**Title: Using HVault Secrets in Domino with Client Certificates**

## **1. Introduction**

This document outlines the step-by-step process for retrieving secrets stored in HVault using a client certificate and a private key. The goal is to:

1. Generate an HVault token via TLS authentication (mutual TLS).
2. Access specific secrets in HVault using the generated token.
3. Use these secrets within a Domino project (e.g., for accessing IBM Cloud Object Storage via HMAC keys).

## **2. Prerequisites**

1. **HVault** configured to accept certificate-based authentication (mTLS).
2. **HVault policy** granting the client certificate access to the required secrets.
3. **A Domino project** with a dataset containing:
   * The client certificate (.crt or .pem file).
   * The corresponding private key (.key file).
   * (Optional) The HVault Certificate Authority (CA) file (.pem).
4. **Network access** from the Domino project to the HVault service (firewall rules, traffic permissions, etc.).

## **3. Creating an HVault Policy**

To allow the client certificate to read specific secrets:

1. In HVault, navigate to **Policies**.

Create or update a policy (e.g., cert-policy) with the following:  
  
 path "secret/data/my\_secrets" {

capabilities = ["read"]

}

1. In **Auth Methods**, link your client certificate to this policy by mapping the certificate's Common Name (CN) accordingly.

## **4. Storing the Certificate and Key in a Domino Dataset**

In Domino, create or use an existing dataset to store:

* client.crt (the client certificate)
* client.key (the private key)
* (Optional) hvault\_ca.pem (HVault CA certificate)

Mount this dataset in your Domino project to access the files.

## **5. Generating an HVault Token via TLS (Mutual TLS)**

The process involves:

1. Sending an HTTPS request to HVault with the client certificate and private key.
2. HVault validating the certificate.
3. HVault returning a token that grants access to secrets.

### **Example using cURL**

curl --cert /path/to/client.crt \

--key /path/to/client.key \

--cacert /path/to/hvault\_ca.pem \

https://<hvault-address>:8200/v1/auth/cert/login

The response will contain an auth.client\_token field, which is the token for further API requests.

### **Example using Python**

import requests

HV\_URL = "https://<hvault-address>:8200"

cert = ("/path/to/client.crt", "/path/to/client.key")

ca\_bundle = "/path/to/hvault\_ca.pem"

response = requests.post(f"{HV\_URL}/v1/auth/cert/login", cert=cert, verify=ca\_bundle)

if response.status\_code == 200:

hvault\_token = response.json()["auth"]["client\_token"]

print("HVault Token retrieved:", hvault\_token)

else:

print("Error:", response.text)

## **6. Retrieving Secrets from HVault**

Once the hvault\_token is obtained:

headers = {"X-Vault-Token": hvault\_token}

secret\_path = "secret/data/my\_secrets"

response = requests.get(f"{HV\_URL}/v1/{secret\_path}", headers=headers, verify=ca\_bundle)

if response.status\_code == 200:

data = response.json()

secrets = data["data"]["data"] # Adjust based on HVault version

print("Retrieved secrets:", secrets)

else:

print("Error retrieving secret:", response.text)

For example, the secrets variable may contain:

{

"COS\_HMAC\_ACCESS\_KEY\_ID": "xxx",

"COS\_HMAC\_SECRET\_ACCESS\_KEY": "yyy"

}

These values can then be used for authentication with IBM COS.

## **7. Usage in Domino**

1. **Mount the dataset** containing client.crt, client.key, and optionally hvault\_ca.pem in your Domino project.
2. **Load** these files in your script or notebook (CERT\_FILE, KEY\_FILE, CA\_FILE).
3. **Generate** the HVault token using the auth/cert/login API call.
4. **Query** HVault to fetch required secrets.
5. **Utilize** these secrets in your application (e.g., IBM COS authentication).

## **8. Complete Example in a Domino Notebook**

import requests

# Paths within the mounted dataset in Domino

CERT\_FILE = "/mnt/datasets/my\_cert\_dataset/client.crt"

KEY\_FILE = "/mnt/datasets/my\_cert\_dataset/client.key"

CA\_FILE = "/mnt/datasets/my\_cert\_dataset/hvault\_ca.pem"

HV\_URL = "https://<hvault-address>:8200"

# 1. Generate HVault Token

auth\_url = f"{HV\_URL}/v1/auth/cert/login"

auth\_response = requests.post(auth\_url, cert=(CERT\_FILE, KEY\_FILE), verify=CA\_FILE)

if auth\_response.status\_code != 200:

raise Exception(f"Failed to authenticate with HVault: {auth\_response.text}")

hv\_token = auth\_response.json()["auth"]["client\_token"]

print("HVault Token obtained:", hv\_token)

# 2. Fetch Secret

secret\_path = "secret/data/my\_secrets"

headers = {"X-Vault-Token": hv\_token}

secret\_response = requests.get(f"{HV\_URL}/v1/{secret\_path}", headers=headers, verify=CA\_FILE)

if secret\_response.status\_code != 200:

raise Exception(f"Failed to retrieve secret: {secret\_response.text}")

secret\_data = secret\_response.json()["data"]["data"]

print("Retrieved secret:", secret\_data)

## **9. Security Considerations**

* **Protect keys and certificates**: Restrict access to your dataset in Domino to avoid exposure of sensitive credentials.
* **Certificate rotation**: If the client certificate expires or is renewed, update the files in the dataset accordingly.
* **Policy updates**: If additional secrets are required, ensure the HVault policy allows access to the necessary paths.

## **10. Conclusion**

By configuring HVault for client certificate authentication (mTLS) and securely storing these files in Domino, you can:

1. Generate an HVault token (based on client certificate authentication).
2. Retrieve required secrets securely from HVault.
3. Utilize these secrets directly in your Domino workflows (e.g., IBM COS authentication).

This approach ensures **secure** and **automated** access to secrets while maintaining centralized control via HVault.

**End of Document**