vault-ca

Release alpha

Jan 15, 2021

Index:

CHAPTER 1

Vault CA

Set of tools to create your own CA and manage certificates using hashicorp Vault.

1.1 Install

```
apt-get install libffi-dev libssl-dev
pip install vault-ca
```

1.2 How to

1.2.1 Initialize the Vault

Vault need to be initialized without SSL certificates and than switched. Once initialized and configured, fetch-ssl-cert can be used to retrieve Vault's own certificate/key pair and TLS can be switched on.

Inside the the `example` folder you can find two different Vault server configurations for this purpose

```
 ~ vault server -config example/vault.hcl.init

 ~ export VAULT_ADDR="http://127.0.0.1:8200"

 ~ vault init
Unseal Key 1: SmMc4xXT2oq6d7uJSnMMnuRKLh7EighJUeH4Kh/28naG
Unseal Key 2: ovsPuWWd8dWVBqb+TPtqqQRjGPBNCor3rp1QLYYnhNZJ
Unseal Key 3: XuASk26YBqQo9+cvi17Me5o7PWsQwVlL2hM8G1bjOyIs
Unseal Key 4: NwhxHrkByuVfP4GURwDc/tZ1qPP6bVsmP2hUIQj+nvBN
Unseal Key 5: 0Kfq2jVlE2Db8Nj5mIXVhQTRONYz/0qZ1zm1BObUO4nx
Initial Root Token: 087efd94-3680-0ff4-25b7-a92177cd3c46
```

(continues on next page)

(continued from previous page)

```
Vault initialized with 5 keys and a key threshold of 3. Please
securely distribute the above keys. When the vault is re-sealed,
restarted, or stopped, you must provide at least 3 of these keys
to unseal it again.

Vault does not store the master key. Without at least 3 keys,
your vault will remain permanently sealed.

 ~ vault unseal SmMc4xXT2oq6d7uJSnMMnuRKLh7EighJUeH4Kh/28naG
Sealed: true
Key Shares: 5
Key Threshold: 3
Unseal Progress: 1
Unseal Nonce: 4ba54ac9-d60b-4078-05da-285ef05bf303
 ~ vault unseal ovsPuWWd8dWVBqb+TPtqqQRjGPBNCor3rp1QLYYnhNZJ
Sealed: true
Key Shares: 5
Key Threshold: 3
Unseal Progress: 2
Unseal Nonce: 4ba54ac9-d60b-4078-05da-285ef05bf303
 ~ vault unseal XuASk26YBqQo9+cvi17Me5o7PWsQwVlL2hM8G1bjOyIs
Sealed: false
Key Shares: 5
Key Threshold: 3
Unseal Progress: 0
Unseal Nonce:

 ~ vault status
Sealed: false
Key Shares: 5
Key Threshold: 3
Unseal Progress: 0
Unseal Nonce:
Version: 0.7.2
Cluster Name: test-cluster
Cluster ID: 8f948916-8a3d-b220-f0b4-7c5bb74dc5a6

High-Availability Enabled: false
```

1.2.2 Intialize certificate authority

After initializing and unsealing the Vault, you need to create the certificate authority.

One of the script installed by **vault_ca**, `create-vault-ca`, will do than for you.

Default parameters:

- The script creates a certificate authority valid for 5 years.
- Certificates have a maximum TTL of 5 years and a default one of 30 days.
- Authorization tokens have a maximum TTL of 5 years and a default one of 1 year.

All the parameters above can be configured through command line arguments. See `create-vault-ca --help` for details.

