Vault Integration Guide

This guide provides a quick reference for integrating a Crypto4A HSM (QxHSM or QxEdge) with Hashicorp Vault. When properly integrated, Vault stores the Unseal key in the HSM.

Requirements

Hashicorp Vault Enterprise License

A Hashicorp Vault enterprise license is required in order to integrate Hashicorp Vault with an HSM.

Crypto4A PKCS11 Library

The integration of Hashicorp Vault and a Crypto4A HSM requires Crypto4A's pkcs11 library. Crypto4A's pkcs11 library is usually named `libpkcs11.so`.

Integration Instructions

The following outlines how to configure Hashicorp Vault to integrate with a Crypto4A HSM.

1. Vault Configuration

Hashicorp Vault must first be configured to use Crypto4A's pkcs11 library. This can be done with the following configuration environment variables.

```
None
VAULT_HSM_LIB="<pkcs11-library-path>"
VAULT_SEAL_TYPE="pkcs11"
```

```
// Must be configured to use slot '5' for Crypto4A's pkcs11 library
VAULT_HSM_SLOT="5"

VAULT_HSM_PIN="<number>"

VAULT_HSM_KEY_LABEL="<vault-key-label>"
VAULT_HSM_HMAC_KEY_LABEL="<vault-hmac-key-label>"

// If set to true, vault will instruct the HSM to generate the vault key and
// the vault hmac key
VAULT_HSM_GENERATE_KEY=<boolean>
```

Example Configuration

This is an example configuration for integrating Vault with a Crypto4A HSM via pkcs11.

```
None
VAULT_HSM_LIB="/opt/c4a/pkcs11/lib/libpkcs11.so"
VAULT_SEAL_TYPE="pkcs11"
VAULT_HSM_SLOT="5"
VAULT_HSM_PIN="1234"
VAULT_HSM_KEY_LABEL="hashicorp-vault-key"
VAULT_HSM_HMAC_KEY_LABEL="hashicorp-vault-hmac-key"
VAULT_HSM_GENERATE_KEY=true
```

2. PKCS11 Library Configuration

To enable Vault to connect and utilize a Crypto4A HSM, the pkcs11 library must be configured accordingly. The pkcs11 library configuration instructions are described in the PKCS11 Configuration support portal documentation.

For reference, the pkcs11 library can be configured with the following environment variables:

- **C4A_PKCS11_CONFIG**: Controls the configuration file loaded by the PKCS#11 library.
- **C4A_PKCS11_HSM_CLIENT**: Overrides the pkcs11.hsm-client property.
- **C4A_PKCS11_USE_SYNCHRONOUS_INITIALIZATION**: Overrides the pkcs11.use-synchronous-initialization property.

- **C4A_PKCS11_KEYMAN_ADDR**: Overrides the service.spa-keyman-service.addr property.
- **C4A_PKCS11_KEYMAN_PORT**: Overrides the service.spa-keyman-service.port property.
- **C4A_PKCS11_KEYMAN_GRPC_ADDR**: Overrides the service.spa-keyman-grpc-service.addr property.
- **C4A_PKCS11_KEYMAN_GRPC_PORT**: Overrides the service.spa-keyman-grpc-service.port property.
- **C4A_PKCS11_LOG_LEVEL**: Overrides the pkcs11.logLevel property.
- **C4A_PKCS11_LOG_LIMIT**: Overrides the pkcs11.logLimit property.
- **C4A_PKCS11_LOG_FILENAME**: Overrides the pkcs11.logFileName property.
- **C4A_PKCS11_LOG_APPEND_MODE**: Overrides the pkcs11.logAppendMode property.
- **C4A_PKCS11_GENERATED_ID_SIZE**: Overrides the pkcs11.generatedIdSize property.
- **C4A_PKCS11_QUIRKS_MODE**: Overrides the pkcs11.quirksMode property.
- **C4A_PKCS11_RELAX_TEMPLATE_CONSISTENCY**: Overrides the pkcs11.relaxTemplateConsistency property.
- **C4A_PKCS11_RELAX_MULTI_INITIALIZE**: Overrides the pkcs11.relaxMultiInitialize property.
- **C4A_PKCS11_ONLY_FIND_X509_CERTS**: Overrides the pkcs11.onlyFindX509Certs property.
- **C4A_PKCS11_RETURN_NULL_TERMINATED_STRINGS**: Overrides the pkcs11.returnNullTerminatedStrings property.
- **C4A_PKCS11_RETURN_SIGNED_BIG_INTEGERS**: Overrides the pkcs11.returnSignedBigIntegers property.

Example Configuration

The following is an example configuration that would instruct the pkcs11 library to connect to a QxHSM emulator listening on `127.0.0.1`.

```
None
C4A_PKCS11_KEYMAN_ADDR=127.0.0.1
C4A_PKCS11_KEYMAN_PORT=8106
C4A_PKCS11_HSM_CLIENT=REST
```

3. Validate Integration

Proceed with the following steps to validate the integration of Hashicorp Vault with the Crypto4A HSM.

1. Validate that a secret can be created and fetched

1.1. Create a secret using the vault cli

```
None
vault kv put secret/my-secret username="user" password="password
```

1.2. Get the secret that was just created

```
None
vault kv get secret/my-secret
```

2. Validate that the vault keys exists on the HSM

The output of the two following commands should be the key's uuid with some of its metadata.

2.1 Find the vault key

```
None
skm find label <vault-key-label>
```

2.2 Find the vault hmac key

```
None
skm find label <vault-hmac-key-label>
```

External Resources

- Hashicorp Vault PKCS11 Integration Guide: https://developer.hashicorp.com/vault/docs/configuration/seal/pkcs11