Vault Secrets Access Primer
A Detailed, Practical Guide

Compiled Notes

November 1, 2025

Abstract

This document provides a practical and detailed overview of HashiCorp Vault for secrets management. It covers architecture, security model, authentication and authorization, token lifecycle, policies, secrets engines, leases and revocation, auditing, high availability, operations runbooks, observability, and common troubleshooting. It includes hands-on examples using cURL and policy snippets. Graphical diagrams are intentionally omitted.

Contents

1 Executive Summary

Vault is a platform for securely storing, generating, and brokering access to sensitive data such as API keys, database credentials, certificates, and encryption keys. It encrypts data, enforces least-privilege access via policies, issues short-lived credentials, and records access for auditability. Production-grade deployments focus on strong transport security, careful unseal and key management, a minimal policy surface, short time-to-live (TTL) leases, and robust audit logging.

2 Quick Start (Developer Flow)

This section shows a minimal developer journey. Do not reuse dev settings for production.

2.1 Authenticate to get a client token

```
# Example: AppRole login (role_id + secret_id -> client token)
curl -sS --request POST \
  --data '{"role_id":"ROLE_ID","secret_id":"SECRET_ID"}' \
  https://vault.example.com/v1/auth/approle/login
# Response contains .auth.client_token. Export it as VAULT_TOKEN.
```

2.2 Read or write a secret on KV v2

```
# Read the latest version
curl -sS -H "X-Vault-Token: $VAULT_TOKEN" \
  https://vault.example.com/v1/kv/data/app/backend

# Write a new secret version
curl -sS -X POST -H "X-Vault-Token: $VAULT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"data":{"password":"s3cr3t"}}' \
  https://vault.example.com/v1/kv/data/app/backend
```

3 Architecture Overview

- **Single binary**: The Vault server exposes an HTTP API. The CLI is a thin client that calls the API.
- **Barrier encryption**: Vault never stores plaintext in storage backends. It encrypts data within an internal barrier; the backend only sees ciphertext.
- **Storage backends**: Integrated storage (Raft) or external systems (for example, Consul, cloud stores). Backends store both secret data and operational metadata (tokens, policies, mounts).
- **Pluggable engines**: Auth methods (how clients prove identity) and secrets engines (how data is generated or stored) are modular and enabled at paths.

3.1 Production topologies

- **HA cluster**: Multiple nodes behind a load balancer. One active node serves writes; standbys forward reads and promote on failover.
- **Integrated storage (Raft)**: Simple to operate; supports snapshots for backup and restore.
- **Consul-based**: External KV store provides HA and distribution; operational practices shift to the Consul layer.

4 Security Model

4.1 Transport and at-rest protection

- **In transit**: TLS everywhere. Pin a trusted CA, enforce strong ciphers.
- **At rest**: AES-256-GCM (or comparable) for barrier encryption. Storage backends cannot decrypt data.

4.2 Initialization, seal, and unseal

- **Initialization**: Creates a master key; splits it into shares using Shamir's Secret Sharing.
- **Sealed by default**: Vault starts sealed; administrators unseal by presenting a quorum of key shares.
- **Auto-unseal**: Optionally delegate unseal to a cloud KMS or HSM for hands-off restarts while maintaining hardware security guarantees.

4.3 Namespacing and multi-tenancy

If using a multi-tenant model, isolate teams with separate mounts and policies. Enterprise features can further segment namespaces.

5 Authentication and Authorization

5.1 Auth methods (identity providers)

Common methods include: AppRole, LDAP/AD, GitHub, JWT/OIDC (workload identity), Kubernetes ServiceAccount, cloud provider IAM. Each method issues a token after validating identity.

5.2 Tokens

- **Properties**: Tokens carry policies, a TTL, and possibly usage limits.
- **Types**: Service tokens (renewable), batch tokens (lightweight, not renewable), periodic tokens (fixed renew cadence).
- **Response wrapping**: Server returns a short-lived single-use wrapper token; unwrap at the consumer to reduce secret exposure. Wrapper contents live in the caller's cubbyhole until unwrapped or expired.
- **Orphan tokens**: Tokens without parents that will not be revoked through a parent chain; useful for automation with care.

5.3 Policies (RBAC)

Policies define capabilities over paths (read, list, create, update, delete, sudo). Keep policies small and composable.

