One Identity Safeguard for Privileged Sessions 6.5

Hashicorp Vault as Credential Store

Legend

WARNING: A WARNING icon highlights a potential risk of bodily injury or property damage, for which industry-standard safety precautions are advised. This icon is often associated with electrical hazards related to hardware.

CAUTION: A CAUTION icon indicates potential damage to hardware or loss of data if instructions are not followed.

SPS Hashicorp Vault as Credential Store
Updated - March 2020
Version - 6.5

Contents

Introduction

This tutorial describes how you can connect One Identity Safeguard for Privileged Sessions (SPS) and your Hashicorp Vault with a Credential Store Plugin.

SPS can interact with Hashicorp Vault and can automatically retrieve the password or SSH key of the target host to form a comprehensive Privileged Access Management solution to protect critical assets and meet compliance requirements.

Technical requirements

To successfully connect SPS with Hashicorp Vault, you need the following components:

- A valid, working Hashicorp Vault server or cluster of servers with the following configuration:
  - In case of explicit authentication:

    A proxy user must be created on the Hashicorp Vault that has access to the secrets holding passwords and keys. The plugin will be using this "proxy user" to access Hashicorp Vault.
  - In case of gateway-based authentication:

    SPS reuses the username/password from the gateway authentication to authenticate on the Hashicorp Vault. This requires password-based gateway authentication on SPS and that the same user is available on the Hashicorp Vault with the same password, and has access to the secrets holding passwords and keys. The best way is to use an LDAP/AD-based authentication backend.
- A SPS appliance (virtual or physical), at least version 6.2.0.
- A Credential Store plugin for Hashicorp Vault.

  SPS uses plugins to interact with third-party credential stores and password vaults. One Identity provides the sample Hashicorp Vault plugin free of charge, and provides help to customize it for your environment.

How SPS and Hashicorp Vault work together

Authentication:

The plugin can use either explicit or gateway-based credentials.

- In case of explicit authentication:

  A proxy user must be created on the Hashicorp Vault that has access to the secrets holding passwords and keys. The plugin will be using this "proxy user" to access Hashicorp Vault.

- In case of gateway-based authentication:

  SPS reuses the username/password from the gateway authentication to authenticate on the Hashicorp Vault. This requires password-based gateway authentication on SPS and that the same user is available on the Hashicorp Vault with the same password, and has access to the secrets holding passwords and keys. The best way is to use an LDAP/AD-based authentication backend.

Secret lookup:

Interactive scenario: If the secrets in Hashicorp are stored in an unstructured way, SPS will have to retrieve the path to the secret from the end-user.

Automatic scenario: If the secrets are organized around server user names in Hashicorp Vault, then the path to the secret is generated from configuration and the server user name.

Hashicorp Vault scenarios

The following scenarios are the most common methods to use SPS and Hashicorp Vault together.

Interactive scenario

If the data in Hashicorp is stored in an unstructured way, SPS will have to retrieve the path to the secret from the end-user. This is done in the following way:

1. Configure an Authentication and Authorization (AA) plugin (for example, Okta). For details on configuring the plugin, see the respective plugin documentation.
2. Make sure to configure the `question_1` parameter in that AA plugin the following way:

   ```
   [question_1]
   prompt=Enter the path to the secret that you want to retrieve from Hashicorp.
   key=vp
   disable_echo=no
   ```

   You can change the prompt message in the `prompt` parameter, but make sure that you enter `vp` as the value of `key`.
3. As a result, the Hashicorp Vault plugin can now retrieve the compound path (described in `secrets_path`) from the previously configured AA plugin. Therefore, in the Interactive scenario, you do not have to configure the `secrets_path` parameter.
4. Configure the `key_field` and the `password_field` options of the Hashicorp Vault plugin. The plugin will retrieve the password or the key from these fields of the compound path that the user provides.
5. Configure the `default_type` option of the Hashicorp Vault plugin to set which type of credential the plugin should retrieve by default.
6. Note that when users enter the path to the secret, they can customize the path the following way to override the configuration of the plugin:
   - If the `default_type` is set, but the user wants to authenticate with another credential type (password instead of key, or key instead of password), the user can specify the credential type to use in the prompt.

     For password authentication, use the `password://` or `p://` prefix, for example:

     ```
     password://my/secrets
     ```

     For public key authentication, use the `key://` or `k://` prefix, for example:

```
password://my/secrets
```

- By default, the plugin retrieves the password or the key from the field set in the `password_field` or the `key_field` options. To retrieve the secret from a different field, append the name of the new field to the compound path after a hashtag (#) character, for example:

  ```
  my/secrets#mysecretfield
  ```

- The previous methods can be combined. The following example overrides both the credential type and the field name.

  ```
  key://my/secrets#mysecretfield
  ```

- If the path to the endpoint contains a literal slash (/) or hashmark (#) character, double this character. For example, if the path is `secrets/my#endpoint`, use `secrets/my##endpoint` to escape the special character.

