **Select scopes**: Select the following options:
     - **repo**: Full control of private repositories
     - Optional: **user > user:email**: Access user email addresses (read-only)

       If enabled, the **user:email** scope allows MA-ASM to display the email with which the access token is associated. This makes it easier to differentiate between many GitHub integrations.



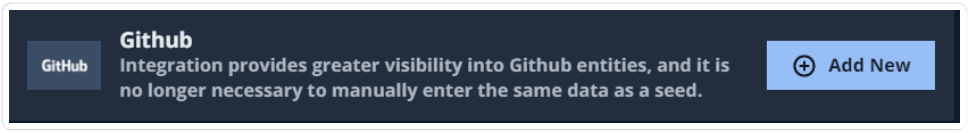
4. Finally, copy the token for use in MA-ASM when establishing the integration.

> This token is only available until you navigate away from this page.

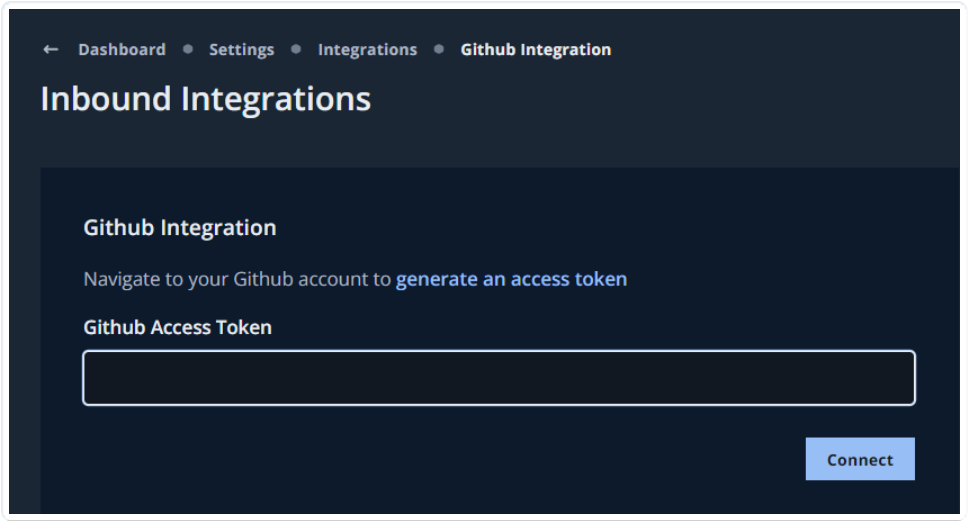


Use GitHub Access Token for MA-ASM Integration

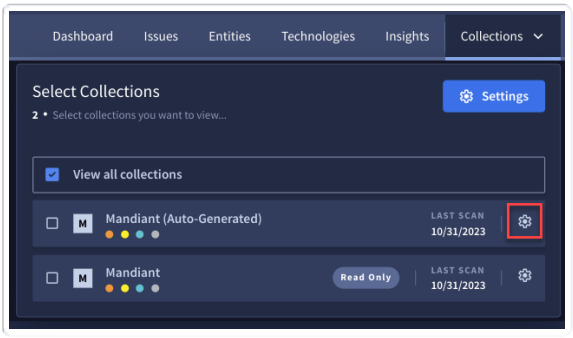
1. From the **Projects and Settings** menu in MA-ASM, select the appropriate Project then click **Account Settings**.
2. Click **Integrations**.
3. From **Inbound Integrations**, click **Add New** for **Github.**



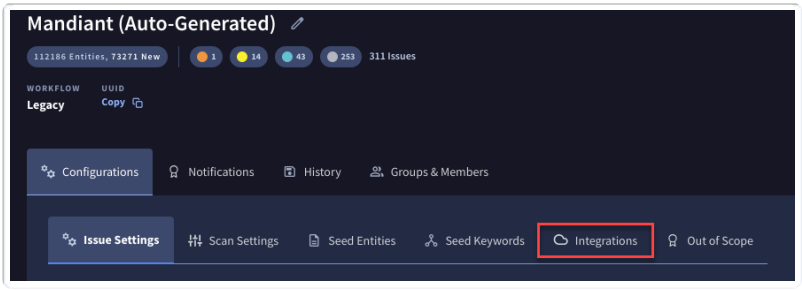
4. Paste the **Github Access Token** value into the **Github Access Token** field and click **Connect**.



