# HashiCorp Vault Enterprise

## nShield® HSM Integration Guide

23 Oct 2023

# Contents

# 1. Introduction

HashiCorp Vault Enterprise (referred to as Vault in this guide) supports the creation/storage of keys within Hardware Security Modules (HSMs). Entrust nShield HSMs provide FIPS or Common Criteria certified solutions to securely generate, encrypt, and decrypt the keys which provide the root of trust for the Vault protection mechanism.

This guide describes how to integrate Vault with an nShield HSM to:

- Offload select PKI operations to the HSMs.
- Generate new PKI key pairs and certificates.
- Verify and sign certificate workflows.

## 1.1. Product configurations

Entrust has successfully tested nShield HSM integration with Vault in the following configurations:

| Product | Version |
|---|---|
| HashiCorp Vault Enterprise | v1.15.0 Enterprise HSM |
| Base OS | Red Hat Enterprise 8.4 |

### 1.1.1. Supported nShield features

Entrust has successfully tested nShield HSM integration with the following features:

| Feature | Support |
|---|---|
| Softcards | Yes |
| Module Only Key | Yes |
| OCS cards | Yes |
| nSaaS | Supported but not tested |

### 1.1.2. Supported nShield hardware and software versions

Entrust has successfully tested with the following nShield hardware and software versions:

#### 1.1.2.1. nShield 5c

| Security World Software | Firmware | Netimage | OCS | Softcard | Module |
|---|---|---|---|---|---|
| 13.4.4 | 13.3.2 (FIPS Pending) | 13.3.2 | ✓ | ✓ | ✓ |

#### 1.1.2.2. Connect XC

| Security World Software | Firmware | Netimage | OCS | Softcard | Module |
|---|---|---|---|---|---|
| 13.4.4 | 12.72.1 (FIPS Certified) | 12.80.5 | ✓ | ✓ | ✓ |
| 13.4.4 | 12.60.15 (CC Certified) | 13.3.2 | ✓ | ✓ | ✓ |

### 1.1.3. Supported nShield key types

Entrust has successfully tested with the following Vault managed keys:

- RSA
- ECDSA

## 1.2. Requirements

- A dedicated Linux server is needed for the installation of Vault.
- Network environment setup, via correct firewall configuration with usable ports: 9004 for the HSM and 8200 for Vault.
- HashiCorp Vault Enterprise Modules license, which is required for using Vault with Hardware Security Modules.

Before installing these products, read the associated nShield HSM *Installation Guide*, *User Guide*, and the Vault documentation. This guide assumes familiarity with the following:

- The importance of a correct quorum for the Administrator Card Set (ACS).
- Whether Operator Card Set (OCS) protection or Softcard protection is required.
- If OCS protection is to be used, a 1-of-N quorum must be used.
- Whether your Security World must comply with FIPS 140 Level 3 or Common Criteria standards. If using FIPS 140 Level 3, it is advisable to create an OCS for FIPS authorization. The OCS can also provide key protection for the Vault master key. For

information about limitations on FIPS authorization, see the *Installation Guide* of the nSheld HSM.

> ℹ Entrust recommends that you allow only unprivileged connections unless you are performing administrative tasks.

- Whether to instantiate the Security World as recoverable or not.

## 1.3. More information

For more information about OS support, contact your HashiCorp Vault Enterprise sales representative or Entrust nShield Support, https://nshieldsupport.entrust.com.

# 2. Install and configure the Entrust nShield HSM

Installation and configuration steps:

1. Select the protection method
2. Install the HSM
3. Install the nShield Security World Software and create the Security World
4. Create the OCS or Softcard

## 2.1. Select the protection method

OCS, Softcard, or Module protection can be used to authorize access to the keys protected by the HSM.

- Operator Cards Set (OCS) are smartcards that are presented to the physical smartcard reader of an HSM. For more information on OCS use, properties, and k-of-N values, see the *User Guide* for your HSM.
- Softcards are logical tokens (passphrases) that protect they key and authorize its use.
- Module protection has no passphrase.

Follow your organization’s security policy to select an authorization access method.

## 2.2. Install the HSM

Install the nShield Connect HSM locally, remotely, or remotely via the serial console. See the following nShield Support articles and the *Installation Guide* for the HSM:

- https://nshieldsupport.entrust.com/hc/en-us/articles/360021378272-How-To-Locally-Set-up-a-new-or-replacement-nShield-Connect
- https://nshieldsupport.entrust.com/hc/en-us/articles/360014011798-How-To-Remotely-Setup-a-new-or-replacement-nShield-Connect
- https://nshieldsupport.entrust.com/hc/en-us/articles/360013253417-How-To-Remotely-Setup-a-new-or-replacement-nShield-Connect-XC-Serial-Console-Model

## 2.3. Install the nShield Security World Software and create the Security World

To install the nShield Security World Software and create the Security World:

1. Install the Security World software as described in *Installation Guide* and the *User Guide* for the HSM. This is supplied on the installation disc.
2. Install the TAC-955 hot fix. This hotfix contains an updated version of the PKCS#11 library and utilities.
3. Add the Security World utilities path `/opt/nfast/bin` to the system path.

