
ATTACK
DEFENSE
by PentesterAcademy

Name	Hashicorp Vault: Basics
URL	https://www.attackdefense.com/challengedetails?cid
Type	DevSecOps Basics: Secrets Management

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Challenge Description

Hashicorp Vault allows the user to securely store the secrets (e.g. tokens, passwords, certificates, encryption keys). The user or applications can interact with it using web UI, CLI, or HTTP API.

In this lab, a Vault server and a Kali machine are provided. The Vault CLI utility and curl are installed on the Kali machine.

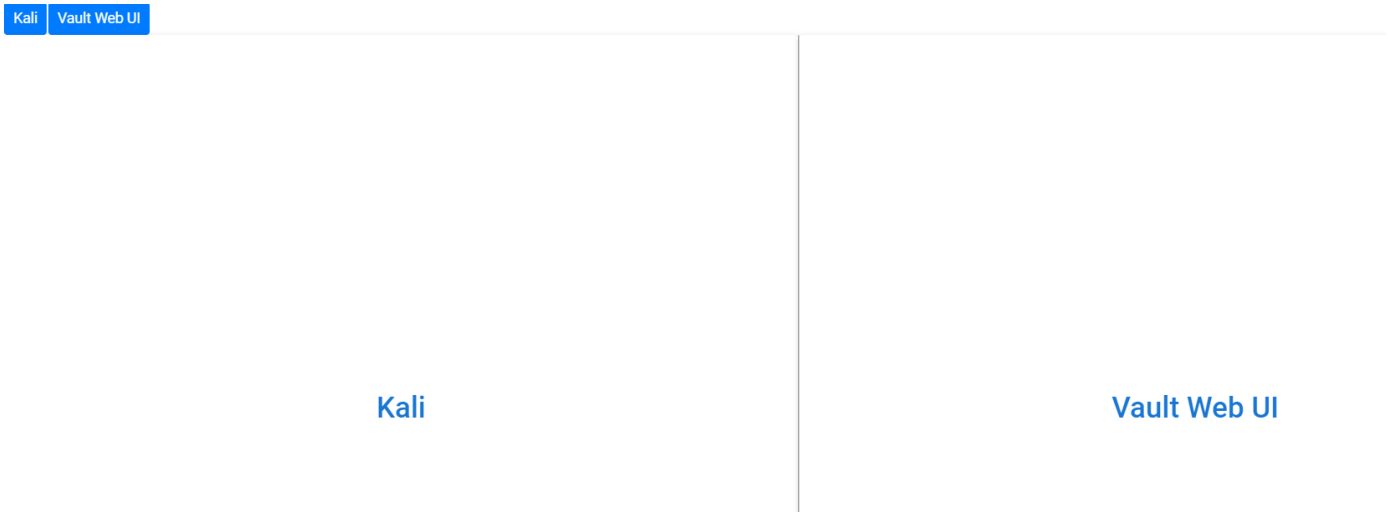
Vault Server Information

URL	http://vault:8200
Token	welcome123

Objective: Follow the manual to learn how to use vault to store/retrieve secrets, encrypt/descrypt text strings using curl and CLI utility!

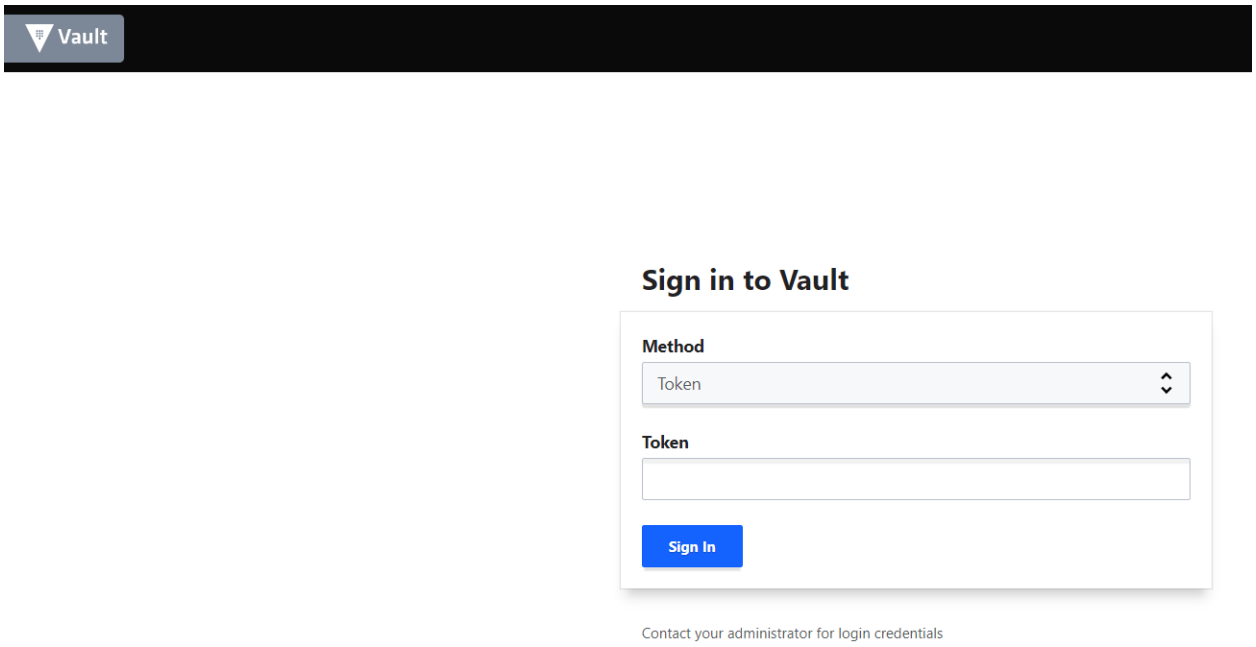
Lab Setup

On starting the lab, the following interface will be accessible to the user.