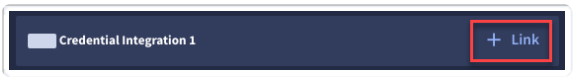
5. Connect the integration to the appropriate Collection.
   a. Click **Collections** and click **Collection Settings** for the Collection that you want to connect the integration to.



   b. Select the **Integrations** tab.

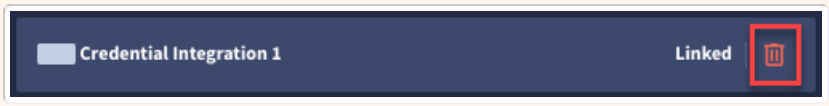


   c. Select **Connect Integration** and **Link** the integration.

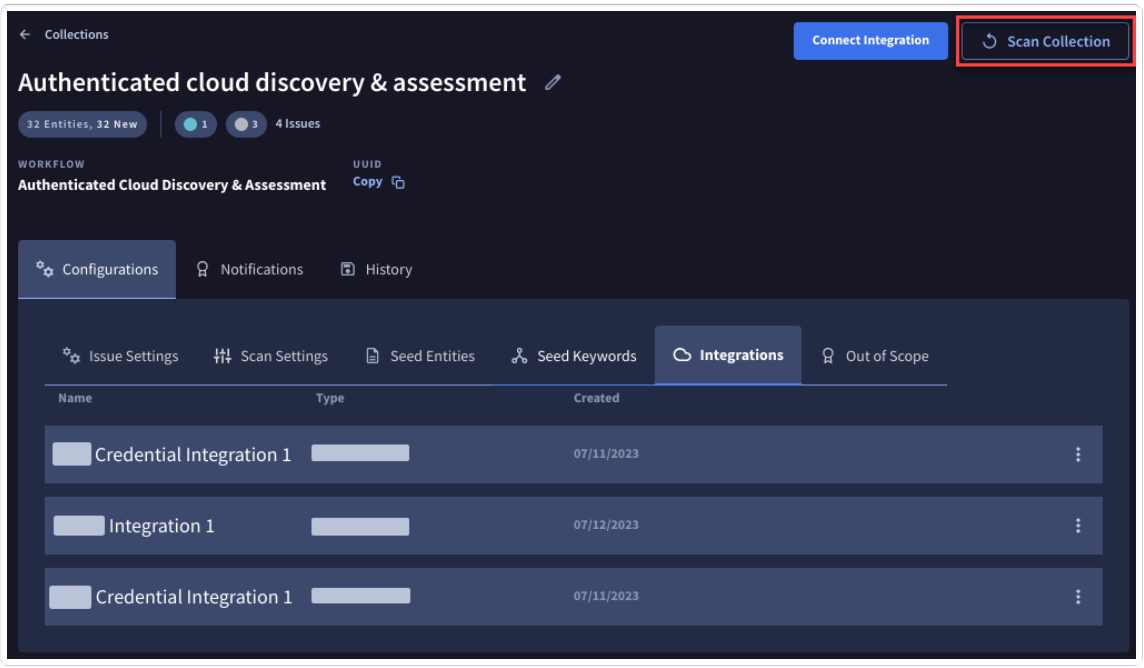


   The integration is immediately added to the Collection.

Click to remove the integration from this Collection.



d. Click ✕ to close the **Connect Integration** pane. Click **Scan Collection** to update your Collection with the current settings and integrations. Otherwise, your newly configured integration is incorporated at your regularly scheduled scan interval.