Minimal example (HCL shown as text).

```
# Allow reading a single KV v2 path and listing its parent
path "kv/data/app/backend" {
  capabilities = ["read"]
}
path "kv/metadata/app" {
  capabilities = ["list"]
}
```

Attach policies to identities.

```
# Create a named policy from an HCL file
vault policy write app-backend ./policy.hcl

# Associate that policy with an AppRole, group, or auth method role as appropriate
```

6 Secrets Engines

Secrets engines run behind mount paths and define how secrets are stored or generated.

6.1 KV engine (v1 vs v2)

- **KV v2**: Versioned key-value store with metadata and soft deletes.
- **KV v1**: Simpler, no versioning. Prefer v2 for most cases.

6.2 Dynamic credentials

On read, Vault generates short-lived credentials (for example, database logins). When the lease expires or is revoked, credentials are disabled automatically.

6.3 Transit engine (encryption as a service)

The transit engine provides crypto operations without revealing keys to clients. Applications send plaintext and receive ciphertext, or vice versa.

6.4 PKI engine (certificates)

The PKI engine mints X.509 certificates. You can create an internal CA, issue leaf certificates with short TTLs, and revoke as needed.

Examples.

```
# Enable KV v2 at path "kv"
vault secrets enable -path=kv kv-v2

# Enable Transit and create a key
vault secrets enable transit
vault write -f transit/keys/app-crypto

# Encrypt and decrypt with Transit
echo -n 'payload' | base64 > b64.txt
vault write transit/encrypt/app-crypto plaintext=$(cat b64.txt)
# => returns ciphertext "vault:v1:..."
vault write transit/decrypt/app-crypto ciphertext="vault:v1:..."

# Enable PKI and set a short default TTL
vault secrets enable pki
vault write pki/config/urls issuing_certificates="https://vault/pki/ca"
↪  crl_distribution_points="https://vault/pki/crl"
vault write pki/roles/app-dot-internal allowed_domains="app.internal"
↪  allow_subdomains=true max_ttl="24h"
vault write pki/issue/app-dot-internal common_name="api.app.internal" ttl="12h"
```

7 Leases, Renewal, and Revocation

- **Leases**: Apply to tokens and dynamic secrets. Each has a TTL and possibly a maximum TTL.
- **Renewal**: Clients can renew before expiry if allowed by policy and max TTL.
- **Revocation**: On expiry or admin action, Vault revokes secrets and, for dynamic credentials, disables them at the source system.

Working with leases.

```
# Renew a token or lease
vault token renew            # for current token
vault lease renew <lease_id> # for specific lease

# Explicit revoke
vault token revoke <token>
vault lease revoke <lease_id>
```

8 Auditing

- **Fail closed**: If Vault cannot write to any active audit device, requests are denied.
- **Devices**: File, syslog, socket, and others. Use structured logging and ship to a centralized SIEM.
- **Coverage**: Authentication attempts, secret access, policy changes, and admin operations.

Examples.

```
# Enable a file-based audit device
vault audit enable file file_path=/var/log/vault_audit.log

# Disable an audit device
vault audit disable file
```

9 High Availability and Performance

- **Active/standby**: One active handles writes; standbys forward and can promote on failure.
- **Request forwarding**: Standbys forward requests to the active when appropriate.
- **Performance standbys**: Offload read-heavy traffic and cryptographic offload in larger clusters.
- **Load balancing**: Health-check for `/v1/sys/health`. Favor sticky sessions for write-heavy clients if needed.

10 Operations Runbooks

10.1 Backups and restore

- **Integrated storage**: Use snapshots. Secure, test restore regularly.
- **External backends**: Follow backend-native snapshot procedures plus encrypt backups and protect keys.

10.2 Key rotation

Rotate transit keys and PKI CAs at planned intervals. For transit, use key versioning and rewrap; for PKI, create new intermediates and roll out leaf cert issuance to the new chain.

10.3 Upgrade procedure

1. Read release notes and verify compatibility.
2. Snapshot storage or take a backend backup.
3. Upgrade standbys first, then failover, then former active.
4. Validate audit, auth mounts, and secrets engine mounts.

10.4 Disaster recovery

Document **RTO** and **RPO**. Practice controlled failovers, unseal workflows, and token reissuance. Verify that audit and monitoring continue after failover.

11 Observability

- **Metrics**: Export to Prometheus or your APM. Track request rates, latency, error codes, seal status, leader elections, and mount health.
- **Logs**: Separate audit logs from operational logs. Protect integrity and retention.
- **Alerts**: On seal events, leader changes, audit write failures, storage lag, auth failures, error spikes, and low TTL or renewal failures.

