HashiCorp Vault Setup for Certificate and Key Discovery

This document outlines how to install and prepare HashiCorp Vault so the discovery extension can enumerate certificates and keys, how to generate the read only credentials it needs, and how to enter those credentials into the discovery configuration. Steps are given for both the Vault Web UI and the Vault CLI.

The discovery reads material from three Vault secrets engines:

- PKI: issued certificates and CA certificates
- Transit: encryption key details such as type, size, and version
- KV: certificates and keys stored inside key value secrets

Discovery is read only. It never creates, changes, exports, or deletes anything in Vault.

---

0. The Two Ways to Work with Vault

Everything below can be done two ways. Pick whichever you prefer for each step:

Mode	How you access it	Best for
Web UI	A browser pointed at `https://<vault_host>:8200/ui`	Point and click setup, one off changes
CLI	The `vault` command line tool on your computer	Scripting, copy and paste setup, generating credentials

> The address 8200 is the Vault default port. Replace `<vault_host>` with your server hostname or IP. Use `http://` only for local testing. Use `https://` for anything real.

---

1. Prerequisites

- A machine (Linux, macOS, or Windows) to run the Vault server, reachable on the network from wherever discovery runs.
- Administrative access to Vault (a root token, or an admin account), needed to create policies and authentication credentials.
- Optional: the `vault` CLI installed on your own computer if you want to use the command line. Download it from https://developer.hashicorp.com/vault/install.

---

2. Install and Start Vault

If you already have a running Vault with certificates and keys in it, skip to Section 3.

Option A: Quick start for evaluation (dev mode)

Dev mode runs a ready to use Vault in memory with the UI enabled. It is for testing only. Data is lost on restart.

CLI

```
vault server -dev
```

On startup Vault prints two things you must copy:

- `Api Address`, for example `http://127.0.0.1:8200`. This becomes your Vault URL.
- `Root Token`, for example `hvs.XXXXXXXXXXXXXXXX`. This is your initial admin login.

> If discovery runs on a different machine than Vault, replace `127.0.0.1` with the real hostname or IP of the server so it is reachable.

Web UI

Open `http://127.0.0.1:8200/ui` in a browser, choose Token as the method, and paste the Root Token.

Option B: Persistent install (for real use)

1. Download and install the Vault binary from https://developer.hashicorp.com/vault/install.
2. Create a configuration file (storage plus listener) per the HashiCorp documentation, then start the server:

`bash vault server -config=vault-config.hcl` 3. Initialize Vault once to get your unseal keys and the first root token:

`bash vault operator init`

Save the Unseal Keys and Initial Root Token somewhere safe. 4. Unseal Vault (repeat with different keys until it reports unsealed):

`bash vault operator unseal` 5. You can now open the Web UI at `https://<vault_host>:8200/ui` and log in with the root token.

---

3. Point Your Tools at Vault

Web UI

Browse to `https://<vault_host>:8200/ui` and sign in with your token (or admin account).

CLI

Tell the CLI where Vault is, then log in:

```
# Windows PowerShell
$env:VAULT_ADDR = "https://<vault_host>:8200"

# macOS / Linux
export VAULT_ADDR="https://<vault_host>:8200"

# Log in (paste your token when prompted)
vault login
```

> Vault Enterprise only: if your organization uses namespaces, also set the namespace (ask your Vault administrator which one to use). Leave this out for open source Vault. PowerShell: `$env:VAULT_NAMESPACE="admin/team-a"` · Linux/macOS: `export VAULT_NAMESPACE="admin/team-a"`

---

4. Enable the Secrets Engines to Discover

Discovery can only find material in engines that are enabled and hold data. If your PKI, Transit, or KV engines already exist, skip this section. Otherwise, enable the ones you want to scan.

4.1 PKI (certificates)

Web UI

1. Go to Secrets Engines, then Enable new engine.
2. Choose PKI, keep the path `pki`, click Enable Engine.
3. Open the engine, then Configure to create or import a CA, then issue certificates as needed.

CLI

```
vault secrets enable pki
# (example) create a root CA so there is something to discover
vault write pki/root/generate/internal common_name="example.com" ttl=8760h
```

4.2 Transit (encryption keys)

Web UI

1. Secrets Engines, then Enable new engine, then Transit, path `transit`, then Enable Engine.
2. Open the engine, then Create encryption key, then give it a name and type.

CLI

```
vault secrets enable transit
vault write -f transit/keys/my-app-key       # example key
```

4.3 KV (key value secrets)

Web UI

1. Secrets Engines, then Enable new engine, then KV, path `secret`, then Enable Engine.
2. Add a secret. Any field containing a certificate or key in PEM text will be discovered.

CLI

```
vault secrets enable -path=secret kv-v2
vault kv put secret/my-app tls_cert=@server.pem   # example
```

---

5. Create the Read Only Discovery Policy

A policy defines what the discovery credential is allowed to do. This one grants only the read and list access needed, nothing more.

The policy contents (adjust the engine paths `pki`, `transit`, `secret` to match yours; add extra lines for extra mounts such as `pki_int`):

```
# See which secrets engines are enabled
path "sys/mounts" {
  capabilities = ["read"]
}

# PKI: list and read certificates
path "pki/certs"      { capabilities = ["list"] }
path "pki/cert/*"     { capabilities = ["read"] }

# Transit: list and read key details (no key material leaves Vault)
path "transit/keys"   { capabilities = ["list"] }
path "transit/keys/*" { capabilities = ["read"] }

# KV v2 (mount "secret"): walk and read secrets to find embedded certs/keys
path "secret/metadata/*" { capabilities = ["list", "read"] }
path "secret/data/*"     { capabilities = ["read"] }
```

Web UI

1. Go to Policies, then ACL Policies, then Create ACL policy.
2. Name it `cbom_discovery`.
3. Paste the policy text above into the editor.
4. Click Create policy.