NOTE: set `--vault-token` option using `Initial Root Token` from above.

```
 ~ create-vault-ca --domain=test.org --component=test --vault-addr="http://127.0.0.
↪1:8200" --vault-token=087efd94-3680-0ff4-25b7-a92177cd3c46
Are you sure this script have not been already run on this vault? It can break /␣
↪override configs [y|N] y
Successfully mounted 'pki' at 'pki/test.org'!
Successfully tuned mount 'pki/test.org'!
Key             Value
---             -----
certificate     -----BEGIN CERTIFICATE-----
MIIC9jCCAd6gAwIBAgIUKybrEs7kUvRSgepQuImxonoznCwwDQYJKoZIhvcNAQEL
BQAwEzERMA8GA1UEAxMIdGVzdC5vcmcwHhcNMTcwNTE0MTcyNDMwWhcNMjIwNTEz
MTcyNTAwWjATMREwDwYDVQQDEwh0ZXN0Lm9yZzCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBALeVV8zUKwhYJz0eVg+6rWfCPz+GdxxIXeiChULAU+zHWvDf
Jxye9JrcTdc/XUI0ZSw33F2JEjLkDasdchfL4ESRbUTdnJj1kYW6KEF9X3rhL/AM
hdX+EqUQ9yvXRlvcSyGObVD7ayRUcG2IDpCLRuFW5bkw+MxvSjyzIf6+W3bs5DVz
mFKqRv5Y3ycsuzc8CiDjxj/1LZWvBfqUFf8jePO0bzL3kw7uViZA4fJ23wPLqTyq
IRX52ODZFC3SeyF600lerCLGY4Bgol8YtZwjsx+MxpPnszlkitxT2wjAghfPTV0W
8BebQi4D+CN4A4C6joyGZrdagzsUF3LoeGoDFh8CAwEAAaNCMEAwDgYDVR0PAQH/
BAQDAgEGMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFHovOm1ZHkAsmCGKn+u3
QmlCrriZMA0GCSqGSIb3DQEBCwUAA4IBAQBgDE//Tktbe16VSSrqP8MNYGtMG+jZ
PV2Ao7FCrgSPwjBQHPXR1fh+g4MMG4S9iI8QtXIz49/ZYXfXPU6LPq8W/zrIlf/g
7PODdOo/w7LA7CBHG6ceQtRXHuaMJvJ8EybTQ4vc7LK2FMdEZbBQQkQfnCunR8bz
oACVTooX2DkSPHCM24XSBsmMsHxImEYrjzsr0RyU+R9Tq+rdhjoEyUzQxklIecTq
8D1yfIrgIfYfT1qf6n2bEb+xIfk47v8yXlIUS3KLDadUtqybHIzsbSKEwiQse7rF
AOoUPGoZMSJAr52y0SW2QE8mJoGyX0HeqeX2ocrKw3WvwXF1oHpOB6Au
-----END CERTIFICATE-----
expiration      1652462700
issuing_ca      -----BEGIN CERTIFICATE-----
MIIC9jCCAd6gAwIBAgIUKybrEs7kUvRSgepQuImxonoznCwwDQYJKoZIhvcNAQEL
BQAwEzERMA8GA1UEAxMIdGVzdC5vcmcwHhcNMTcwNTE0MTcyNDMwWhcNMjIwNTEz
MTcyNTAwWjATMREwDwYDVQQDEwh0ZXN0Lm9yZzCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBALeVV8zUKwhYJz0eVg+6rWfCPz+GdxxIXeiChULAU+zHWvDf
Jxye9JrcTdc/XUI0ZSw33F2JEjLkDasdchfL4ESRbUTdnJj1kYW6KEF9X3rhL/AM
hdX+EqUQ9yvXRlvcSyGObVD7ayRUcG2IDpCLRuFW5bkw+MxvSjyzIf6+W3bs5DVz
mFKqRv5Y3ycsuzc8CiDjxj/1LZWvBfqUFf8jePO0bzL3kw7uViZA4fJ23wPLqTyq
IRX52ODZFC3SeyF600lerCLGY4Bgol8YtZwjsx+MxpPnszlkitxT2wjAghfPTV0W
8BebQi4D+CN4A4C6joyGZrdagzsUF3LoeGoDFh8CAwEAAaNCMEAwDgYDVR0PAQH/
BAQDAgEGMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFHovOm1ZHkAsmCGKn+u3
QmlCrriZMA0GCSqGSIb3DQEBCwUAA4IBAQBgDE//Tktbe16VSSrqP8MNYGtMG+jZ
PV2Ao7FCrgSPwjBQHPXR1fh+g4MMG4S9iI8QtXIz49/ZYXfXPU6LPq8W/zrIlf/g
7PODdOo/w7LA7CBHG6ceQtRXHuaMJvJ8EybTQ4vc7LK2FMdEZbBQQkQfnCunR8bz
oACVTooX2DkSPHCM24XSBsmMsHxImEYrjzsr0RyU+R9Tq+rdhjoEyUzQxklIecTq
8D1yfIrgIfYfT1qf6n2bEb+xIfk47v8yXlIUS3KLDadUtqybHIzsbSKEwiQse7rF
AOoUPGoZMSJAr52y0SW2QE8mJoGyX0HeqeX2ocrKw3WvwXF1oHpOB6Au
-----END CERTIFICATE-----
serial_number   2b:26:eb:12:ce:e4:52:f4:52:81:ea:50:b8:89:b1:a2:7a:33:9c:2c

Success! Data written to: pki/test.org/roles/cert
Policy 'pki/test.org/cert' written.
Success! Data written to: auth/token/roles/services
Success! Data written to: auth/token/roles/users

Generating services token (REMEMBER TO SAVE IT)
Key             Value
---             -----
token           9a6b0ee1-c159-3710-dc3f-7641a5ef9222
token_accessor  3730bbec-d8ae-eac0-cc26-c4fb6efa3e0a
```

