Integrate GitHub Custom Action with Password Safe

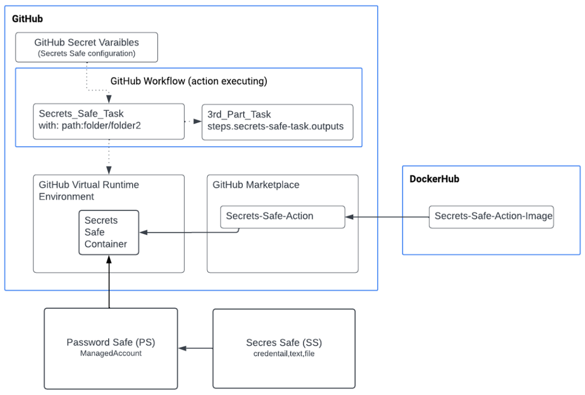
The Secrets Safe custom action is a GitHub integration for Secrets Safe, which enables using Secrets Safe secrets management capabilities with GitHub workflows.

Prerequisites

- The Secrets Safe action supports retrieval of secrets from BeyondInsight and Password Safe versions 23.1 and later releases.
- For this extension to retrieve a secret, the Secrets Safe instance must be preconfigured with the secret in question and authorized to read it.
- Runners must use a Linux operating system. Additionally, self-hosted runners must install Docker.

Overview

The Secrets Safe custom action retrieves ASCII secrets from BeyondTrust Secrets Safe and makes them available in the GitHub action workflow. Secrets are requested using either a Secrets Safe path or a path to a managed account which is composed of a managed system and account. The action output returns the secrets with an output_id specified in the action request. This allows immediate retrieval and usage of secrets from your BeyondTrust Secrets Safe instance. Retrieved secrets are masked on the GitHub runner used to retrieve the secret. This helps reduce the chance that secrets are printed or logged in error.



> ***Note:*** *It is important that security-minded engineers review workflow composition before changes are run with access to secrets.*

Configure Password Safe

> ***Note:*** *The following instructions are meant to be a quick start guide to help with Password Safe setup. For more information, please see the Password Safe Administration Guide at https://www.beyondtrust.com/docs/beyondinsight-password-safe/ps/admin/index.htm.*

Set Up OAuth Application User

1. Create an API access registration in BeyondInsight. Reduce your **Access Token Duration** to one that makes sense.
2. Create or use an existing Secrets Safe group.
3. Create or use an existing application user.
4. Add the API registration to the application user.
5. Add the user to the group.
6. Add the Secrets Safe feature to the group.

Set Up Managed Accounts

1. Create or use an existing access policy that has the **View Password Auto Approve** option set.
2. Add the **All Managed Accounts** Smart Group to the BeyondInsight group.
3. Add the access policy to the Password Safe role for the **All Managed Accounts** Smart Group, and ensure that both **Requestor** and **Approver** are checked in the role.
4. Create or use an existing managed system.
5. Create or use an existing managed account associated with the managed system.
6. Configure the managed account with the **API Enabled** and **Max Concurrent Requests Unlimited** options selected.

Setup for GitHub Custom Action

Inputs

- **client_id** (Required): API OAuth Client ID.
- **client_secret** (Required): API OAuth Client Secret.
- **api_url** (Required): BeyondTrust Password Safe API URL.
    - https://example.com:443/beyondtrust/api/public/V3
- **secret_path** (Required): Path of the secret to retrieve.

Note: *Special characters must be escaped.*

Example: *Secret Path*

```
[
   {
      "path": "folder1/folder2/title",
      "output_id": "title"
   },
   {
       "path": "folder1/folder2/title2",
       "output_id": "title2"
   }
[
```

- **managed_account_path** (Required): Path of the Managed account to retrieve.

Note: *Special characters must be escaped.*

Example: *Managed Account Path*

```
[
   {
      "path": "system/account",
      "output_id": "account"
   },
   {
       "path": "system/account2",
       "output_id": "account2"
   }
[
```

- **certificate**: Content of the certificate (cert.pem) for use when authenticating with an OAuth Client ID using a Client Certificate.
- **certificate_key**: Certificate private key (key.pem). For use when authenticating with an OAuth Client ID.
- **verify_ca**: Indicates whether to verify the certificate authority on the Secrets Safe instance. Defaults to **True** if not specified.
  - **VERIFY_CA**: True

***Note: False** means **insecure** and instructs the Secrets Safe custom action not to verify the certificate authority.*

- **log_level**: Level of logging verbosity. Default is **INFO**.
  - Levels:
    - CRITICAL
    - FATAL, ERROR
    - WARNING
    - WARN
    - INFO
    - DEBUG
    - NOTSET

Outputs

- **output_id**: The action stores the retrieved secrets in output variables defined by the end user. The **output_id** must be a unique identifier within the outputs object. The **output_id** must start with a letter or an underscore (_), and contain only alphanumeric characters, hyphens (-), or underscores. See **secret_path** and **managed_account_path**.

Example Usage

```
uses: BeyondTrust/secrets-safe-action@v1

env:
   API_URL: ${{vars.API_URL}}
   VERIFY_CA: ${{vars.VERIFY_CA}}
   CLIENT_ID: ${{secrets.CLIENT_ID}}
   CLIENT_SECRET: ${{secrets.CLIENT_SECRET}}
   CERTIFICATE: ${{secrets.CERTIFICATE}}
   CERTIFICATE_KEY: ${{secrets.CERTIFICATE_KEY}}
with:
   SECRET_PATH: '{"path": "folder1/folder2/title", "output_id": "title"}'
   MANAGED_ACCOUNT_PATH: '{"path": "system/account", "output_id": "account"}'
```

Example Usage

Download the pfx certificate from Secrets Safe and extract the certificate and the key to be pasted into a GitHub secret:

```
openssl pkcs12 -in client_certificate.pfx -nocerts -out ps_key.pem -nodes
openssl pkcs12 -in client_certificate.pfx -clcerts -nokeys -out ps_cert.pem
```

Copy the text from the ps_key.pem to a secret:

```
-----BEGIN PRIVATE KEY-----

...

-----END PRIVATE KEY-----
```

Copy the text from the ps_cert.pem to a secret:

```
-----BEGIN CERTIFICATE-----

...

-----END CERTIFICATE-----
```