EIN VAULT FÜR ALLE FÄLLE

Java Forum Stuttgart 2018

Dominik Schadow
bridgingIT
spring:
datasource:
  name: myDatabase
  username: myDatabaseUser
  password: mySuperSecretDatabasePassword
management:
  context-path: /admin
  security:
    enabled: true
logging:
  level: warn

- [hard coded]
- [widely accessible]
- [rarely rotated]
Store secrets securely (cloud & on-premises)
1. **Vault** basics that make storing any credentials easy and secure

2. Using **Spring Cloud Vault**, **Spring Vault** and **Vault** in a Spring Boot application

3. **Spring (Cloud) Vault** and **Vault** in action
A tool for managing secrets.
Vault manages static and dynamic secrets

1. Various **authentication** and **authorization** possibilities

2. Extensible **storage** and **secret backend architecture**

3. Auditing of who accessed **what secret when**
Authentication

**Static and dynamic tokens** (main method of authentication)

Enable **additional auth backends**: AppRole, AWS, Azure, Google Cloud, Kubernetes, LDAP, Okta, RADIUS, TLS Certificates, Username & Password

```
$ vault auth list
Path   Type      Accessor                      Description               
----   ----      ---------                    ------------------------
token/ token    auth_token_dba60d7b          token based credentials
userpass/ userpass auth_userpass_99e7f134 n/a
```
$ vault auth enable userpass
Success! Enabled userpass auth method at: userpass/

$ vault write auth/userpass/users/dummyuser password=dummypassword
Success! Data written to: auth/userpass/users/dummyuser

$ vault login -method=userpass username=dummyuser
Password (will be hidden):
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.
Authorization with policies

**Access control policies** for secrets to protect them (deny by default)

Policies determine what **specific actions are (not) allowed on specific paths or endpoints**

**Provided in HCL** - HashiCorp Configuration Language (like JSON)
path "secret/*" { 
    capabilities = ["create", "read", "update", "delete", "list"]
}

# Explicitly denies secret/production
# (takes precedence)
path "secret/production" { 
    capabilities = ["deny"]
}
$ vault write sys/policy/dev-policy policy=@dev-policy.hcl
Success! Data written to: sys/policy/dev-policy

$ vault list sys/policy
Keys
----
default
dev-policy
root

$ vault write auth/userpass/users/dummyuser password=dummypassword policies=dev-policy
Success! Data written to: auth/userpass/users/dummyuser
Secret backend never sees plaintext

Responsible for **generating and storing secrets** as key/value, Consul, databases, AWS, Google …

**Static and dynamic secrets**, e.g. for AWS and databases

**Lease time** for new secrets
$ vault secrets enable kv
Success! Enabled the kv secrets engine at: kv/

$ vault secrets disable kv
Success! Disabled the secrets engine (if it existed) at: kv/
Auditing is disabled by default

**Detailed audit log of all client interaction** (sensitive data hashed)

Can **block Vault operations** once enabled

```bash
$ vault audit enable file file_path=~:/vault_audit.log
Success! Enabled the file audit device at: file/
```
Setting up Vault for dev or prod
# start Vault in dev mode

```
vault server -dev
  -dev-root-token-id="00000000-0000-0000-0000-000000000000"
  -dev-listen-address="127.0.0.1:8200"
```

# Vault is initialized and unsealed

# single unseal key
# in-memory storage
# authenticated with CLI
# start Vault in prod mode
vault server -config=vault-local.conf

# init Vault
vault operator init -key-shares=5 -key-threshold=2

# Shamir’s Secret Sharing

Unseal Key 1: 915Xw+AZDusVCzeILkadNImFYqy0FRgy3+APKWu7WNGx
Unseal Key 2: zUHWmiuyOtgtbGbF7IuygLSdOZT9CAPA0lrdsLres2Km
Unseal Key 3: YuPDKST15rD6gC5rJAw8da+L8kNF01Ek5dz4s9tKfSP/
Unseal Key 4: thhnx95x6NO/g7JYaV7wGhYdjQbguJNea21RgSbYVCF2
Unseal Key 5: ZHvU4RRADCxnY7w6jopjSN91od760eRJUg3mSp4yPmg9