   ```
   # sudo vi /etc/profile.d/nfast.sh
   ```

   Add the following info to `nfast.sh` and save:

   ```
   # Entrust Security World path variable
   export PATH=$PATH:/opt/nfast/bin
   ```

4. Open the firewall port 9004 for the HSM connections.

   ```
   # sudo firewall-cmd --permanent --add-port=9004/tcp
   # sudo firewall-cmd --reload
   ```

5. Open a command window and run the following to confirm the HSM is `operational`.

   ```
   # enquiry
   Server:
    enquiry reply flags  none
    enquiry reply level  Six
    serial number        7852-268D-3BF9
    mode                 operational
   ...
   Module #1:
    enquiry reply flags  none
    enquiry reply level  Six
    serial number        7852-268D-3BF9
    mode                 operational
    ...
   ```

6. Create your Security World if one does not already exist or copy an existing one. Follow your organization's security policy for this. Create extra ACS cards as spares in case of a card failure or a lost card.

   > ACS cards cannot be duplicated after the Security World is created.

7. Confirm the Security World is `usable`.

```
# nfkminfo
World
 generation  2
 state       0x3737000c Initialised Usable ...
 ...
Module #1
 generation 2
 state      0x2 Usable
 ...
```

## 2.4. Create the OCS or Softcard

The OCS or Softcard and associated passphrase will be used to authorize access to the keys protected by the HSM. Typically, one or the other will be used, but rarely both. Follow your organization’s security policy to select an authorization access method.

### 2.4.1. Create the OCS

1. Ensure file `/opt/nfast/kmdata/config/cardlist` contains the serial number of the card(s) to be presented, or the wildcard "*".
2. Open a command window as an administrator.
3. Run the `createocs` command as described below, entering a passphrase or password at the prompt.

   Follow your organization’s security policy for this for the values K/N, where K=1 as mentioned above. Use the same passphrase for all the OCS cards in the set (one for each person with access privilege, plus the spares). Note that `slot 2`, remote via a Trusted Verification Device (TVD), is used to present the card.

   > After an OCS card set has been created, the cards cannot be duplicated.

   > Vault requires k = 1 whereas N can be up to, but not exceed, 64.

```
# createocs -m1 -s2 -N testOCS -Q 1/1

FIPS 140-2 level 3 auth obtained.

Creating Cardset:
 Module 1: 0 cards of 1 written
 Module 1 slot 0: Admin Card #1
 Module 1 slot 2: empty
 Module 1 slot 3: empty
 Module 1 slot 2: blank cardSteps:

 Module 1 slot 2:- passphrase specified - writing card
Card writing complete.

cardset created; hkltu = a165a26f929841fe9ff2acdf4bb6141c1f1a2eed
```

Add the **-p** (persistent) option to the command above to have authentication after the OCS card has been removed from the HSM front panel slot, or from the TVD. The authentication provided by the OCS as shown in the command line above is non-persistent and only available while the OCS card is inserted in the HSM front panel slot, or the TVD.

4. Verify the OCS was created:

```
# nfkminfo -c
Cardset list - 2 cardsets:  (P)ersistent/(N)ot, (R)emoteable/(L)ocal-only
 Operator logical token hash               k/n timeout  name
 edb3d45a28e5a6b22b033684ce589d9e198272c2  1/5  none-NL testOCS
```

The rocs utility also shows the OCS was created:

```
# rocs
`rocs' key recovery tool
Useful commands: `help', `help intro', `quit'.
rocs> list cardset
No. Name                     Keys (recov) Sharing
  1 testOCS                  0 (0)        1 of 5
rocs> quit
```

## 2.4.2. Create the Softcard

1. Create an /opt/nfast/cknfastrc file if it does not exist to enable Softcard protection. This is an example:

```
# Enable Softcard protection
CKNFAST_LOADSHARING=1

# Enable Module protection
CKNFAST_FAKE_ACCELERATOR_LOGIN=1

# PKCS #11 log level and file location
CKNFAST_DEBUG=10
CKNFAST_DEBUGFILE=/opt/nfast/log/pkcs11.log
```

2. Run the following command, and enter a passphrase or password at the prompt:

```
# ppmk -n testSC

Enter new pass phrase:
Enter new pass phrase again:
New softcard created: HKLTU d9414ed688c6405aab675471d3722f8c70f5d864
```