Automatic scenario

In Hashicorp Vault, there is an endpoint under which the user names and passwords are stored as secrets. For example, `secrets/users`. The server username is then appended by the plugin to the path on-the-fly. This compound path points to an object that has the password or key as one of its fields.

For this scenario, you must configure the following parameters in the `[engine-kv-v1]` section:

- `secrets_path`
- `key_field`
- `password_field`

SPS Hashicorp Vault plugin parameter reference

This section describes the available options of the SPS Hashicorp Vault plugin.

The plugin uses an ini-style configuration file with sections and name=value pairs. This format consists of sections, led by a [section] header and followed by name=value entries. Note that the leading whitespace is removed from values. The values can contain format strings, which refer to other values in the same section. For example, the following section would resolve the `%(dir)s` value to the value of the `dir` entry (`/var` in this case).

```
[section  name]
 dirname=%(dir)s/mydirectory
 dir=/var
```

All reference expansions are done on demand. Lines beginning with `#` or `;` are ignored and may be used to provide comments.

You can edit the configuration file from the SPS web interface. The following code snippet is a sample configuration file.

```
[hashicorp]
 address=<address>
 port=8200
 authentication_method=<authentication_method>
 use_credential=explicit
 username=<username>
 password=<password>

 [engine-kv-v1]
 secrets_path=<path>
 key_field=key
 password_field=password
 delimiter=

 [tls]
 ca_cert  =  $[<trusted-ca-list-name>]

 [credential_store]
 name=<name-of-credential-store-policy-that-hosts-sensitive-data>

 [logging]
 log_level=info

 [https_proxy]
 server=<proxy-server-name-or-ip>
 port=3128
```

[hashicorp]

This section contains the options related to your Hashicorp Vault account.

```
[hashicorp]
address=<address>
port=8200
authentication_method=<authentication_method>
use_credential=gateway
username=<username>
password=<password>
```

address

Type:	string
Required:	no
Default:	N/A

Description: The address or hostname of the Hashicorp Vault. Separate more than one addresses with a comma (,).

port

Type:	integer
Required:	no
Default:	8200

Description: The port number of the Hashicorp Vault.

authentication_method

Type:	string
Required:	no
Default:	N/A

Description: The authentication method to use to connect to the Hashicorp Vault. The value can be one of the following: `ldap` or `userpass`.

use_credential

Type:	string
Required:	no
Default:	gateway

Description: The credential type to use. The value can be one of the following: `explicit` or `gateway`.

If you use the `explicit` credential type, you must also configure the `username` and `password` parameters.

username

Type:	string
Required:	If you have configured the `use_credential` parameter as `explicit`
Default:	N/A

Description: The username used to authenticate to the Hashicorp Vault.

password

Type:	string
Required:	If you have configured the `use_credential` parameter as `explicit`
Default:	N/A

Description: The password used to authenticate to the Hashicorp Vault.

CAUTION:

This parameter contains sensitive data. Make sure to store this data in your local Credential Store. Type the `$` value for this parameter in production.

For details, see Store sensitive plugin data securely.

Only enter a value different than `$` for this parameter in the configuration for testing purposes in a secure, non-production environment.

[engine-kv-v1]

This section contains the options related to your Hashicorp Vault account.

```
[engine-kv-v1]
secrets_path=<path>
key_field=key
password_field=password
default_type=
```

secrets_path

Type:	string
Required:	only in Automatic scenario
Default:	N/A

Description: The path of the endpoint under which the user names and passwords are stored as secrets. For example, `secrets/users`. The server username is then appended to the path on-the-fly. This compound path points to an object that has the password or key as one of its fields. You can specify the name of the field that stores the password and the key in the `password_field` and `key_field` options.

The user can override this field when using the Interactive scenario, see Interactive scenario.

If the path to the endpoint contains a literal slash (/) or hashmark (#) character, double this character. For example, if the path is `secrets/my#endpoint`, use `secrets/my##endpoint` to escape the special character.

key_field

Type:	string
Required:	no
Default:	key

Description: The value field to retrieve the SSH private key secret from.

The user can override this field when using the Interactive scenario, see Interactive scenario.

password_field

Type:	string
Required:	no
Default:	password

Description: The value field to retrieve the password secret from. This parameter is not related to the `password` parameter.

The user can override this field when using the Interactive scenario, see Interactive scenario.

default_type

Type:	key | password | empty string
Required:	no
Default:	empty string

Description: Determines the type of credential (key or password) that the plugin retrieves from the Hashicorp Vault. If not specified, the plugin attempts to retrieve both a key and a password.

If the `default_type` is set, but the user wants to authenticate with another credential type (password instead of key, or key instead of password), the user can specify the credential type in the prompt when using the Interactive scenario by beginning the secret path with `password://` or `key://` (you can use the `p://` or `k://` abbreviations as well).