Keyword Seeds and the GitHub Integration

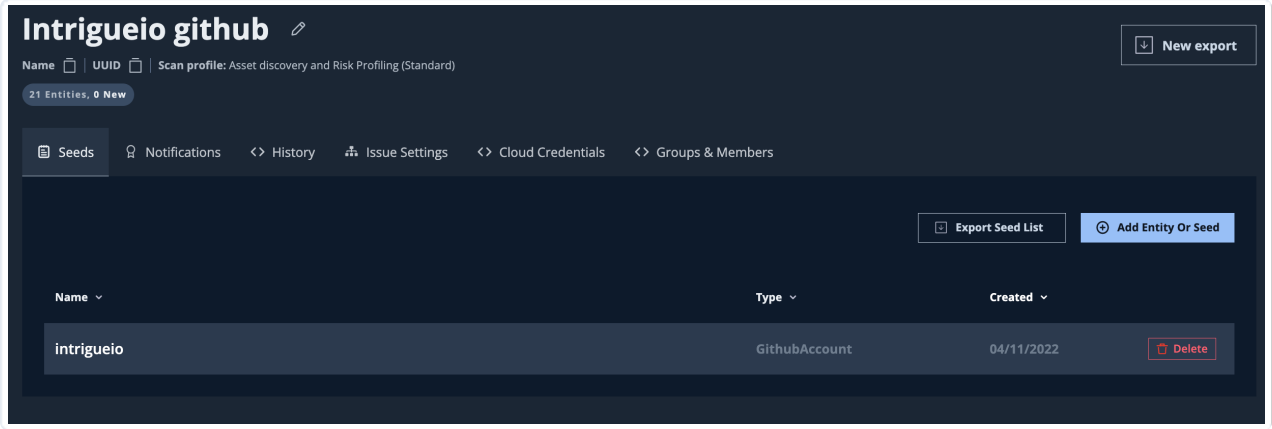
After integrating with GitHub, when a Seed is added (see **Creating & Seeding Collections (https://docs.mandiant.com/home/creating-seeding-collections)**) and used, MA-ASM works in the following ways:

- **GitHub Organization:** MA-ASM locates the public users and creates `GithubAccount` Entities for them.
- **GitHub Account:** For each `GithubAccount` Entity, available repositories are created as `GithubRepository` Entities.
- **GitHub Repository:** MA-ASM uses **Gitleaks (https://github.com/zricethezav/gitleaks)** with a modified configuration file to detect items such as: common API keys, tokens, private keys, suspicious file names, or file extensions.