CLI

1. Save the text above to a file named `cbom_discovery_policy.hcl`.
2. Upload it:

`bash vault policy write cbom_discovery cbom_discovery_policy.hcl`

---

6. Set Up Authentication and Get the Credentials

Discovery signs in to Vault using one of three methods. AppRole is recommended for automated use. Whichever you choose, it must be attached to the `cbom_discovery` policy from Section 5.

This is where the values you will enter into the discovery configuration come from. Each method produces different credentials.

6.1 AppRole (recommended)

AppRole gives you a Role ID (like a username) and a Secret ID (like a password).

Web UI

1. Go to Access, then Authentication Methods, then Enable new method, then AppRole, then Enable.
2. Open the AppRole method, then Create role.
3. Name it `cbom_discovery`, and under Token policies add `cbom_discovery`. Save.
4. Open the role. The Role ID is shown on the role overview.
5. Use Generate Secret ID on the role to create a Secret ID. Copy it immediately, as it is shown once.

CLI

```
# Enable AppRole (once)
vault auth enable approle

# Create the role bound to the discovery policy
vault write auth/approle/role/cbom_discovery \
    token_policies="cbom_discovery" \
    token_ttl=1h token_max_ttl=4h

# Get the Role ID (stable, safe to reuse)
vault read auth/approle/role/cbom_discovery/role-id

# Generate a Secret ID (treat like a password, copy it now)
vault write -f auth/approle/role/cbom_discovery/secret-id
```

Copy: the Role ID and the Secret ID.

6.2 Token

A token is a single secret string.

Web UI

Tokens are most easily generated with the CLI below. In dev mode, the Root Token printed at startup can be used for a quick test. Do not use root tokens in production.

CLI

```
vault token create -policy="cbom_discovery" -ttl=4h
```

Copy: the token value (begins with `hvs.`).

6.3 Username and Password

Web UI

1. Access, then Authentication Methods, then Enable new method, then Userpass, then Enable.
2. Open the method, then Create user, set a username and password, and add `cbom_discovery` under Token policies. Save.

CLI

```
vault auth enable userpass
vault write auth/userpass/users/cbom_discovery \
    password="<strong-password>" \
    token_policies="cbom_discovery"
```

Copy: the username and password.

---

7. Collect Your Connection Values

Before configuring discovery, gather these:

Value	Where it comes from
Vault URL	The address of your server, for example `https://vault.example.com:8200` (the `Api Address` from Section 2)
Namespace	Vault Enterprise only: from your Vault administrator; leave blank for open source Vault
Authentication method	The method you set up in Section 6: AppRole, Token, or Username and Password
Credentials	AppRole: Role ID plus Secret ID · Token: token · Userpass: username plus password
Engines to scan	Any of PKI, Transit, KV (the ones you enabled in Section 4)

---

8. Enter the Values in the Discovery Configuration

In the CBOM platform, create a new HashiCorp Vault discovery task and fill in the form:

Field	What to enter
Vault URL	Your Vault address from Section 7 (include `https://` and the port)
Vault Namespace	Enterprise namespace, or leave empty
Authentication Method	Choose AppRole, Token, or Username and Password
AppRole Role ID and Secret ID	AppRole only: the two values copied in Section 6.1
Vault Token	Token only: the token copied in Section 6.2
Username and Password	Userpass only: the values from Section 6.3
Secrets Engines to Scan	Select PKI, Transit, and/or KV
Scan Name	A label for this run, for example `HashiCorp_Vault_Discovery`

Save and run the task.

---

9. Validate

After running the discovery task, confirm:

- The task completes without a permission or authentication error.
- Certificates from your PKI engine and keys from your Transit engine appear in the inventory.

You can double check the credential access directly in Vault using the same account discovery uses:

```
# AppRole example: log in, then confirm the paths are readable
vault write auth/approle/login role_id="<ROLE_ID>" secret_id="<SECRET_ID>"
vault list pki/certs
vault list transit/keys
vault kv list secret/
```

---

10. Troubleshooting

Symptom	Likely cause	Fix
`permission denied`	The policy is missing a path, or the credential is not attached to `cbom_discovery`	Check Section 5 again and confirm the auth role or user lists `cbom_discovery` under token policies
Nothing discovered	The engine names do not match, or the engines are empty	Verify the mount paths (`pki` , `transit` , `secret`) and that they contain material
`connection refused` or timeout	Wrong Vault URL, or Vault not reachable from where discovery runs	Confirm the host, port, and `http` or `https` . Use a network reachable address, not a loopback address

Symptom	Likely cause	Fix
`Vault is sealed`	Vault was restarted and not unsealed	Run `vault operator unseal` (Section 2, Option B)
AppRole login fails	Wrong Role ID or Secret ID, or the Secret ID expired	Generate the Secret ID again (Section 6.1)
TLS or certificate error connecting	The Vault TLS certificate is not trusted by the discovery host	Use a Vault certificate issued by a trusted CA

Final Checklist

- ☐ Vault installed, unsealed, and reachable at a known Vault URL
- ☐ PKI, Transit, and KV engines enabled and holding the material you want discovered
- ☐ Read only `cbom_discovery` policy created
- ☐ An authentication method set up (AppRole recommended) and attached to the policy
- ☐ Credentials copied (Role ID plus Secret ID, or Token, or Username plus Password)
- ☐ Values entered in the HashiCorp Vault discovery configuration and the task runs without errors