(continues on next page)

(continued from previous page)

```
token_duration  8760h0m0s
token_renewable true
token_policies  [default pki/test.org/cert]

Have you saved the services token above? [y|N] y

Generating users token (REMEMBER TO SAVE IT)
Key             Value
---             -----
token           3f2ebc35-1793-dc9e-f8da-f6dd6d081ca2
token_accessor  7fcf4113-4918-54a1-d24c-8533b3b10e53
token_duration  8760h0m0s
token_renewable true
token_policies  [default pki/test.org/cert]

Have you saved the users token above? [y|N] y

To boostrap the CA use "fetch-ssl-cert -c test -n <common_name> -t <services_token> -
↪b -o <output_dir>"
To create / renew a certificate use "fetch-ssl-cert -c test -n <common_name> -t
↪<services_token> -o <output_dir>"
```

From now on, you can use one of the two tokens created during the CA setup:

- services: used by automated services to fetch certificate/key pairs.
- users: used by humans to fetch certificate/key pairs.

1.2.3 Secure Vault with TLS

Once the CA is setup, Vault itself need to get its own certificate/key pair and the CA need to be downloaded and bootstrapped into the system.

I am assuming you are on Debian and you have `update-ca-certificates` available.

NOTE: from now on the token used is `services` from above.

```
 fetch-ssl-cert -c test -n vault.test.org -d test.org -i 127.0.0.1 -t 9a6b0ee1-c159-
↪3710-dc3f-7641a5ef9222 -b -A http://127.0.0.1:8200 -D
2017-05-14 18:52:58,443 __init__.py:44 DEBUG:vault address is `http://127.0.0.1:8200`
2017-05-14 18:52:58,443 __init__.py:174 DEBUG:request url is `http://127.0.0.1:8200/
↪v1/pki/test.org/issue/cert`
2017-05-14 18:52:58,443 __init__.py:131 DEBUG:requesting new cert / key part for CA␣
↪domain: `test.org`, component: `test`, common_name: `vault.test.org`, ip_sans: `127.
↪0.0.1`, alt_names: `None`, ttl: `8760h`
2017-05-14 18:52:58,450 connectionpool.py:207 DEBUG:Starting new HTTP connection (1):␣
↪127.0.0.1
2017-05-14 18:52:58,590 connectionpool.py:395 DEBUG:http://127.0.0.1:8200 "PUT /v1/
↪pki/test.org/issue/cert HTTP/1.1" 200 None
2017-05-14 18:52:58,591 __init__.py:59 DEBUG:directory `/usr/local/share/ca-
↪certificates/test.org` already exists, skipping creation
2017-05-14 18:52:58,591 __init__.py:106 DEBUG:writing certificate for vault.test.org␣
↪on /usr/local/share/ca-certificates/test.org/test-vault.test.org.pem
2017-05-14 18:52:58,591 __init__.py:109 DEBUG:writing private key for vault.test.org␣
↪on /usr/local/share/ca-certificates/test.org/test-vault.test.org.key
2017-05-14 18:52:58,591 __init__.py:114 DEBUG:writing CA on /usr/local/share/ca-
↪certificates/test.org/test.crt
```