3. Verify the Softcard was created:

```
# nfkminfo -s
SoftCard summary - 1 softcards:
 Operator logical token hash               name
 925f67e72ea3c354cae4e6797bde3753d24e7744  testSC
```

The `rocs` utility also shows that the OCS and Softcard were created:

```
# rocs
`rocs' key recovery tool
Useful commands: `help', `help intro', `quit'.
rocs> list cards
No. Name                     Keys (recov) Sharing
  1 testOCS                  0 (0)        1 of 5
  2 testSC                   0 (0)        (softcard)
rocs> quit
```

# 3. Create the Vault encryption and HMAC keys

Follow these steps to create the Vault keys with a single HSM:

1. Create the keys using an OCS protection
2. Create the keys using Softcard protection
3. Create the keys using Module protection
4. Find the slot value for each protection method

The Vault encryption and HMAC keys can be protected with an OCS, Softcard, or Module: Key generation with all three protection methods are shown below. Choose those that apply to you.

## 3.1. Create the keys using an OCS protection

1. Create the Vault encryption key `vault_v1_ocs`:

```
# generatekey --generate --batch -m1 -s0 pkcs11 protect=token cardset=testOCS plainname=vault_v1_ocs type=AES
size=256
key generation parameters:
 operation    Operation to perform      generate
 application  Application               pkcs11
 protect      Protected by              token
 slot         Slot to read cards from   2
 recovery     Key recovery              yes
 verify       Verify security of key    yes
 type         Key type                  AES
 size         Key size                  256
 plainname    Key name                  vault_v1_ocs
 nvram        Blob in NVRAM (needs ACS) no

Loading `testOCS':
 Module 1: 0 cards of 1 read
 Module 1 slot 0: `testOCS' #2
 Module 1 slot 2: Admin Card #15
 Module 1 slot 3: empty
 Module 1 slot 4: empty
 Module 1 slot 5: empty
 Module 1 slot 0:- passphrase supplied - reading card
Card reading complete.

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_ucedb3d45a28e5a6b22b033684ce589d9e198272c2-
3ea7edc9ff8a7c2b17401920b12a3a67a3e21dd7
```

2. Create the Vault HMAC key `vault_hmac_v1_ocs`.

```
# generatekey --generate --batch -m1 -s0 pkcs11 protect=token cardset=testOCS plainname=vault_hmac_v1_ocs
type=HMACSHA256 size=256
key generation parameters:
 operation    Operation to perform       generate
 application  Application                pkcs11
 protect      Protected by               token
 slot         Slot to read cards from    2
 recovery     Key recovery               yes
 verify       Verify security of key     yes
 type         Key type                   HMACSHA256
 size         Key size                   256
 plainname    Key name                   vault_hmac_v1_ocs
 nvram        Blob in NVRAM (needs ACS)  no

Loading `testOCS':
 Module 1: 0 cards of 1 read
 Module 1 slot 0: `testOCS' #2
 Module 1 slot 2: Admin Card #15
 Module 1 slot 3: empty
 Module 1 slot 4: empty
 Module 1 slot 5: empty
 Module 1 slot 0:- passphrase supplied - reading card
Card reading complete.

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_ucedb3d45a28e5a6b22b033684ce589d9e198272c2-
5e0252dea777e36934160cbd072bf03cd1e9ba70
```

## 3.2. Create the keys using Softcard protection

1. Create the encryption key vault_v1_sc:

```
# generatekey --generate --batch -m1 pkcs11 protect=softcard softcard=testSC plainname=vault_v1_sc type=AES size=256
key generation parameters:
 operation    Operation to perform       generate
 application  Application                pkcs11
 protect      Protected by               softcard
 softcard     Soft card to protect key   testSC
 recovery     Key recovery               yes
 verify       Verify security of key     yes
 type         Key type                   AES
 size         Key size                   256
 plainname    Key name                   vault_v1_sc
 nvram        Blob in NVRAM (needs ACS)  no
Please enter the pass phrase for softcard `testSC':

Please wait........

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_uc925f67e72ea3c354cae4e6797bde3753d24e7744-
cdd81cd59c7a4a8518cdcc6c2b7beeac4a88c340
```

2. Create the HMAC key vault_hmac_v1_sc:

```
# generatekey --generate --batch -m1 pkcs11 protect=softcard softcard=testSC plainname=vault_hmac_v1_sc
type=HMACSHA256 size=256
key generation parameters:
 operation    Operation to perform       generate
 application  Application                pkcs11
 protect      Protected by               softcard
 softcard     Soft card to protect key   testSC
 recovery     Key recovery               yes
 verify       Verify security of key     yes
 type         Key type                   HMACSHA256
 size         Key size                   256
 plainname    Key name                   vault_hmac_v1_sc
 nvram        Blob in NVRAM (needs ACS)  no
Please enter the pass phrase for softcard `testSC':

Please wait........

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_uc925f67e72ea3c354cae4e6797bde3753d24e7744-
6c07551e82281c8cc6a531d12c934701409d42be
```

