AppRole Auth Method

The `approle` auth method allows machines or *apps* to authenticate with Vault Vault-defined *roles*. The open design of `AppRole` enables a varied set of workflows and configurations to handle large numbers of apps. This auth method is oriented to automated workflows (machines and services), and is less useful for human operators.

An "AppRole" represents a set of Vault policies and login constraints that must be met to receive a token with those policies. The scope can be as narrow or broad as desired. An AppRole can be created for a particular machine, or even a particular user on that machine, or a service spread across machines. The credentials required for successful login depend upon the constraints set on the AppRole associated with the credentials.

Authentication

Via the CLI

The default path is `/approle`. If this auth method was enabled at a different path, specify `auth/my-path/login` instead.

```
$ vault write auth/approle/login \
    role_id=db02de05-fa39-4855-059b-67221c5c2f63 \
    secret_id=6a174c20-f6de-a53c-74d2-6018fcceff64

Key                Value
---                -----
token              65b74ffd-842c-fd43-1386-f7d7006e520a
token_accessor     3c29bc22-5c72-11a6-f778-2bc8f48cea0e
token_duration     20m0s
token_renewable    true
token_policies     [default]
```

Via the API

The default endpoint is `auth/approle/login`. If this auth method was enabled at a different path, use that value instead of `approle`.

```
$ curl \
    --request POST \
    --data '{"role_id":"988a9df-...","secret_id":"37b74931..."}' \
    http://127.0.0.1:8200/v1/auth/approle/login
```

The response will contain the token at `auth.client_token`:

```
{
  "auth": {
    "renewable": true,
    "lease_duration": 2764800,
    "metadata": {},
    "policies": [
      "default",
      "dev-policy",
      "test-policy"
    ],
    "accessor": "5d7fb475-07cb-4060-c2de-1ca3fcbf0c56",
    "client_token": "98a4c7ab-b1fe-361b-ba0b-e307aacfd587"
  }
}
```

Configuration

Auth methods must be configured in advance before users or machines can authenticate. These steps are usually completed by an operator or configuration management tool.

Via the CLI

1. Enable the AppRole auth method:

   ```
   $ vault auth enable approle
   ```

2. Create a named role:

   ```
   $ vault write auth/approle/role/my-role \
       secret_id_ttl=10m \
       token_num_uses=10 \
       token_ttl=20m \
       token_max_ttl=30m \
       secret_id_num_uses=40
   ```

   For the complete list of configuration options, please see the API documentation.

3. Fetch the RoleID of the AppRole:

   ```
   $ vault read auth/approle/role/my-role/role-id
   role_id     db02de05-fa39-4855-059b-67221c5c2f63
   ```

4. Get a SecretID issued against the AppRole:

   ```
   $ vault write -f auth/approle/role/my-role/secret-id
   secret_id               6a174c20-f6de-a53c-74d2-6018fcceff64
   secret_id_accessor      c454f7e5-996e-7230-6074-6ef26b7bcf86
   ```

Via the API

1. Enable the AppRole auth method:

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data '{"type": "approle"}' \
    http://127.0.0.1:8200/v1/sys/auth/approle
```

2. Create an AppRole with desired set of policies:

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data '{"policies": "dev-policy,test-policy"}' \
    http://127.0.0.1:8200/v1/auth/approle/role/my-role
```

3. Fetch the identifier of the role:

```
$ curl \
    --header "X-Vault-Token: ..." \
    http://127.0.0.1:8200/v1/auth/approle/role/my-role/role-id
```

   The response will look like:

```
{
  "data": {
    "role_id": "988a9dfd-ea69-4a53-6cb6-9d6b86474bba"
  }
}
```

4. Create a new secret identifier under the role:

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
     http://127.0.0.1:8200/v1/auth/approle/role/my-role/secret-id
```

   The response will look like:

```
{
  "data": {
    "secret_id_accessor": "45946873-1d96-a9d4-678c-9229f74386a5",
    "secret_id": "37b74931-c4cd-d49a-9246-ccc62d682a25"
  }
}
```

Credentials/Constraints

RoleID

RoleID is an identifier that selects the AppRole against which the other credentials are evaluated. When authenticating against this auth method's login endpoint, the RoleID is a required argument (via `role_id`) at all times. By default, RoleIDs are unique UUIDs, which allow them to serve as secondary secrets to the other credential information. However, they can be set to particular values to match introspected information by the client (for instance, the client's domain name).

SecretID

SecretID is a credential that is required by default for any login (via `secret_id`) and is intended to always be secret. (For advanced usage, requiring a SecretID can be disabled via an AppRole's `bind_secret_id` parameter, allowing machines with only knowledge of the RoleID, or matching other set constraints, to fetch a token). SecretIDs can be created against an AppRole either via generation of a 128-bit purely random UUID by the role itself (`Pull` mode) or via specific, custom values (`Push` mode). Similarly to tokens, SecretIDs have properties like usage-limit, TTLs and expirations.

Pull And Push SecretID Modes

If the SecretID used for login is fetched from an AppRole, this is operating in Pull mode. If a "custom" SecretID is set against an AppRole by the client, it is referred to as a Push mode. Push mode mimics the behavior of the deprecated App-ID auth method; however, in most cases Pull mode is the better approach. The reason is that Push mode requires some other system to have knowledge of the full set of client credentials (RoleID and SecretID) in order to create the entry, even if these are then distributed via different paths. However, in Pull mode, even though the RoleID must be known in order to distribute it to the client, the SecretID can be kept confidential from all parties except for the final authenticating client by using Response Wrapping (/docs/concepts/response-wrapping.html).

Push mode is available for App-ID workflow compatibility, which in some specific cases is preferable, but in most cases Pull mode is more secure and should be preferred.

Further Constraints

`role_id` is a required credential at the login endpoint. AppRole pointed to by the `role_id` will have constraints set on it. This dictates other `required` credentials for login. The `bind_secret_id` constraint requires `secret_id` to be presented at the login endpoint. Going forward, this auth method can support more constraint parameters to support varied set of Apps. Some constraints will not require a credential, but still enforce constraints for login. For example, `bound_cidr_list` will only allow requests coming from IP addresses belonging to configured CIDR blocks on the AppRole.

API

The AppRole auth method has a full HTTP API. Please see the AppRole API (/api/auth/approle/index.html) for more details.