On choosing (clicking the text in the center) left left panel, **Kali CLI** will open in a new tab

On selecting the right panel, a web UI of **Vault** will open in a new tab.



Solution

Step 1: On the Kali machine, export the following environment variables.

Commands:
export VAULT_ADDR=http://vault:8200
export VAULT_TOKEN=welcome123

```
root@kali:~# export VAULT_ADDR=http://vault:8200
root@kali:~# export VAULT_TOKEN=welcome123
root@kali:~#
```

The mapping for vault is present in /etc/hosts file.

Command: cat /etc/hosts

```
root@kali:~# cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.208.200.3   kali
192.208.200.2 HelloWorld
192.208.200.3 kali
192.208.200.4 vault
```

Step 2: Check the status of the Vault server by using the vault command.

Commands: vault status

```
root@kali:~# vault status
Key             Value
---             -----
Seal Type       shamir
Initialized     true
Sealed          false
Total Shares    1
Threshold       1
Version         1.7.2
Storage Type    inmem
Cluster Name    vault-cluster-1e928747
Cluster ID      ce5fecf8-70ed-0572-48ad-54dc486204ce
HA Enabled      false
```

Secret storage and Retrieval

Task I: Creating and storing a secret

Command: vault kv put secret/test key1=secret-stash

```
root@kali:~# vault kv put secret/test key1=secret-stash
Key              Value
---              -----
created_time     2021-05-22T09:55:54.54089542Z
deletion_time    n/a
destroyed        false
version          1
```

Task II: Retrieve a secret

Command: vault kv get secret/test

```
root@kali:~# vault kv get secret/test
====== Metadata ======
Key              Value
---              -----
created_time     2021-05-22T09:55:54.54089542Z
deletion_time    n/a
destroyed        false
version          1

==== Data ====
Key     Value
---     -----
key1    secret-stash
```

Task III: Delete a secret

Command: vault kv delete secret/test

Verify if it delete

Command: vault kv get secret/test

```
root@kali:~# vault kv delete secret/test
Success! Data deleted (if it existed) at: secret/test
root@kali:~#
root@kali:~# vault kv get secret/test
====== Metadata ======
Key              Value
---              -----
created_time     2021-05-22T09:55:54.54089542Z
deletion_time    2021-05-22T09:59:29.59401166Z
destroyed        false
version          1
```

The secret is deleted.

Encryption as a Service

Enable transit engine

Command: vault secrets enable transit

```
root@kali:~#  vault secrets enable transit
Success! Enabled the transit secrets engine at: transit/
root@kali:~#
```

Task I: Creating a key

Command: vault write -f transit/keys/new-key

```
root@kali:~# vault write -f transit/keys/new-key
Success! Data written to: transit/keys/new-key
root@kali:~#
```

Task II: Encrypt a string with this key

Command: vault write transit/encrypt/new-key plaintext=$(base64 <<< "This is secret")

```
root@kali:~# vault write transit/encrypt/new-key plaintext=$(base64 <<< "This is secret")
Key            Value
---            -----
ciphertext     vault:v1:5u896VvUHHC8wGF4nCKaGc9H+lfWNYeRekb/twHVlej0nCJEm+3VYxinDQ==
key_version    1
```

Task III: Decrypt the ciphertext with the same key

Please remember to use the ciphertext returned in the previous step

Command: vault write transit/decrypt/new-key ciphertext="vault:v1:cZNHVx+sxdMErXRSuDa1q/pz49fXTn1PScKfhf+PIZPvy8xKfkytpwKcbC0f F2U="

And then decode the base64 encoding

Command: echo VGhpcyBpcyBzZWNyZXQK | base64 -d

```
root@kali:~# vault write transit/decrypt/new-key ciphertext="vault:v1:5u896VvUHHC8wGF4nCKaGc9H+lfWNYeRekb/twHVlej0nCJEm+3VYxinDQ=="
Key          Value
---          -----
plaintext    VGhpcyBpcyBzZWNyZXQK
root@kali:~#
root@kali:~#
root@kali:~# echo VGhpcyBpcyBzZWNyZXQK | base64 -d
This is secret
root@kali:~#
```

Task IV: Read key information

Command: vault read transit/keys/new-key

```
root@kali:~# vault read transit/keys/new-key
Key                       Value
---                       -----
allow_plaintext_backup    false
deletion_allowed          false
derived                   false
exportable                false
keys                      map[1:1621676968]
latest_version            1
min_available_version     0
min_decryption_version    1
min_encryption_version    0
name                      new-key
supports_decryption       true
supports_derivation       true
supports_encryption       true
supports_signing          false
type                      aes256-gcm96
```

Vault Web UI and Curl

The corresponding curl commands and web UI steps can be found on the following links:

- Secret Storage and Retrieval (https://learn.hashicorp.com/tutorials/vault/getting-started-apis?in=vault/getting-started)
- Encryption as a Service (https://learn.hashicorp.com/tutorials/vault/eaas-transit)