HashiCorp Vault

Overview

Vault provides encryption as a service with centralized key management to simplify encrypting data in transit and at rest.

Security Considerations

- TLS between cloud service and vault
- master token used for authentication
- Vault initially sealed

Work Flows

Incorporating HashiCorp Vault into Keystone will require the implementation of the following work flows:

Encrypt/Decrypt

Vault supports encryption as a service using it's transit engine. The vault manages the encryption keys. This service does not store the encrypted data.

Key Rotation

Key rotation can be performed by interacting with the vault via CLI/API calls. Once a key has been rotated, it is the responsibility of the Keystone application to re-encrypt with the new key.

Encrypting with the later key can performed by:

- re-encrypting when the data is used
- background task to re-encrypt when key changes detected

To facilitate re-encryption, vault offers a "**rewrap**" api which takes the current ciphertext and returns new ciphertext generated by the latest key.

Other Considerations

Sealing/Unsealing the Vault

When a Vault server is started, it starts in a *sealed* state. In this state, Vault is configured to know where and how to access the physical storage, but doesn't know how to decrypt any of it.

Unsealing is the process of constructing the master key necessary to read the decryption key to decrypt the data, allowing access to the Vault.

Prior to unsealing, almost no operations are possible with Vault. For example authentication, managing the mount tables, etc. are all not possible. The only possible operations are to unseal the Vault and check the status of the unseal.

Encryption as a Service Walkthrough

Start server

> *vault server -dev*

```
    ==> Vault server configuration:

             Api Address: http://127.0.0.1:8200
                     Cgo: disabled
          Cluster Address: https://127.0.0.1:8201
               Listener 1: tcp (addr: "127.0.0.1:8200", cluster address: "127.0.0.1:8201", max_request_duration:
"1m30s", max_request_size: "33554432", tls: "disabled")
                Log Level: info
                    Mlock: supported: false, enabled: false
                  Storage: inmem
                  Version: Vault v1.1.3
              Version Sha: 9bc820f700f83a7c4bcab54c5323735a581b34eb

WARNING! dev mode is enabled! In this mode, Vault runs entirely in-memory
and starts unsealed with a single unseal key. The root token is already
authenticated to the CLI, so you can immediately begin using Vault.

You may need to set the following environment variable:

    $ export VAULT_ADDR='http://127.0.0.1:8200'

The unseal key and root token are displayed below in case you want to
seal/unseal the Vault or re-authenticate.

Unseal Key: r1lUG12+XaOYe1F8cQCbLpD7afuuj48Gwvoq1qD6b38=
Root Token: s.Jxg2qInZ9vxTbRB3hhN7DUAX

Development mode should NOT be used in production installations!

==> Vault server started! Log data will stream in below:
```

Create our encryption as a service

Navigate to http://127.0.0.1:8200 and login using the root token.

blocked URL

Enable a transit secret by clicking Enable new engine.

blocked URL

Enable it

blocked URL

blocked URL

Now, create an encryption Key

blocked URL

blocked URL

Let's Use our encryption service

We can use our encryption service via the web ui, cli and rest calls.

Using the Web UI, select **Key actions.**

blocked URL

From this this screen, we can encrypt/decrypt/rewrap. All text to be encrypt must be base64 encoded.

Using the CLI, we would first export our vault address and token:

> *export VAULT_ADDR='http://127.0.0.1:8200'*
>
> *export VAULT_DEV_ROOT_TOKEN_ID="s.EPqGpjt3EpU2siBkmzR47nm0"*

Encrypt

> *vault write transit/encrypt/keystone plaintext=1234*

Decrypt

> *vault write transit/decrypt/keystone ciphertext="vault:v3:7DO5ZmJUHUP8Y1hYaO0rqt44/Vpt28A4yPmXHVUdCg=="*

Rewrap

> *vault write transit/rewrap/keystone ciphertext="vault:v3:7DO5ZmJUHUP8Y1hYaO0rqt44/Vpt28A4yPmXHVUdCg=="*

Deployment

Docker-compose

```
    version: '3.6'
services:
  vault:
    image: vault:1.1.3
    ports:
    - '8200:8200'
    restart: always
    networks:
      keystone-net:
    cap_add:
      - IPC_LOCK
    entrypoint: vault server -config=/vault/config/vault.json
    volumes:
      - vault-data:/vault/file:rw
      - vault-config:/vault/config:rw

volumes:
  vault-data:
    driver: local
    driver_opts:
      type: 'none'
      o: 'bind'
      device: '$PWD/vault/data'
  vault-config:
    driver: local
    driver_opts:
      type: 'none'
      o: 'bind'
      device: '$PWD/vault/config'

networks:
  keystone-net:
```

Configuration File

```
    {
   "backend":{
      "file":{
         "path":"/vault/file"
      }
   },
   "listener":{
      "tcp":{
         "address":"0.0.0.0:8200",
         "tls_disable":1
      }
   },
   "default_lease_ttl":"168h",
   "max_lease_ttl":"0h",
   "ui":"true"
}
```

Initialize the Vault

When initializing the vault, we need to define the number of unseal keys and the number of these keys required to unseal the vault.