Initial Root Token: 91cfa64f-04e5-297f-6bb7-66b28c726483
# unseal Vault key 1

```
vault operator unseal
```

915Xw+AZDusVCzeILkadNImFYqyoFRgy3+APKWu7WNGx

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seal Type</td>
<td>shamir</td>
</tr>
<tr>
<td>Sealed</td>
<td>true</td>
</tr>
<tr>
<td>Total Shares</td>
<td>5</td>
</tr>
<tr>
<td>Threshold</td>
<td>2</td>
</tr>
<tr>
<td>Version</td>
<td>0.10.1</td>
</tr>
<tr>
<td>Unseal Progress</td>
<td>1/2</td>
</tr>
<tr>
<td>Unseal Nonce</td>
<td>0a73fece-4d26-7f9-...</td>
</tr>
</tbody>
</table>
# unseal Vault key 2

```
vault operator unseal
```

```
zUHWmiuyOtgtbGbF7IuygLSdOZT9CAPA0lrdsLres2Km
```

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seal Type</td>
<td>shamir</td>
</tr>
<tr>
<td>Sealed</td>
<td>false</td>
</tr>
<tr>
<td>Total Shares</td>
<td>5</td>
</tr>
<tr>
<td>Threshold</td>
<td>2</td>
</tr>
<tr>
<td>Version</td>
<td>0.10.1</td>
</tr>
<tr>
<td>Cluster Name</td>
<td>vault-cluster-6f6d10ea</td>
</tr>
<tr>
<td>Cluster ID</td>
<td>c374b289-3fb7-e547-5a59-...</td>
</tr>
</tbody>
</table>
Init and manage Vault and all secrets in the browser

Vault UI is enabled by default in dev mode only

http://localhost:8200/ui
# vault-local.conf

storage "inmem" {
    
}

listener "tcp" {
    address = "127.0.0.1:8200"
    tls_disable = 1
}

# Enable Vault UI
ui = true
Access **resources** via path

- Authentication backends
- Storage backends
- Secrets
- Policies
- Configurations
vault write secret/conferences/2018
title="Ein Vault für alle Fälle"

key/value format
(generic backend)

$ vault read secret/conferences/2018
Key          Value
  ___        _____
refresh_interval  768h
title          Ein Vault für alle Fälle
All data stored securely

Multiple unseal keys

Vault server security
Spring Cloud Vault, Spring Vault and Vault
Spring Cloud Vault

1. **Externalized configuration in Vault** for Spring Cloud Config Server

2. Initialize **Spring Environment** with secrets from Vault

3. **Naming conventions** as with property files
Spring Cloud Vault

/secret/{application}/{profile}
/secret/{application}
/secret/{defaultContext}/{profile}
/secret/{defaultContext}

```bash
$ vault read secret/config-client-vault
Key                  Value
---                  -----          
refresh_interval     768h
application.name     Config Client Vault
application.profile  Demo
```
spring:
  application:
    name: config-client-vault
cloud:
  config:
    uri: http://localhost:8888
    username: user
    password: secret
Spring Vault

1. **Client side** support for Vault

2. **Manage** (list, read, write, delete) secrets
spring:
  vault:
    host: localhost
    port: 8200
    scheme: http
    authentication: token
    token: ac91ad0b-038f-b088-e101-dd6af9...
# VaultOperations (VaultTemplate) for
# list, read, write, and delete operations
# Similar to RestTemplate

```java
vault.list("secret");

vault.read("secret/12345", Secret.class);

vault.write("secret/12345", secret);

vault.delete("secret/12345");
```
Vault in Action
Demo
Summary

Vault stores secrets securely and provides secret management functionality.

Vault integrates with Spring Boot and Spring Cloud Config.

Vault requires strict access control and must be maintained as any other application.
Demo Project
https://github.com/dschadow/CloudSecurity

Spring Cloud
https://projects.spring.io/spring-cloud

Spring Cloud Vault
https://cloud.spring.io/spring-cloud-vault

Spring Vault
https://projects.spring.io/spring-vault

Vault
https://www.vaultproject.io

Pictures
https://www.dreamstime.com