Securing Oracle Database with HashiCorp Vault Enterprise

Date: March 12, 2021

Author: Bobby Curtis, MBA

Table of Contents

Executive Summary

Oracle provides many security options within its database framework, but when organizations start to scale their infrastructure these options can quickly become complicated and hard to manage. HashiCorp Vault provides a centralized way of managing security within a larger, scaled environment. The theme of this paper is aimed at providing organizations a view into how HashiCorp Vault can be installed and configured against a single Oracle database and then expanded out to multiple databases from a single interface.

Installation

The installation process outlined here is for the installation of HashiCorp Vault using the pre-compiled binary. For further installation instructions, refer to the online documentation at https://www.vaultproject.io/docs/install.

To install HashiCorp Vault, download the pre-compiled binary for the platform where it is going to run. In this case, the platform will be on Oracle Enterprise Linux (x86-64).

1. Download the binary set for HashiCorp Vault (Open Source) (here).

   Note: If you are using HashiCorp Vault Enterprise, you will need to talk to a HashiCorp Sales rep to obtain the enterprise binaries.

2. After obtaining the zip file, unzip the zip file into any directory where HashiCorp Vault will run.

   ```
   $ unzip vault_1.5.2_linux_amd64.zip -d {{ directory }}
   ```

3. Update the $PATH environment variable to specify the location of the HashiCorp Vault binary. If wanting to manage HashiCorp Vault from the command line, execute the following:

   ```
   $ export PATH={{ directory }}:$PATH
   ```

4. Verify the installation of HashiCorp Vault. Using the help command will produce output that will give you commands and options used by HashiCorp Vault.

   ```
   $ vault -h
   ```

Configure Oracle Database

To secure Oracle Database with HashiCorp Vault, some minimal items need to be established. One of these items is a dedicated user that will act as the primary user required for HashiCorp Vault to work against the database. This user will also be known as the "vault root" user.

Within an Oracle Pluggable Database (PDB), the Vault user needs to be created. Example 1 shows what the vault root user should look like. This user will not have any database objects and will only log in to the Oracle Pluggable Database and managed secrets for other users and service accounts.

Example 1: Vault Root User Creation

```
CREATE USER VAULTADMIN
IDENTIFIED BY {{ Password }}
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP
QUOTA UNLIMITED ON USERS
ACCOUNT UNLOCK;

GRANT CONNECT TO VAULTADMIN;
GRANT RESOURCE TO VAULTADMIN;

GRANT CREATE SESSION TO VAULTADMIN;
GRANT SELECT ANY DICTIONARY TO VAULTADMIN;
GRANT SELECT ANY TABLE TO VAULTADMIN;
GRANT UNLIMITED TABLESPACE TO VAULTADMIN;
GRANT ALTER USER TO VAULTADMIN;
```

Note: HashiCorp Vault does not support the use of a common user (C##), preventing a user in the Container Database from being used. If container database support is needed, the SYS or SYSTEM user needs to be configured as the account within the HashiCorp Vault connection.

Configure HashiCorp Vault

The configuration file for HashiCorp Vault uses the HashiCorp Configuration Language (HCL) to specify how the Vault is to start and run. This HCL file provides details on storage, listener address, and any other information needed to bring up the HashiCorp Vault. Example 2 is an example of a typical HCL file:

Example 2: Vault HCL file (vault.hcl)

```
cat > ${HASHICORP_CONFIG}/vault.hcl <<EOF
storage "file" {
        path = "/opt/app/hashicorp/data"
}

listener "tcp" {
        address = "0.0.0.0:8200"
        tls_disable = 1
}

api_addr = "http://0.0.0.0:8200"

ui = true

disable_mlock = true

plugin_directory = "/opt/app/hashicorp/plugin"
EOF
```

The `${HASHICORP_CONFIG}` variable is an environment variable that specifies where the config directory is within the system. In a new system, this would be under the directory created for HashiCorp Vault.

For more details on how to configure HashiCorp Vault, refer to documentation here.

Start HashiCorp Vault

To start the HashiCorp Vault Server, simply use the server command. The `server` command starts a Vault server that responds to API requests.

To start a Vault Server, issue the following command:

```
$ vault server
```

After starting the server, the Vault will be in a sealed state.

Initialize HashiCorp Vault

With the HashiCorp Vault server running, the next thing to do is to initialize the Vault. Initialization is the process by which Vault's storage backend is prepared to receive data. During this process, Vault generates an "in-memory" master key. It applies Shamir's secret sharing algorithm to disassemble that master key into several key shares that are required to come together to generate the master key.

Initializing HashiCorp Vault requires defining the address to which HashiCorp Vault will listen and then initialize that server. The following steps are performed from the command line to initialize the Vault:

```
$ export VAULT_ADDR=http://{{ host }}:{{ port }}
```

Initialization using default options:

```
$ vault operator init
```

To specify the number of recovery shares and recovery threshold:

```
$ vault operator init -recovery-shares=1 -recovery-threshold=1
```

The generated keys can be saved to a text file stored on the operating system if desired. The "master key" is only stored in memory, and if the HashiCorp Vault restarts, this key will change, and new keys will need to be reinitialized.

Open HashiCorp Vault

After initializing Vault, the next thing to be done is to "unseal" the Vault. To "unseal" Vault, provide the secret key shares. The number of key shares required is determined by the specified options when Vault is initialized.

To unseal the Vault, the following must be done:

```
$ vault operator unseal {{ key_1 }}
$ vault operator unseal {{ key_2 }}
$ vault operator unseal {{ key_3 }}
$ vault login {{ master_key }}
```

The above commands will open the HashiCorp Vault and allow the user who unsealed it to interact with Vault.