## 3.3. Create the keys using Module protection

1. Create the encryption key `vault_v1_m`:

```
#  generatekey --generate --batch -m1 pkcs11 protect=module plainname=vault_v1_m type=AES size=256
key generation parameters:
 operation    Operation to perform       generate
 application  Application                pkcs11
 protect      Protected by               module
 verify       Verify security of key     yes
 type         Key type                   AES
 size         Key size                   256
 plainname    Key name                   vault_v1_m
 nvram        Blob in NVRAM (needs ACS)  no

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_uab23eff123cdbd108ff958fae07b12c1da92762dc
```

2. Create the HMAC key `vault_hmac_v1_m`:

```
# generatekey --generate --batch -m1 pkcs11 protect=module plainname=vault_hmac_v1_m type=HMACSHA256 size=256
key generation parameters:
 operation    Operation to perform       generate
 application  Application                pkcs11
 protect      Protected by               module
 verify       Verify security of key     yes
 type         Key type                   HMACSHA256
 size         Key size                   256
 plainname    Key name                   vault_hmac_v1_m
 nvram        Blob in NVRAM (needs ACS)  no

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_ua47626b663321b99fb7ce1d035bb211a5311abf0f
```

3. Verify the keys created using the `rocs` utility:

```
# rocs
`rocs' key recovery tool
Useful commands: `help', `help intro', `quit'.
rocs> list keys
  No. Name                    App        Protected by
    1 vault_v1_ocs            pkcs11     testOCS
    2 vault_hmac_v1_ocs       pkcs11     testOCS
    3 vault_v1_sc             pkcs11     testSC (testSC)
    4 vault_hmac_v1_sc        pkcs11     testSC (testSC)
    5 vault_v1_m              pkcs11     module
    6 vault_hmac_v1_m         pkcs11     module
rocs> exit
```

4. Verify the keys created using the `nfkminfo` utility.

```
# nfkminfo -l

Keys with module protection:
 key_pkcs11_ua47626b663321b99fb7ce1d035bb211a5311abf0f `vault_hmac_v1_m'
 key_pkcs11_uab23eff123cdbd108ff958fae07b12c1da92762dc `vault_v1_m'

Keys protected by softcards:
 key_pkcs11_uc925f67e72ea3c354cae4e6797bde3753d24e7744-6c07551e82281c8cc6a531d12c934701409d42be `vault_hmac_v1_sc'
 key_pkcs11_uc925f67e72ea3c354cae4e6797bde3753d24e7744-cdd81cd59c7a4a8518cdcc6c2b7beeac4a88c340 `vault_v1_sc'

Keys protected by cardsets:
 key_pkcs11_ucedb3d45a28e5a6b22b033684ce589d9e198272c2-3ea7edc9ff8a7c2b17401920b12a3a67a3e21dd7 `vault_v1_ocs'
 key_pkcs11_ucedb3d45a28e5a6b22b033684ce589d9e198272c2-5e0252dea777e36934160cbd072bf03cd1e9ba70 `vault_hmac_v1_ocs'
```

## 3.4. Verify the PKCS#11 library is available

1. Present the OCS. Use the nfkminfo utility to find the slot number of the OCS.

```
# nfkminfo
World
 generation  2
 state       0x3737000c Initialised Usable ...
...
Module #1 Slot #0 IC 161
 generation    1
 phystype      SmartCard
 slotlistflags 0x180002 SupportsAuthentication DynamicSlot Associated
 state         0x5 Operator
 flags         0x10000
 shareno       2
 shares        LTU(PIN) LTFIPS
 error         OK
Cardset
 name          "testOCS"
 k-out-of-n    1/5
 flags         NotPersistent PINRecoveryForbidden(disabled) !RemoteEnabled
 timeout       none
 card names    "" "" "" "" ""
 hkltu         edb3d45a28e5a6b22b033684ce589d9e198272c2
 gentime       2023-07-20 18:50:48
 ...
```

2. Execute the `ckcheckinst` command to test the library. Enter the slot number above when prompted. Enter the OCS passphrase when prompted.

```
# ckcheckinst
PKCS#11 library interface version 2.40
                            flags 0
                   manufacturerID "nCipher Corp. Ltd               "
               libraryDescription "nCipher PKCS#11 13.4.4-379-58f7e"
           implementation version 13.04
         Loadsharing and Failover enabled

Slot  Status            Label
====  ======            =====
   0  Fixed token       "loadshared accelerator          "
   1  No token present
   2  Operator card     "testOCS                         "
   3  Soft token        "testSC                          "

Select slot number to run library test or 'R'etry or to 'E'xit: 0
Using slot number 0.

Test                      Pass/Failed
----                      -----------

1 Generate RSA key pair   Pass
2 Generate DSA key pair   Pass
3 Encryption/Decryption   Pass
4 Signing/Verification    Pass

Deleting test keys         ok

PKCS#11 library test successful.
```

## 3.5. Find the slot value for each protection method

Each protection method is loaded to a virtual slot. The decimal value of this slot will be needed further down to configure the Vault.