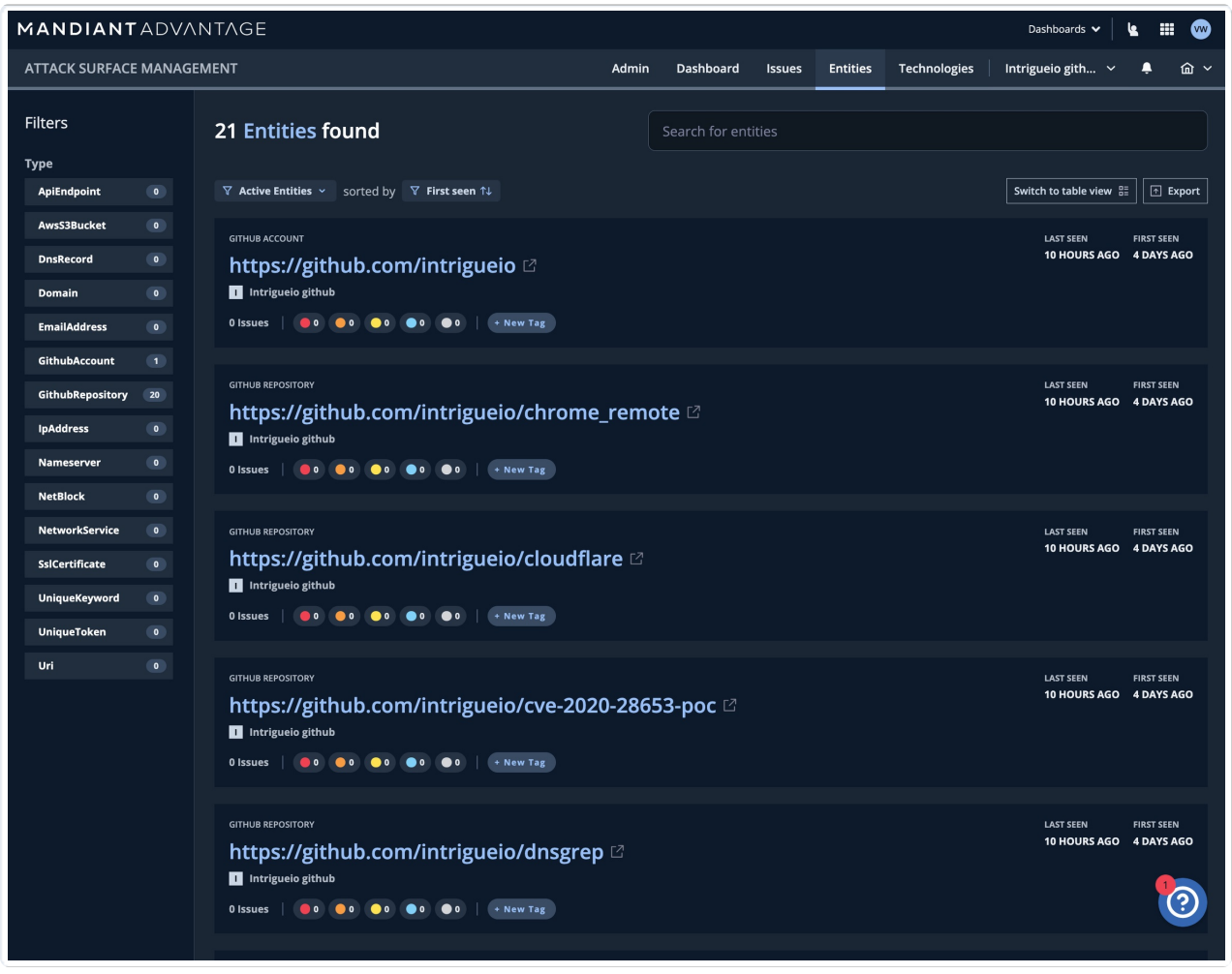
An Example Use Case

This example illustrates a sample MA-ASM GitHub tracking workflow.

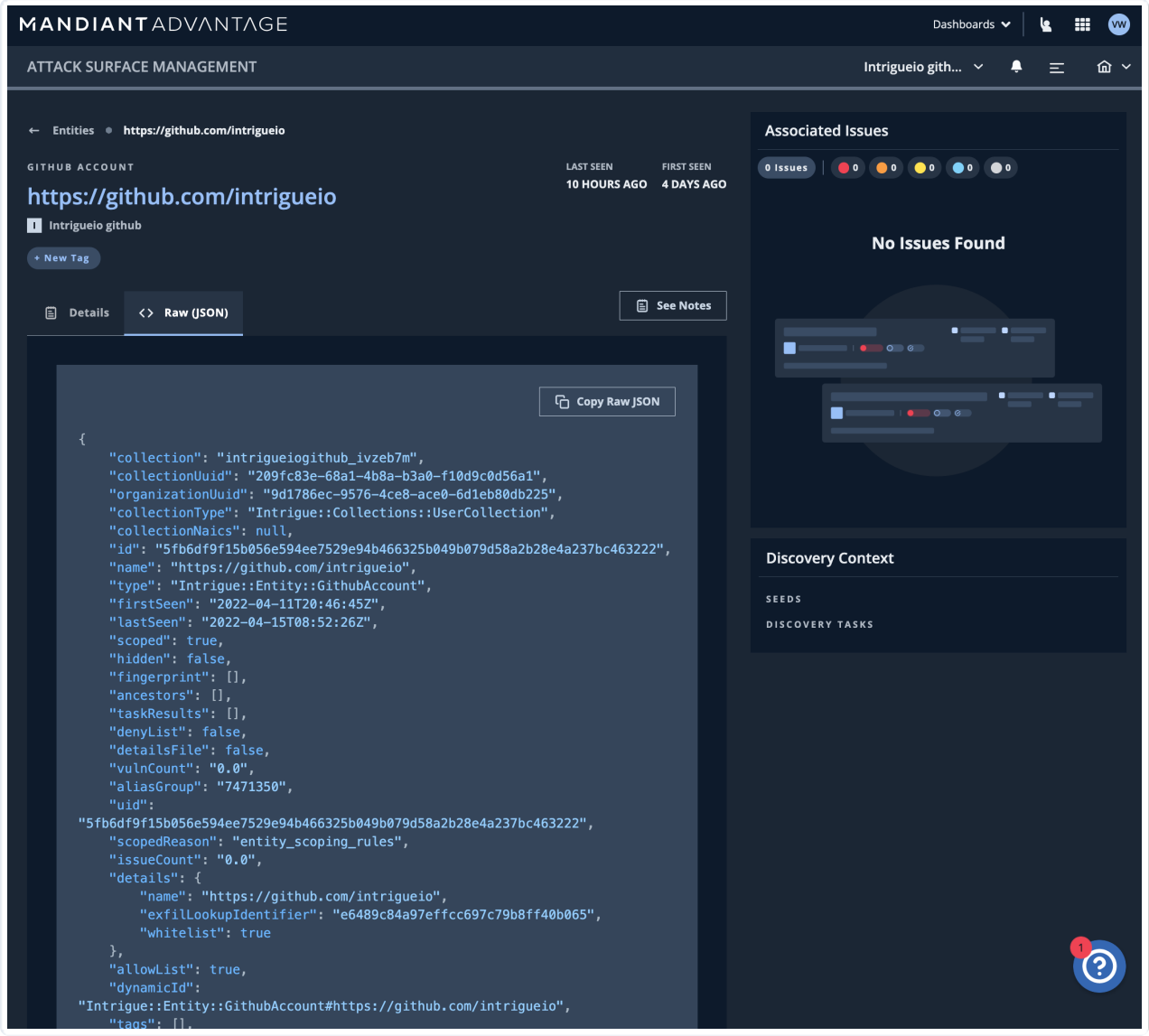
In this example, the user adds a Seed, a `GithubAccount` : **"intrigueio."**