Providing a License Key – Enterprise Version Only

As with any enterprise product, the HashiCorp Vault product requires the purchase of a license key. When you purchase the license key, HashiCorp will provide the key via email as a SHA256 key. This key needs to be added to the system registry within the Vault.

To add a license key to HashiCorp Vault, with the Vault unsealed and open, run the following:

```
$ vault write sys/license text={{ SHA256_KEY }}
```

Oracle Plug-In for HashiCorp Vault

HashiCorp Vault, by default, does not support Oracle Database authentication. Others have recognized this and have built a plug-in that is supported for use with HashiCorp Vault. Some pre-requisites have to be met before the plug-in will work. Additional information on the Oracle Plug-In can be found on the GitHub site where the plug-in is maintained (here).

Pre-Requisites

Pre-requisites have to be defined and configured before the plug-in will work. These pre-requisites are:

Oracle Instant Client and associated development tools

- `Oracle-instantclient{{ version }}-basic.x86_64`
- `Oracle-instantclient{{ version }}-devel.x86_64`
- `Oracle-instantclient{{ version }}-tools.x86_64`
- `Oracle-instantclient{{ version }}-sqlplus.x86_64`

These versions of the Oracle Instant Client will be installed in default locations on a Linux platform. HashiCorp Vault Plug-In for Oracle will look in these default locations to verify that the needed libraries exist. If the plug-in cannot find the library files it expects, then the plug-in will fail to initialize and connect to an Oracle Database.

Define a Plug-in Directory

Before installing the Oracle Plug-In, define a directory structure where the plug-in will reside. After setting up the directory structure, update the `vault.hcl` file to identify the directory and restart Vault.

Download the Oracle Plug-In

The Oracle Plug-In for HashiCorp Vault can be downloaded and saved into the plug-in directory. To download the current release of the plug-in, access the plug-in releases page – here.

Registering the Oracle Plug-In

After downloading the Oracle plug-in, the next thing is to register the Oracle plug-in with HashiCorp Vault. The registration process requires you to find the SHASUM for the plugin and write the plug-into Vault's plug-in structure.

Find the SHASUM

Once the Oracle Plug-In is unzipped, then you need to find the SHASUM value and change the permissions on the Oracle Plug-In so the Vault user can access it.

```
$ shasum -a 256 ${HASHICORP_PLUGIN}/vault-plugin-database-oracle | sed
's/\s.*$//' > ${HASHICORP_PLUGIN}/oracle-plugin.sha256
$ chown vault:vault ${HASHICORP_PLUGIN}/oracle-plugin.sha256
```

Enable Database Key Vault

Before writing the Oracle plug-into the system registry for Vault, you need to enable the Secrets Engine for databases. The following is executed to enable the secrets engine for databases:

```
$ vault secrets enable database
```

Write to Vault

You will tell Vault that the plug-in is available for security operations against the Oracle database by writing the plug-in to Vault. Execute the following:

```
$ vault write sys/plugins/catalog/database/oracle-database sha256=$(cat
${HASHICORP_PLUGIN}/oracle-plugin.sha256) command=vault-plugin-database-
oracle
```

The plug-in is placed in the systems registry of Vault that corresponds to the catalog related to databases.

Validating the Plug-In

Once the plug-in has been registered with HashiCorp Vault, it can be validated by merely checking the system registry. This is done by running the following command:

```
$ vault plug-in list database
```

There should be an entry related to the plug-in that was added for Oracle in the resulting list.

Adding Credentials

With the HashiCorp Vault setup, the next thing is to establish a connection to the Oracle Database and assign credentials that will be managed via Vault.

Connection

Before HashiCorp Vault can manage any credentials, a database connection needs to be established. This is done by ensuring that the local tnsnames.ora file is configured and that sql*plus from the Vault host can connect. After a successful connection, then Vault will be able to connect and manage credentials.

To setup a connection the following command should be ran:

```
$ vault write database/config/PDB plugin_name=oracle-database
connection_url="vaultadmin/**********@ORCLPDB1" allowed_roles="*"
```

After the connection has been configured, it can be confirmed by a simple command

```
$ vault read database/config/PDB
```

The output of this command will show you what is configured for the connection.

Static Roles

Roles are how HashiCorp Vault manages database users within its structure. For most Oracle Database implementation, static roles can be used to manage the password rotation process. To create a static role, the following commands must be executed:

```
$ vault write database/static-roles/tstusr_acct db_name=PDB
rotation_statements=@/vagrant/scripts/rotate_common.sql username="tstusr"
rotation_period="300s"
creation_statements=@/vagrant/scripts/rotate_common.sql
```

After writing the static role to the Vault data store, it can be confirmed by the following command:

```
$ vault list database/static-roles
```

Get Password

After the static role has been created, the password for the role needs to be identified. To retrieve the password, the static role has to be read.

```
$ vault read database/static-creds/tstusr_acct
```

The password that is provided will work for the duration that it is intended to live. This can be seen in the `TTL` setting when reading the credential. See example 3 for a sampling of the output.

Example 3: Read output for static credential

```
Key                    Value
---                    -----
last_vault_rotation    2021-03-12T02:16:26.055097177Z
password               A1a-pSVk5Bm0nsPJx0wF
rotation_period        5m
ttl                    4m43s
username               tstusr
```

In the above example, the password that is listed is random and will expire after five minutes. The `TTL` setting indicates the time remaining on the password. Once the `TTL` expires, then the password is changed, and the user account will not be able to log in until a new password is provided.

RheoData
8410 Fairthron Way
Douglasville, GA 30135
(888) 451-5594

www.rheodata.com