SCC Tools: **Connector**

Security Cloud Command Center Tools

Contents

1 Introduction

Estimated time to complete the installation: ***15 minutes***

Installation guide for the **Connector** component of the SCC Tools package.

2 Requirements

Before start, make sure you've gone through the section **'How to install the tools'** in the main `README-${version}.pdf` file delivered in this package. It contains **important pre-requisites and pre-installation instructions you must do** to proceed to the installation of this tool.

3 Prepare the environment

Open **Google Cloud Shell** and upload the following file to your `${HOME}` directory:

- scc-connector-${version}.zip

Set the environment variables required by the installation scripts.

Note: *You must set them with values that make sense in your context, editing the snippet below before running the commands.*

```
# the scc tools release version you received.
export version=<release_version>

# directory to unzip the installation zip files.
export working_dir=${HOME}/scc-tools-install

# GCP Organization ID.
export organization_id=<your-organization_id>

# GCP Project ID to be created.
export connector_project_id=<your-connector-project>

# a valid billing Account ID (ask your Organization Administrator which one to use)
# [Billing accounts](https://console.cloud.google.com/billing)
export billing=<your-billing-account_id>

# the bucket used to upload the partner findings files which will be created if it does not exist
export connector_bucket=<your-connector-bucket>

# the bucket used to upload the cloud function source code which will be created if it does not exist
export connector_cf_bucket=<your-cloud-function-bucket>

# selected region listed from the list in the link below
# [Compute Engine Regions and Zones](https://cloud.google.com/compute/docs/regions-zones)
export region=<your-region>

# selected location listed from the list in the link below
# [App Engine Locations](https://cloud.google.com/appengine/docs/locations)
export gae_location=<your-gae-location>

# Absolute path to the Service Account file for the the Security Command Center API Project
export scc_sa_file=<absolute_path_to_service_account_file>

# comma separated values for custom roles that can be added to the deployer service account
export custom_roles=custom.gaeAppCreator
```

Unzip the uploaded file and enter the working directory:

```
# unzip the uploaded files to a work directory
unzip -qo scc-connector-${version}.zip -d ${working_dir}

# enter the installation working directory
cd ${working_dir}
```

4 Install the tool

Note: *If you want to simulate the execution of the following commands, use the option* `--simulation`.

4.1 Pre-requisites

- Python version 3.5.3
- A user with the curated role:
  - Pub/Sub Publisher - roles/pubsub.publisher

4.2 Steps for Billing Administrator

Create the GCP project and enable billing

```
(cd setup; \
pipenv run python3 create_project_with_billing.py \
  --organization_id ${organization_id} \
  --billing_account_id ${billing} \
  --project_id ${connector_project_id} \
  --no-simulation)
```

Enable the APIs needed by this project

```
(cd setup; \
pipenv run python3 enable_apis.py \
  --project_id ${connector_project_id} \
  --connector-apis \
  --no-simulation)
```

4.3 Steps for Deployer

- Creates the custom role needed to create App Engine applications (if it's not already created)
- Creates the service account that will be used to deploy
- Creates the partner project
- Creates the GCS buckets in the project, both for the partner findings and the cloud functions deploy
- Creates the Pub/sub topics in the project
- Deploys the cloud functions that are the connector app in the project
- Deploys a minimal AppEngine application in the project to enable Datastore
- Creates a pub/sub notification on the partner bucket in the project

Here you will be prompted about which translator you want to use. Please choose one by typing its number.

The translator works as an adapter to convert a security findings discovered by a partner from the partner format to the Security Command Center SourceFinding format. See the official documentation for details: Findings Create API reference

For now we have options of CAT, Cloudflare, Crowdstrike, Dome9, Forseti, Palo Alto, Qualys and Redlock.

When you select an option the corresponding yaml will be used to map the incoming jsons.

Also it can be changed later by updating the translator cloud function.

On these sample translators, Cloudflare, Crowdstrike, Dome9, Palo Alto and Redlock are examples that show how to convert partner findings where the partner finding don't have the Id of the Google Cloud Platform resource(Asset) in the format needed by the API.

Cloudflare is an example for fixing one single asset id for all the findings and Crowdstrike, Dome9, Palo Alto and Redlock are examples where you can define a map from values on the partner finding to valid asset ids for the SCC API.

It's recommended to use CAT for validation because it is the simplest to read and understand.

Run the next command to ensure you have the necessary custom role created in your organization:

```
(cd setup; \
pipenv run python3 create_custom_role.py \
  --custom_role_name custom.gaeAppCreator \
  --project_id ${connector_project_id} \
  --organization_id ${organization_id} \
  --deployment_name gae-creator-custom-role \
  --template_file templates/custom_gae_creator_role.py \
  --no-simulation)
```

Create the service account that will be used to deploy the application:

```
(cd setup; \
pipenv run python3 create_service_account.py \
  --name deployer \
  --project_id ${connector_project_id} \
  --organization_id ${organization_id} \
  --roles_file roles/connector.txt \
  --custom_roles ${custom_roles} \
  --no-simulation)
```

Run the setup script:

```
(cd setup; \
 export deploy_credentials=./service_accounts/${connector_project_id}_deployer.json;
 pipenv run python3 run_setup_connector.py \
  --organization_id ${organization_id} \
  --key_file ${deploy_credentials} \
  --billing_account_id ${billing} \
  --region ${region} \
  --gae_region ${gae_location} \
  --connector_project ${connector_project_id} \
  --connector_bucket ${connector_bucket} \
  --cf_bucket ${connector_cf_bucket} \
  --connector_sa_file ${scc_sa_file} \
  --no-simulation)
```

4.4 Configuring the application

You can configure the operation mode (prod or demo) of the Connector App publishing a message to a configuration topic:

It's recommended to set the application to demo mode for validation with the following script

```
gcloud pubsub topics publish projects/${connector_project_id}/topics/configuration \
  --message "{\"mode\": \"demo\"}"
```

On demo mode, to force the processing of the last loaded file to the application bucket you can use:

```
gcloud pubsub topics publish projects/${connector_project_id}/topics/flushbuffer \
  --message "{}"
```

Then set it to production mode:

```
gcloud pubsub topics publish projects/${connector_project_id}/topics/configuration \
  --message "{\"mode\": \"prod\"}"
```