> *vi payload.json*

```
   {
  "secret_shares": 10,
  "secret_threshold": 5
}
```

> *curl --request PUT --data @payload.json http://127.0.0.1:8200/v1/sys/init*


```
   {"keys":["fd434d308efa04328277fdc44a2129f69d8b89fa7ca223cf6ef666e4bcd3b0c182","
1ed287fc3cfa6a7492b3da0a9819a0bc09a7fe91a87fc910b17eb9aac70f757709","
896737781c4957c70b4f543e6098109b4c20037e417983150afe48fe91382f85b2","
c5824dadf99006fa67eccc4c34758e02faeb607c98d8fe4f1fb89d6fb44776fd05","
fe4ead84d70258c315bcfb5be5a86d58007892eb9db81c51bebdd706f1e1481781","
2f95999d2f6e44783e7053fef4649cf5dd861c3ec35651d062d14a02c2f89db361","
40b4036820c02d9a26657e5ea35b7c38362d1341dbd0eb06bec9b1df2dd2e203fb","
d58fc88cf1c39ecc98dcef8ca52c428cdd19a31e711b6aeb493bac21764c3f7e64","
9e9f65192b582987f3b68ce1268f9059f9d098d189328cd1f8994e8488f509ffbd","
4995122e2e9f5b6cd1d26efdd3047e5183adc384811ba647174346fd14bbe43b41"],
"keys_base64":
["/UNNMI76BDKCd/3ESiEp9p2Lifp8oiPPbvZm5LzTsMGC","HtKH/Dz6anSSs9oKmBmgvAmn/pGof8kQsX65qscPdXcJ","
iWc3eBxJV8cLT1Q+YJgQm0wgA35BeYMVCv5I/pE4L4Wy","xYJNrfmQBvpn7MxMNHWOAvrrYHyY2P5PH7idb7RHdv0F","
/k6thNcCWMMVvPtb5ahtWAB4kuuduBxRvr3XBvHhSBeB","L5WZnS9uRHg+cFP+9GSc9d2GHD7DVlHQYtFKAsL4nbNh","
QLQDaCDALZomZX5eo1t8ODYtE0Hb0OsGvsmx3y3S4gP7","1Y/IjPHDnsyY3O+MpSxCjN0Zox5xG2rrSTusIXZMP35k","
np9lGStYKYfztozhJo+QWfnQmNGJMozR+JlOhIj1Cf+9","SZUSLi6fW2zR0m790wR+UYOtw4SBG6ZHF0NG/RS75DtB"],
"root_token":"s.Fm8mxDMIcUHXUPyIbTypwxpw"}
```


Get status of Vault

> *curl http://127.0.0.1:8200/v1/sys/init*

```
   {"initialized":true}
```

Unseal the vault

By default, the vault is sealed at startup. In order to unseal the vault, you need to enter a number of keys defined by the secret_threshold.

Vault CLI

Install Client Binary

Navigate to: https://www.vaultproject.io/downloads.html

Using the CLI Tool

To start a development server, issue the following command

> *vault server -dev*

From another terminal:

Export the vault_address and root token

> *export VAULT_ADDR='http://127.0.0.1:8200'*
>
> *export VAULT_DEV_ROOT_TOKEN_ID="s.U9tziXMGQHIT1MHzXC9qMnbj"*

Commands:

Client Command	Description	Sample Output
vault status	Status of vault	Key Value --- ----- Seal Type shamir Initialized true Sealed false Total Shares 1 Threshold 1 Version 1.1.3 Cluster Name vault-cluster-f241dd27 Cluster ID 448a5c8a-6685-b66c-ab99-9664618e9006 HA Enabled false

`vault operator init`	Initialize the vault	`Unseal Key 1: dW2PXpPdjWZvXCUvE/GWxJ+CdeEp6SziEKh6xNYRpB8k`
		`Unseal Key 2: 5K52IOOU+rZf+6Aj7PBOTclnL80Ftb1Wta1GbrJDWX8f`
		`Unseal Key 3: ykK/Q5Il7OOp/qKTdT75U1q6EDzMo2LkM0KRWv7I11Lb`
		`Unseal Key 4: /1EVEn1UDG4LbqI2h5MQPWRI1wpCbirELJyVBo+D2QR1`
		`Unseal Key 5: H47Vch2d0AxuA43kxOlW+MzC/YtjoGU8wCoZLDmRg29r`
		
		`Initial Root Token: s.1ee2zxWvX43sAwjlcDaSGGSC`
		`...`

`vault operator unseal`	Unseal the vault. Must be repeated until the threashold has been reached.	`Unseal Key (will be hidden):`
		`Key Value`
		`--- -----`
		`...`
		`Sealed true`
		`Unseal Progress 1/3`

vault write transit/encrypt/keystone plaintext=1234	Encrypt	ciphertext
vault write transit/decrypt/keystone ciphertext="vault:v3:7DO5ZmJUHUP8Y1hYaO0rqt44/Vpt28A4yPmXHVUdCg=="	Decrypt	plaintext
vault write transit/rewrap/keystone ciphertext="vault:v3:7DO5ZmJUHUP8Y1hYaO0rqt44/Vpt28A4yPmXHVUdCg=="	Rewrap	ciphertext

References

Reference	URL

Encryption as a Service	https://learn.hashicorp.com/vault/encryption-as-a-service/eaas-transit
Vault CLI Reference	https://www.vaultproject.io/docs/commands/
Transit Secret Engine	https://www.vaultproject.io/docs/secrets/transit/index.html
Auto-Unseal Vault	https://github.com/tolitius/cault
Vault and Secret Management in Kubernetes	https://www.hashicorp.com/resources/vault-secret-management-kubernetes