[tls]

This section contains the options related to TLS settings.

Declaration

```
[tls]
enabled = yes
ca_cert = $[<trusted-ca-list-name>]
client_cert = <client-certificate-and-key>
```

enabled

Type:	boolean (`yes`|`no`)
Required:	no
Default:	yes

Description: To disable TLS completely, enter `no` as the value of this parameter.

ca_cert

Type:	string
Required:	no
Default:	N/A

Description: Configure this parameter to enable client-side verification. The certificate shown by the server will be checked with this CA.

If the value of this parameter is `$[<trusted-ca-list-name>]`, the certificates are retrieved from the trusted CA list configured on SPS, identified by the name.

When the certificate is inserted into the configuration file (`<ca-certificate-chain>`, it must be in PEM format and all the new lines must be indented with one whitespace. If it is a chain, insert the certificates right after each other.

client_cert

Type:	string
Required:	no
Default:	N/A

Description: Configure this parameter to enable server-side verification.

If the value of this parameter is `$`, the certificate identified by the section and option pair is retrieved from the configured Credential Store.

When the certificate is inserted into the configuration file, it must be in PEM format and all the new lines must be indented with one whitespace. Note that encrypted keys are not supported.

[credential_store]

This section contains settings related to storing sensitive information of the plugin.

Declaration

```
[credential_store]
name=<name-of-credential-store-policy-that-hosts-sensitive-data>
```

name

Type:	string
Required:	no
Default:	N/A

Description: The name of a local Credential Store policy configured on SPS. You can use this Credential Store to store sensitive information of the plugin in a secure way (for example, the `secrets_path` value in the `[hashicorp]` section).

For details, see Store sensitive plugin data securely.

[logging]

This section contains logging-related settings.

Declaration

```
[logging]
log_level=info
```

log_level

Type:	integer or string
Required:	no
Default:	info

Description: The logging verbosity of the plugin. The plugin sends the generated log messages to the SPS syslog system. You can check the log messages in the **Basic settings > Troubleshooting > View log files** section of the SPS web interface. To show only the messages generated by the plugins, filter on the `plugin:` string.

The possible values are:

- `debug`
- `info`
- `warning`
- `error`
- `critical`

For details, see Python logging API's log levels: Logging Levels.

[https_proxy]

This section contains HTTPS proxy-related settings.

Declaration

```
[https_proxy]
 server=<proxy-server-name-or-ip>
 port=3128
```

server

Type:	string
Required:	no
Default:	N/A

Description: The name or IP address of the HTTPS proxy server.

name

Type:	integer
Required:	no
Default:	3128

Description: The port number of the HTTPS proxy server.

Store sensitive plugin data securely

By default, the configuration of the plugin is stored on SPS in the configuration of SPS. Make sure that you store the sensitive parameters (for example, `password`) of the plugin in an encrypted way.

To store sensitive plugin data securely

1. Log in to SPS, navigate to **Policies > Credential Stores** and create a **Local** Credential Store. For details, see "Configuring local Credential Stores" in the Administration Guide.

   Instead of usernames and passwords, you will store the configuration parameters of the plugin in this Credential Store.
2. Add the plugin parameters you want to store in an encrypted way to the Credential Store. You can store any configuration parameter of the plugin in the Credential Store, but note that if an option appears in the Credential Store, the plugin will use it. If the same parameter appears in the configuration of the plugin, it will be ignored.
   - Enter the name of the configuration section without the brackets in the **Host** field (for example, `hashicorp`).
   - Enter the name of the plugin parameter in the **Username** field (for example, `password`).
   - Enter the value of the plugin parameter in the **Passwords** field.
   - Click 

.
3. Navigate to the configuration of the plugin on the **Policies > AA Plugin Configurations** page.
4. In the plugin configuration file, enter the name of the local Credential Store under the `[credential_store]` section as the value of the `name` parameter.
5. Enter `$` as the value of the parameter storing sensitive data.

About us

One Identity solutions eliminate the complexities and time-consuming processes often required to govern identities, manage privileged accounts and control access. Our solutions enhance business agility while addressing your IAM challenges with on-premises, cloud and hybrid environments.

Contacting us

For sales and other inquiries, such as licensing, support, and renewals, visit https://www.oneidentity.com/company/contact-us.aspx.

Technical support resources

Technical support is available to One Identity customers with a valid maintenance contract and customers who have trial versions. You can access the Support Portal at https://support.oneidentity.com/.

The Support Portal provides self-help tools you can use to solve problems quickly and independently, 24 hours a day, 365 days a year. The Support Portal enables you to:

- Submit and manage a Service Request
- View Knowledge Base articles
- Sign up for product notifications
- Download software and technical documentation
- View how-to videos at www.YouTube.com/OneIdentity
- Engage in community discussions
- Chat with support engineers online
- View services to assist you with your product