Using that `GithubAccount`, MA-ASM finds 21 related Entities which include related repositories:



Selecting the first `GithubAccount` Entity from this list displays the following page of detailed information:

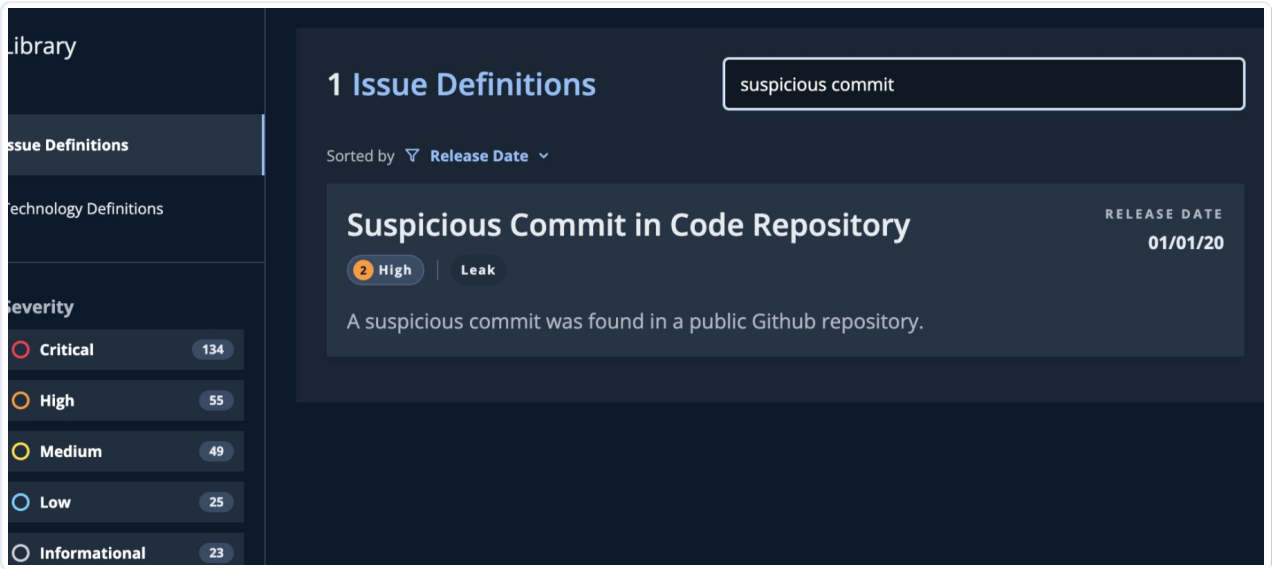


When a keyword is used as a Seed, MA-ASM turns on the GitHub tracking workflow for this collection. Keywords can contain phrases and are not required to be one word. This keyword finds related GitHub Repositories and if any have leaked secrets, an Issue type of **"Suspicious commit"** is created.

Two limitations associated with this type of workflow include:

1. Choosing poor keywords or phrases can create an overload of irrelevant results.
2. The API rate limit can result in incomplete search results.

If any Issues are found relating to that keyword, they show up as a **Suspicious Commit in Code Repository** Issue:



In some instances, you could find 1000 `GithubRepository` Entities from a keyword, and they could all have no leaked secrets. In this case, no Issues would be created but you would still have the Entities created.