1. Run the `cklist` command. Notice the lines below.

```
# cklist
Listing contents of slot 0
 (token label "loadshared accelerator          ")
...
Skipping slot 1 (not present)
...
Listing contents of slot 2
 (token label "testOCS                         ")
...
Listing contents of slot 3
 (token label "testSC                          ")
```

`loadshared accelerator`

Module protection (slot 0).

`testOCS`

The name given to the OCS created above (slot 2).

`testSC`

  The name given to the Softcard token created above (slot 3).

2. Search file `/opt/nfast/log/pkcs11.log` for **pSlotList**. Notice the hex value for each slot. For example:

```
...
2023-10-13 13:04:19 [28493]: pkcs11: 00000000 <    pSlotList[0] 0x1D622495
2023-10-13 13:04:19 [28493]: pkcs11: 00000000 <    pSlotList[1] 0x1D622496
2023-10-13 13:04:19 [28493]: pkcs11: 00000000 <    pSlotList[2] 0x1D622497
...
```

3. Convert to decimal:

| Protection Method | Slot Number | Value (Hex) | Value (Decimal) |
|---|---|---|---|
| Module | 0 | 0x2D622495 | 761406613 |
| OCS | 1 | 0x2D622496 | 761406614 |
| Softcards | 2 | 0x2D622497 | 761406615 |

Note or save the decimal values.

> Adding or deleting Softcard tokens, or adding or deleting OCS, or adding or deleting Modules keys will change the values above. Redo the step to find the new values if necessary.

# 4. Install Vault

Follow these steps to install and configure Vault with a single HSM:

1. System preparation
2. Create Vault user and group
3. *Install Vault*
4. Install the Vault license
5. Create a configuration file
6. Create and configure Vault directories
7. Enable Vault

## 4.1. System preparation

1. Open the appropriate firewall port for incoming Vault connections:

```
# sudo firewall-cmd --permanent --add-port=8200/tcp
# sudo firewall-cmd --permanent --add-port=8201/tcp
# sudo firewall-cmd --reload
```

2. Install `open-vm-tools`:

```
# sudo yum install open-vm-tools unzip opensc
```

## 4.2. Create Vault user and group

1. Create the Vault group:

```
# sudo groupadd --system vault
```

2. Create the Vault user:

```
# sudo useradd --system --shell /sbin/nologin --gid vault vault
```

3. Add the Vault user to the nShield **nfast** group:

```
# sudo usermod --append --groups nfast vault
```

## 4.3. Install Vault

1. Download the Vault package from HashiCorp at https://releases.hashicorp.com/vault/, ensuring that it is the binary file for Enterprise with HSM support:

```
# cd Downloads

# wget https://releases.hashicorp.com/vault/1.15.0+ent.hsm/vault_1.15.0+ent.hsm_linux_amd64.zip
--2023-10-16 16:28:25--  https://releases.hashicorp.com/vault/1.15.0+ent.hsm/vault_1.15.0+ent.hsm_linux_amd64.zip
Resolving releases.hashicorp.com (releases.hashicorp.com)... 18.160.181.25, 18.160.181.50, 18.160.181.55, ...
Connecting to releases.hashicorp.com (releases.hashicorp.com)|18.160.181.25|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 132555901 (126M) [application/zip]
Saving to: ‘vault_1.15.0+ent.hsm_linux_amd64.zip’

vault_1.15.0+ent.hsm_linux 100%[=====================================>] 126.42M  10.1MB/s    in 12s

2023-10-16 16:28:38 (10.6 MB/s) - ‘vault_1.15.0+ent.hsm_linux_amd64.zip’ saved [132555901/132555901]
```

2. Unzip the binary file and extract it to the working directory on the host machine, for example `/usr/local/bin`. There should only be a single binary file named `vault`.

```
# unzip vault_1.15.0+ent.hsm_linux_amd64.zip -d /usr/local/bin
Archive:  vault_1.15.0+ent.hsm_linux_amd64.zip
replace /usr/local/bin/TermsOfEvaluation.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: /usr/local/bin/TermsOfEvaluation.txt
replace /usr/local/bin/vault? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: /usr/local/bin/vault
replace /usr/local/bin/EULA.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: /usr/local/bin/EULA.txt
```

