Lab 2: Static Secrets

Duration: 25 minutes

This lab demonstrates both CLI commands and API to interact with key/value and cubbyhole secret engines.

- Task 1: Write Key/Value Secrets using CLI
- Task 2: List Secret Keys using CLI
- Task 3: Delete Secrets using CLI
- Task 4: Working with Key/Value Secret Engine using API
- Task 5: Hiding Secrets from History

Task 1: Write Key/Value Secrets using CLI

First, write your very first secrets in the key/value secret engine.

Step 2.1.1

First, check the current version of the key/value secret engine. Execute the following command:

```
$ vault secrets list -detailed
```

In the output, locate "`secret/`" and check its **version** under **Options**.

```
Path          Type         Accessor              ...  Options
----          ----         --------              ...  -------
cubbyhole/    cubbyhole    cubbyhole_8f752112    ...  map[]
identity/     identity     identity_8fb35fba     ...  map[]
secret/       kv           kv_00c670a4           ...  map[version:2]
...
```

Step 2.1.2

Execute the following command to read secrets at `secret/training` path:

```
$ vault kv get secret/training
```

Expected output:

```
No value found at secret/training"
```

Step 2.1.3

Write a secret into `secret/training` path:

```
$ vault kv put secret/training username="student01" password="pAssw0rd"
```

Expected output:

```
Key                 Value
---                 -----
created_time        2018-05-02T18:12:33.258249295Z
deletion_time       n/a
destroyed           false
version             1
```

Step 2.1.4

Now, read the secrets in `secret/training` path.

```
$ vault kv get secret/training
```

Expected output:

```
====== Metadata ======
Key                 Value
---                 -----
created_time        2018-05-02T18:12:33.258249295Z
deletion_time       n/a
destroyed           false
version             1

====== Data ======
Key            Value
---            -----
password       pAssw0rd
username       student01
```

Step 2.1.5

Retrieve only the **username** value from `secret/training`.

```
$ vault kv get -field=username secret/training
```

Expected output:

```
student01
```

Question

What will happen to the contents of the secret when you execute the following command:

```
$ vault kv put secret/training password="another-password"
```

Answer

Creates another version of the secret.

```
Key              Value
---              -----
created_time     2018-05-02T18:20:18.348234014Z
deletion_time    n/a
destroyed        false
version          2
```

When you read back the data, **username** no longer exists!

```
$ vault kv get secret/training

====== Metadata ======
Key              Value
---              -----
created_time     2018-05-02T18:20:18.348234014Z
deletion_time    n/a
destroyed        false
version          2

====== Data ======
Key         Value
---         -----
password    another-password
```

This is very important to understand. The key/value secret engine does **NOT** merge or add values. If you want to add/update a key, you must specify all the existing keys as well; otherwise, ***data loss*** can occur!

Step 2.1.6

If you wish to partially update the value, use `patch`:

```
$ vault kv patch secret/training course="Vault 101"
```

```
```

This time, you should see that the `course` value is added to the existing key.

```
$ vault kv get secret/training
...
====== Data ======
Key         Value
---         -----
course      Vault 101
password    another-password
```

Step 2.1.7

Review a file named, `data.json` in the `/workstation/vault` directory:

```
$ cat data.json
{
  "organization": "hashicorp",
  "region": "US-West",
  "zip_code": "94105"
}
```

Step 2.1.8

Now, let's upload the data from `data.json`:

```
$ vault kv put secret/company @data.json
```

Read the secret in the `secret/company` path:

```
$ vault kv get secret/company
====== Metadata ======
Key              Value
---              -----
created_time     2018-05-02T18:24:52.03750902Z
deletion_time    n/a
destroyed        false
version          1

======== Data ========
Key             Value
---             -----
organization    hashicorp
region          US-West
zip_code        94105
```

Task 2: List Secret Keys using CLI

Step 2.2.1

Get help on the list command:

```
$ vault kv list -h
```

This command can be used to list keys in a given secret engine.

Step 2.2.2

List all the secret keys stored in the key/value secret backend.

```
$ vault kv list secret
```

Expected output:

```
Keys
----
company
training
```

The output displays only the keys and not the values.

Task 3: Delete Secrets using CLI

Step 2.3.1

Get help on the delete command:

```
$ vault kv delete -h
```

This command deletes secrets and configuration from Vault at the given path.

Step 2.3.2

Delete `secret/company`:

```
$ vault kv delete secret/company
```

Step 2.3.3

Try reading the `secret/company` path again.

Expected output includes the `deletion_time`:

```
====== Metadata ======
Key                 Value
---                 -----
created_time        2018-05-02T18:24:52.03750902Z
deletion_time       2018-05-02T18:46:19.9948457Z
destroyed           false
version             1
```

NOTE: To permanently delete `secret/company`, use `vault kv destroy` or `vault kv metadata delete` commands instead.

Task 4: Working with Key/Value Secret Engine API

In this task, you are going to write, read, and delete secrets in key/value secret engine via API.

Step 2.4.1

To write secrets in the key/value secret engine via API using cURL:

```
$ curl --header "X-Vault-Token: <token>" --request POST \
       --data <data>
       <VAULT_ADDRESS>/v1/secret/data/<path>
```

Refer to the online API documentation for more detail:
https://www.vaultproject.io/api/secret/kv/kv-v2.html

Check the vault address on your student workstation:

```
$ echo $VAULT_ADDR
```

Expected output:

```
http://127.0.0.1:8200
```

Step 2.4.2

Execute the following cURL command to write data in `secret/apikey/google` path:

```
$ curl --header "X-Vault-Token: root" --request POST \
       --data '{"data": {"apikey": "my-api-key"} }' \
       $VAULT_ADDR/v1/secret/data/apikey/google | jq
```

In this exercise, parsing the output using `jq` tool just for the readability of the JSON response message.

> **NOTE:** If you are tailing the `audit.log` (optional step in Lab 1), you should see the trace log of this API call.

Step 2.4.3

Read the data in `secret/apikey/google` path:

```
$ curl --header "X-Vault-Token: root"  \
      $VAULT_ADDR/v1/secret/data/apikey/google | jq
```

Expected output:

```
{
  "request_id": "dda623da-ff4f-7417-f354-4dcfa68cff5e",
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": {
    "data": {
      "apikey": "my-api-key"
    },
    "metadata": {
      "created_time": "2018-05-02T18:59:24.293039655Z",
      "deletion_time": "",
      "destroyed": false,
      "version": 1
    }
  },
  "wrap_info": null,
  "warnings": null,
  "auth": null
}
```

Step 2.4.4

To retrieve the `apikey` value alone:

```
$ curl -s --header "X-Vault-Token: root" \
     $VAULT_ADDR/v1/secret/data/apikey/google | jq ".data.data.apikey"
```

Step 2.4.5

Delete the latest version of `secret/apikey/google` using API.

```
$ curl --header "X-Vault-Token: root" \
        --request DELETE \
        $VAULT_ADDR/v1/secret/data/apikey/google
```

Challenge

How can an organization protect the secrets in `secret/data/certificates` from being unintentionally overwritten?

Hint: - *Check-and-Set* parameter: https://www.vaultproject.io/docs/secrets/kv/kv-v2.html#writing-reading-arbitrary-data - Check the command options: `vault kv put -h`