12 Security Best Practices (Checklist)

Area	**Recommended Practices**
TLS	Enforce TLS with modern ciphers; pin CA; rotate server certs regularly.
Unseal	Use auto-unseal with a cloud KMS or HSM; protect and rotate key material.
Auth	Prefer workload identity (Kubernetes, IAM, OIDC) over long-lived static creds.
Policies	Default deny; minimal scopes; split read from write; review regularly.
TTLs	Use short TTLs and max TTLs; prefer dynamic credentials.
Transit	Use Transit for application crypto instead of embedding keys in apps.
Audit	Enable at least two audit devices; ship logs to SIEM; alert on write failures.
Backups	Automate snapshots; encrypt at rest; test restore; document RTO and RPO.
Upgrades	Stage in lower environments; snapshot before change; roll forward with validation.
Secrets Hygiene	Tag owners; set rotation cadence; remove unused mounts and policies.

13 Common Troubleshooting

13.1 Permission denied on a path

- Check token policies; confirm capabilities match engine path format.
- For KV v2, data paths are `kv/data/...` for data and `kv/metadata/...` for listing.

13.2 Client cannot connect

- Verify `VAULT_ADDR` and TLS trust chain.
- Check load balancer forwarding and health checks.

13.3 Sealed state after restart

- Provide the required unseal shares or verify KMS/HSM connectivity for auto-unseal.

13.4 Dynamic creds not revoked

- Confirm lease TTLs and revocation settings on the engine.
- Check backend permissions for revocation actions (for example, DB role).

14 API and CLI Recipes

14.1 Enable and configure engines

```
# Enable a versioned KV engine at path "kv"
vault secrets enable -path=kv kv-v2

# Tune default and max TTLs for KV metadata
vault secrets tune -default-lease-ttl=1h -max-lease-ttl=24h kv/
```

14.2 Issue and revoke tokens

```
# Create a child token with restricted policy and TTL
vault token create -policy="app-backend" -ttl=30m -renewable=true

# Revoke a specific token
vault token revoke hvs.xxxxx
```

14.3 Use response wrapping

```
# Wrap a secret for out-of-band delivery (TTL 2 minutes)
vault kv get -wrap-ttl=2m kv/app/backend
# Receiver unwraps:
vault unwrap <wrapping_token>
```

14.4 Kubernetes auth (conceptual)

```
# Configure the Kubernetes auth method (service account JWT, CA cert, host)
vault auth enable kubernetes
vault write auth/kubernetes/config \
  kubernetes_host="https://$K8S_HOST" \
  kubernetes_ca_cert="@/path/ca.crt" \
  token_reviewer_jwt="@/var/run/secrets/kubernetes.io/serviceaccount/token"

# Map a service account to a policy
vault write auth/kubernetes/role/app \
  bound_service_account_names=api \
  bound_service_account_namespaces=default \
  policies=app-backend \
  ttl=30m
```

15 Operational Policies and Review Cadence

- **Quarterly**: Rotate Transit keys; review policies and mounts; verify backups and restores.
- **Monthly**: Sample audit trails; confirm access ownership; rotate long-lived app credentials if any remain.
- **Per release**: Read release notes; stage and validate upgrades; check health and audit after deployment.

16 Appendix A: Minimal Dev Compose (Optional)

This is a quick-start `docker-compose.yml` to experiment locally. Do not use for production.

```yaml
version: "3.8"
services:
  vault:
    image: hashicorp/vault:latest
    environment:
      - VAULT_DEV_ROOT_TOKEN_ID=root
      - VAULT_DEV_LISTEN_ADDRESS=0.0.0.0:8200
    ports:
      - "8200:8200"
    cap_add:
      - IPC_LOCK
    command: "server -dev"
```

17 Appendix B: Handy Endpoints (Cheat Sheet)

Endpoint	Purpose
`/v1/sys/health`	Health and seal status.
`/v1/auth/approle/login`	Exchange role_id and secret_id for a client token.
`/v1/kv/data/...`	KV v2 read/write data.
`/v1/transit/encrypt/...`	Encrypt plaintext; returns ciphertext.
`/v1/transit/decrypt/...`	Decrypt ciphertext; returns plaintext.
`/v1/pki/issue/...`	Issue certificates from a PKI role.
`/v1/sys/audit`	Configure audit devices.
`/v1/auth/token/revoke`	Revoke a token.