3. Set Vault permissions:

```
# sudo chmod 755 /usr/local/bin/vault
# sudo setcap cap_ipc_lock=+ep /usr/local/bin/vault
# ls -la /usr/local/bin/vault
-rwxr-xr-x. 1 root root 397524552 Sep 22 17:22 /usr/local/bin/vault
```

4. Add the Vault binary file to the path:

```
# sudo vi /etc/profile.d/vault.sh
```

   Add the following information to `vault.sh` and save it:

```
# HashiCorp Vault Enterprise path variable
export PATH="$PATH:/usr/local/bin"
export VAULT_ADDR=http://127.0.0.1:8200
```

5. Create the Vault data directories:

```
# sudo mkdir --parents /opt/vault/data
# sudo mkdir --parents /opt/vault/logs
# sudo chmod --recursive 750 /opt/vault
# sudo chown --recursive vault:vault /opt/vault
```

6. Reboot the server.

```
# reboot
```

7. Confirm that the binary file is available:

```
# vault version
Vault v1.15.0+ent.hsm (d3729711f875a9dedea802079cd7f0e4b1d6e8d5), built 2023-09-22T21:04:53Z (cgo)
```

## 4.4. Install the Vault license

1. Open a new terminal and create a directory for the Vault license and configuration files:

```
# sudo mkdir /etc/vault
```

2. Three options are given in the Install a HashiCorp Enterprise License page of the online documentation for enabling an enterprise license, as well as a procedure to request a trail license. For this guide, create a file containing the enterprise license key:

```
# cat /etc/vault/license.hclic
02MV4UU43B...
```

## 4.5. Create a configuration file

Set up a `/etc/vault/config.hcl` configuration file to enable Vault to be run as a service. See also *Vault commands*.

An example configuration file for using Vault with OCS protection is shown below. The **pin** is the **passphrase** entered when the OCS was created.

```
# PKCS#11 Seal, Entrust nShield Integration
seal "pkcs11" {
lib = "/opt/nfast/toolkits/pkcs11/libcknfast.so"
slot = "761406615"
pin = "ncipher"
key_label = "vault_v1_ocs"
hmac_key_label = "vault_hmac_v1_ocs"
# Vault is commanding HSM to generate keys if these don't already exists
generate_key = true
}

# Vault listener with TLS disabled
listener "tcp" {
address = "0.0.0.0:8200"
tls_disable = true
}

# Storage
storage "file" {
path = "/opt/vault/data/hsm"
}

ui = true

# License file
license_path = "/etc/vault/license.hclic"

disable_mlock = true
api_addr = "http://127.0.0.1:8200"
cluster_addr = "https://127.0.0.1:8201"

# Managed Key Library
kms_library "pkcs11" {
name = "hsm1" # This can be re-named to anything you like
library = "/opt/nfast/toolkits/pkcs11/libcknfast.so" #PKCS11 Library Location
}
```

In this example:

- The **slot** and **pin** parameters will change according to the protection selected. See section Find the slot value for each protection method.
- The entropy seal mode is set to augmentation. This leverages the HSM for augmenting system entropy via the PKCS#11 protocol.
- The seal wrap is enabled. By enabling seal wrap, Vault wraps your secrets with an extra layer of encryption leveraging the HSM encryption and decryption.
- Notice the path to the license file.

## 4.6. Create and configure Vault directories

1. Create a vault file in `sysconfig`:

   ```
   # sudo touch /etc/sysconfig/vault
   ```

2. Create a service file:

```
# vi /etc/systemd/system/vault.service
```

3. Add the following information to the file:

   If deploying on a server with more than two CPUs, you may increase the value of `Environment=GOMAXPROCS` accordingly.

```
[Unit]
Description="HashiCorp Vault"
Requires=network-online.target
After=network-online.target nc_hardserver.service
ConditionFileNotEmpty=/etc/vault/config.hcl
[Service]
User=vault
Group=vault
EnvironmentFile=/etc/sysconfig/vault
ExecStart=/usr/local/bin/vault server -config=/etc/vault/config.hcl
StandardOutput=/opt/vault/logs/output.log
StandardError=/opt/vault/logs/error.log
ExecReload=/bin/kill --signal -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=5
TimeoutStopSec=30
StartLimitInterval=60
StartLimitBurst=3
AmbientCapabilities=CAP_IPC_LOCK
LimitNOFILE=65536
LimitMEMLOCK=infinity
[Install]
WantedBy=multi-user.target
```

