Removing passwords from Helm Charts (REQ-235, OOM-2051): DCAE

2019-11-15/Jack Lucas

Background

The Helm charts for DCAE include passwords, stored in plain text. This isn't a good practice from a security perspective. The high-level requirement is to remove the passwords from the charts.

Where are the passwords?

For DCAE, passwords appear in the Helm charts in various places. The installation process uses the password information in different ways, depending on which component is being installed.

The sections that follow list where passwords can be found in the dcaegen2 Helm charts.

Bootstrap container

dcae-bootstrap/values.yaml:

- postgres-related passwords:
  - `postgres.config.pgPrimaryPassword`
  - `postgres.config.pgRootPassword`
  - `postgres.pgpool.credentials.pgpassword`

dcae-bootstrap/resources/config/dmaap-plugin.json:

- `dmaap.password`: not actually used and set to a nonsense value

This file is pushed into the Consul KV store by the bootstrap script.

dcae-bootstrap/resources/inputs/k8s-datafile-collector-inputs.yaml:

- `dmaap_mr_passwd`: not clear if it's actually needed, or just a placeholder
- `dmaap_dr_passwd`: not clear if it's actually needed, or just a placeholder

dcae-bootstrap/resources/inputs/k8s-k8s-pgaas-initdb-inputs.yaml:

- `k8s_initial_password`: taken from `postgres.config.pgRootPassword` from *values.yaml*

Cloudify Manager

dcae-cloudify-manager/values.yaml:

- `global.repositoryCred.password`: Docker repo password. Don't think it's needed.

Cloudify Manager protects access to its API and Web UI with a password. The password is set in a configuration file that's part of the container file system. We currently don't change it from the default setting. If we were to change it on a per deployment basis, we would need changes to

the Cloudify Manager container, to add a script that sets edits the configuration file to change the password. The CM version we're currently using (19.01.24) has a problem (in the containerized version that we use) that makes the reconfiguration operation unreliable. We would need to run the reconfiguration operation in order to change the password.

Config binding service

dcae-config-binding-service/values.yaml:

- `global.repositoryCred.password`: Docker repo password. Don't think it's needed.

DCAE dashboard

dcae-dashboard/values.yaml:

- postgres-related passwords:
  - `postgres.config.pgPrimaryPassword`
  - `postgres.config.pgUserPassword`
  - `postgres.config.pgRootPassword`
  - `postgres.pgpool.credentials.pgpassword`

dcae-dashboard/templates/deployment.yaml:

- `spec.spec.containers[0].env` sets postgres_password_dashboard as an environment variable

Deployment handler

dcae-deployment-handler/values.yaml:

- `global.repositoryCred.password`: Docker repo password. Don't think it's needed. *dcae-deployment-handler/templates/deployment.yaml:*
- `spec.spec.containers[0].env` sets CLOUDIFY_PASSWORD as an environment variable

DCAE healthcheck

No passwords

Policy handler

dcae-policy-handler/values.yaml:

- `global.repositoryCred.password`: Docker repo password. Don't think it's needed.

dcae-policy-handler/resources/config/config.json:

- `policy_engine.headers.ClientAuth`: auth info embedded in pre-constructed HTTP header
- `policy_engine.headers.Authorization`: auth info embedded in pre-constructed HTTP header

Note that these are ***not*** passwords--they're base-64 encoded strings. I do not know what exactly is in `ClientAuth` (it's not a standard HTTP header). `Authorization` is a standard header that consists of the user name and password, separated by ':' and base-64 encoded.

This file is pushed into the Consul KV store by the init-consul container.

dcae-policy-handler/templates/deployment.yaml:

- `spec.spec.containers[0].env` sets CLOUDIFY_PASSWORD as an environment variable

Redis

No passwords

Service change handler

dcae-servicechange-handler/values.yaml:

- `global.repositoryCred.password`: Docker repo password. Don't think it's needed.
- `sdc.password`: For authenticating to SDC.

dcae-servicechange-handler/resources/config/config.json:

- `asdcDistributionClient.password`: references `sdc.password` from values.yaml
- `asdcDistributionClient.keyStorePassword`: currently null

Inventory API

dcae-servicechange-handler/charts/dcae-inventory-api/values.yaml:

- `global.repositoryCred.password`: Docker repo password. Don't think it's needed.
- postgres-related passwords:
  - `postgres.config.pgPrimaryPassword`
  - `postgres.config.pgUserPassword`
  - `postgres.config.pgRootPassword`
  - `postgres.pgpool.credentials.pgpassword`

dcae-servicechange-handler/charts/dcae-inventory-api/resources/config/config.json:

- `databusControllerConnection.password`: currently set to null, DBC currently doesn't need password This file is used to create a configmap that is mounted at /opt/config.json.
- `server.applicationConnectors[0].keyStorePassword`: password for JKS keystore. Ideally this would come from the file set up by the TLS init container, but inventory relies on the Dropwizard framework. Dropwizard defines its own configuration file format.

This file is used to create a configmap that is mounted at /opt/config.json.

Implications

If the solution for replacing passwords keeps the current OOM charts as is, simply replacing the current passwords with placeholders that are filled in by a script that generates passwords,

almost all of DCAE will continue to work without changes. The exceptions:

- Cloudify Manager currently can't change its password at deployment time, so the password used by any script for CM would need to use the fixed password. Or we would need changes to the CM container.
- The policy handler doesn't store simple passwords. It would probably need to change to accept a simple password and generate the HTTP headers that are currently in the configuration file.

Note that this solution doesn't keep passwords encrypted after deployment--the passwords supplied by the script are stored in configmaps mounted as files or exposed as environment variables or stored in the Consul KV store, all unencrypted. But this solution would keep passwords out of the Helm charts and out of the source control system.

If the solution is more complex--for instance, if the solution involves creating Kubernetes secrets that hold generated passwords--then there would be numerous additional changes. The first step would be to look at the details of the solution and evaluate alternatives for changing DCAE applications to work with the solution.