Now that the certificate/key pair and the CA are saved on the disk, we need to update the system certificate authorities

```
 update-ca-certificates --fresh
Updating certificates in /etc/ssl/certs... 1 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d....done.
```

Stop vault and restart it using the other configuration from `example` directory and repeat the unseal process.

```
 vault server -config example/vault.hcl.ssl
```

Now vault is fully secure and usable.

NOTE: remember to register a local DNS for `vault.test.org` or use the command line option to specify the Vault address.

1.2.4 Fetch certificate/key pair

Let's fetch a certificate/key pair for a test domain.

```
fetch-ssl-cert -c test -n test.test.org -d test.org -t 9a6b0ee1-c159-3710-dc3f-
↪7641a5ef9222 -A https://127.0.0.1:8200 -D
2017-05-14 18:52:58,443 __init__.py:44 DEBUG:vault address is `https://127.0.0.1:8200`
2017-05-14 18:52:58,443 __init__.py:174 DEBUG:request url is `https://127.0.0.1:8200/
↪v1/pki/test.org/issue/cert`
2017-05-14 18:52:58,443 __init__.py:131 DEBUG:requesting new cert / key part for CA␣
↪domain: `test.org`, component: `test`, common_name: `test.test.org`, ip_sans:␣
↪`None`, alt_names: `None`, ttl: `8760h`
2017-05-14 18:52:58,450 connectionpool.py:207 DEBUG:Starting new HTTPS connection␣
↪(1): 127.0.0.1
2017-05-14 18:52:58,590 connectionpool.py:395 DEBUG:https://127.0.0.1:8200 "PUT /v1/
↪pki/test.org/issue/cert HTTP/1.1" 200 None
2017-05-14 18:52:58,591 __init__.py:59 DEBUG:directory `/usr/local/share/ca-
↪certificates/test.org` already exists, skipping creation
2017-05-14 18:52:58,591 __init__.py:106 DEBUG:writing certificate for test.test.org␣
↪on /usr/local/share/ca-certificates/test.org/test-test.test.org.pem
2017-05-14 18:52:58,591 __init__.py:109 DEBUG:writing private key for vault.test.org␣
↪on /usr/local/share/ca-certificates/test.org/test-test.test.org.key
```

CHAPTER 2

API documentation

This is the vault-ca API documentation. It contains the documentation extracted from the docstrings of the various classes, methods, and functions in the vault-ca package. If you want to know what a certain function/method does, this is the place to look.

Contents

2.1 `vault_ca` Module

‘Source <https://github.com/crisidev/vault-ca/blob/master/vault_ca/__init__.py>

class vault_ca.**VaultCA**(*kwargs*)

Object to handle fetching ot certificate/key pairs and CA.

fetch(*common_name*, *ip_sans=None*, *alt_names=None*, *ttl=None*)

Fetch new certificate / key pair from Vault.

If attribute *self.bootstrap_ca* is set to True, also the CA is fetched.

Fetched object are written on disk.

Parameters

- **common_name** (*str*) – common name for the certificate / key pair
- **ip_sans** (*str*) – list of IP for the current certificate, comma separated
- **alt_names** (*str*) – list of alternative names for the current certificate, comma separated
- **ttl** – TTL for the certificate / key pair

Raises ***VaultCAError*** – if the request return errors

exception vault_ca.**VaultCAError**
VaultCA custom exception

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index

F

V