4. If you are setting paths different from the default, you must edit the following lines as well in the configuration file:

```
ConditionFileNotEmpty=/etc/vault/config.hcl
EnvironmentFile=-/etc/sysconfig/vault
ExecStart=/opt/vault/bin/vault server -config=/etc/vault/config.hcl
StandardOutput=/opt/vault/logs/output.log
StandardError=/opt/vault/logs/error.log
```

## 4.7. Enable Vault

1. Set the following environment variable to allow Vault to be accessed from a web browser via the web user interface (web UI). Append the following line to the `/etc/profile.d/vault.sh` file created above, and restart the system:

```
export VAULT_ADDR=http://127.0.0.1:8200
```

2. Enable Vault:

```
# systemctl enable vault.service
```

# 5. Test the integration

## 5.1. Start Vault

The HSM will be accessed as part of starting Vault. Therefore, the OCS or Softcard is needed.

1. Start the Vault in a separate window.

   If the protection method defined in `/etc/vault/config.hcl` is OCS protection, the OCS card created in Create the keys using an OCS protection must be inserted in the HSM slot. Otherwise the Vault will fail to start. The OCS card is not required for the Vault to start if the protection method is Softcard on Module.

   ```
   # vault server -config=/etc/vault/config.hcl

   WARNING: storage configured to use "file" which is not supported for Vault
   Enterprise, must be "raft" or "consul"

   ==> Vault server configuration:

   Administrative Namespace:
                Api Address: http://127.0.0.1:8200
                        Cgo: enabled
            Cluster Address: https://127.0.0.1:8201
      Environment Variables: DBUS_SESSION_BUS_ADDRESS, DISPLAY, GDK_BACKEND, GODEBUG, HISTCONTROL, HISTSIZE, HOME,
   HOSTNAME, LANG, LESSOPEN, LOGNAME, LS_COLORS, MAIL, OLDPWD, PATH, PWD, SELINUX_LEVEL_REQUESTED,
   SELINUX_ROLE_REQUESTED, SELINUX_USE_CURRENT_RANGE, SHELL, SHLVL, SSH_ASKPASS, SSH_AUTH_SOCK, SSH_CLIENT,
   SSH_CONNECTION, SSH_TTY, TERM, USER, VAULT_ADDR, XDG_DATA_DIRS, XDG_RUNTIME_DIR, XDG_SESSION_ID, _
                 Go Version: go1.21.1
                 Listener 1: tcp (addr: "0.0.0.0:8200", cluster address: "0.0.0.0:8201", max_request_duration: "1m30s",
   max_request_size: "33554432", tls: "disabled")
                  Log Level:
                      Mlock: supported: true, enabled: false
              Recovery Mode: false
                    Storage: file
                    Version: Vault v1.15.0+ent.hsm, built 2023-09-22T21:04:53Z
                Version Sha: d3729711f875a9dedea802079cd7f0e4b1d6e8d5

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

2. Initialize the Vault back in the original window.

   The `vault operator init` command returns the Recovery Key(s) and Initial Root Token. Keep a note of these.

```
# vault operator init
Recovery Key 1: PK0s3VaswduJGkng079G3EPDU1vXifZt27tSnnnJ2kdO
Recovery Key 2: R8rNXyj1CA77UKPuV4zf4MvNv4CODN/AhyLraYcikHKx
Recovery Key 3: 0Bw0TVnq7+zbGMjsJyuzWda7HpBVzlRXzp/0JWwIqAF9
Recovery Key 4: A4t1XIhAcvQKfYPaR6aCVgB6mVCu50zDwI03IHnEsxvD
Recovery Key 5: wc/QtGFBPhKDwXGHW2OCKJEm8XJbEwsCHzKnU1p3Tr+b

Initial Root Token: hvs.7QEXwRx230xNd67I44nJjAxZ

Success! Vault is initialized

Recovery key initialized with 5 key shares and a key threshold of 3. Please
securely distribute the key shares printed above.
```

## 5.2. Log in from the command line

Log in to Vault using the Initial Root Token saved above and save the token below.

```
# vault login hvs.7QEXwRx230xNd67I44nJjAxZ
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

Key                  Value
---                  -----
token                hvs.7QEXwRx230xNd67I44nJjAxZ
token_accessor       GeheYAQMr1dL4VjzGzeSbbzH
token_duration       ∞
token_renewable      false
token_policies       ["root"]
identity_policies    []
policies             ["root"]
```

## 5.3. Create Managed Key In Vault

1. Create an RSA-managed key **hsm-key-ocs-rsa** in Vault **VaultKeyOCSRSA**, protected by the OCS **testOCS** in the nShield HSM.

```
# vault write /sys/managed-keys/pkcs11/hsm-key-ocs-rsa library=hsm1 slot=761406615 pin=ncipher
key_label="VaultKeyOCSRSA" allow_generate_key=true allow_store_key=true mechanism=0x0001 key_bits=2048
Success! Data written to: sys/managed-keys/pkcs11/hsm-key-ocs-rsa
```

2. Write to the nShield HSM the new managed key **hsm-key-ocs-rsa**.

```
# vault write -f /sys/managed-keys/pkcs11/hsm-key-ocs-rsa/test/sign
Success! Data written to: sys/managed-keys/pkcs11/hsm-key-ocs-rsa/test/sign
```

3. Create a ECDSA managed key **hsm-key-ocs-ecdsa** in Vault labeled **VaultKeyOCSRSA**, and protected by the OCS **testOCS** in the nShield HSM.

```
# vault write /sys/managed-keys/pkcs11/hsm-key-ocs-ecdsa library=hsm1 slot=761406615 pin=ncipher
key_label="VaultKeyOCSECDSA" allow_generate_key=true allow_store_key=true mechanism=0x1041 curve=P256
Success! Data written to: sys/managed-keys/pkcs11/hsm-key-ocs-ecdsa
```

4. Write to the nShield HSM the new managed key **hsm-key-ocs-ecdsa**.

```
# vault write -f /sys/managed-keys/pkcs11/hsm-key-ocs-ecdsa/test/sign
Success! Data written to: sys/managed-keys/pkcs11/hsm-key-ocs-ecdsa/test/sign
```

5. List all keys created in the nShield HSM. Notice the new keys **VaultKeyOCSRSA** and **VaultKeyOCSECDSA**

```
# nfkminfo -l

Keys with module protection:
 key_pkcs11_ua47626b663321b99fb7ce1d035bb211a5311abf0f `vault_hmac_v1_m'
 key_pkcs11_uab23eff123cdbd108ff958fae07b12c1da92762dc `vault_v1_m'

Keys protected by softcards:
 key_pkcs11_uc925f67e72ea3c354cae4e6797bde3753d24e7744-6c07551e82281c8cc6a531d12c934701409d42be `vault_hmac_v1_sc'
 key_pkcs11_uc925f67e72ea3c354cae4e6797bde3753d24e7744-cdd81cd59c7a4a8518cdcc6c2b7beeac4a88c340 `vault_v1_sc'

Keys protected by cardsets:
 key_pkcs11_ucedb3d45a28e5a6b22b033684ce589d9e198272c2-3ea7edc9ff8a7c2b17401920b12a3a67a3e21dd7 `vault_v1_ocs'
 key_pkcs11_ucedb3d45a28e5a6b22b033684ce589d9e198272c2-5e0252dea777e36934160cbd072bf03cd1e9ba70 `vault_hmac_v1_ocs'
 key_pkcs11_ucedb3d45a28e5a6b22b033684ce589d9e198272c2-625a46d71da150187d78ecc095a3176b00f97296 `VaultKeyOCSRSA'
 key_pkcs11_ucedb3d45a28e5a6b22b033684ce589d9e198272c2-779846e6edc71265bcc561ba3acaa1c64fa68204 `VaultKeyOCSECDSA'
```

6. Enable the PKI secrets engine at the path `pki` and reference a managed key **hsm-key** stored in the HSM.

```
# vault secrets enable -path=pki -allowed-managed-keys=hsm-key pki
Success! Enabled the pki secrets engine at: pki/
```

7. Perform PKI operations as needed. See the PKI Secrets Engine page in the online documentation.

# Appendix A: Troubleshooting

| Error Message | Resolution |
|---|---|
| Vault fails to start. There may not be a log file created if the vault fails to start upon executing `# systemctl start vault.service`. | Execute the following instead to get some debugging information. `# vault server -config=/etc/vault/config.hcl`. |
| Error: **failed to decrypt encrypted stored keys: error initializing session for decryption: error logging in to HSM: pkcs11: 0xE0: CKR_TOKEN_NOT_PRESENT** | Ensure that the Operator card is inserted in the physical slot of the nShield HSM. |

# Appendix B: Vault commands

## B.1. Vault commands

| Task | Command |
|---|---|
| Log into Vault | `# vault login s.InitialRootToken` |
| Check Vault status | `# vault status` |
| Unseal Vault | `# vault operator unseal -address=http://127.0.0.1:8200` |
| Seal Vault | `# vault operator seal` |

## B.2. vault.service commands

| Task | Command |
|---|---|
| Enable Vault Service | `# systemctl enable vault.service` |
| Disable Vault service | `# systemctl disable vault.service` |
| Start Vault service | `# systemclt start vault.service` |
| Stop Vault service | `# systemclt stop vault.service` |
| Restart Vault service | `# systemctl restart vault.service` |