Azure Authenticator Docs (POC)

This document is intended to be used as a guide for the Azure Authenticator at the POC stage. Configurations and flows are subject to change with the advancement of this feature. At current, we support applications running in Azure VMs and only *1* DAP *Follower* running in Azure.

The DAP Azure Authenticator enables Azure applications to authenticate to DAP according to Azure properties and retrieve a DAP access token in order retrieve secrets stored in DAP. With this, Azure instances can leverage their Azure identities to authenticate with DAP, in-place of their DAP ones.

How it works

1. Application that is deployed in an Azure VM can send authentication request to DAP with its Azure access token in the request body and a DAP Host ID in the URL
2. DAP verifies that the given access token is issued by the Active Directory (AD) Provider defined in the Azure Authenticator policy and that the token is valid
3. DAP validates the identity of the Azure access token properties against the application identity defined in the DAP Host annotations
4. Application receives a DAP access token and can fetch secrets stored in DAP

Configure the Azure Authenticator

In order to communicate with DAP and fetch secrets stored in DAP, the Azure application needs to authenticate to DAP and receive DAP access token. This section describes how properly to configure the Azure Authenticator and define the entities that can authenticate to DAP using the Azure Authenticator.

Define Azure Authenticator Policy

In order to begin using the Azure Authenticator, you must define it in policy and detail a group of permissioned hosts that will be able to use the Azure Authenticator to authenticate with DAP

All Azure Authenticator policy IDs must have the prefix **conjur/authn-azure**

A DAP server can have multiple instances of the same **conjur/authn-azure** authenticator type. To differentiate between each webservice, append a **servic e ID** to the authenticator type.

1. Copy the following policy and, in the policy's ID, append the selected service ID.

```
- !policy
  id: conjur/authn-azure/<service-id>
  body:
  - !webservice

  - !variable
    id: provider-uri

  - !group apps
    annotations:
      description: Group of hosts who can authenticate using the authn-azure/<service-id> authenticator

  - !permit
    role: !group apps
    privilege: [ read, authenticate ]
    resource: !webservice
```

   Provide the following:

Term	Explanation	Example value
Service ID	The ID of your Azure Authenticator webservice	prod

2. Save the policy as a .yml file using the following file naming convention: **authn-azure-<*service-id*>.yml**
3. Load the policy file into root: **conjur policy load root authn-azure-<*service-id*>.yml**
4. Give the `provider-uri` a value by running the following in the CLI

```
$ conjur variable values add conjur/authn-azure/<service_id>/provider-uri <provider_uri>
```

   Provide the following:

Term	Explanation	Example value
provider URI	URI of the Azure Provider	`https://sts.windows.net/<tenant_id>`

Define Azure resource as a Host in Policy

Now that the Azure Authenticator has been configured, we must now give the Azure instance a DAP identity and define the application identity for each Azure instance in DAP in the Host annotations. When defining the DAP Host, the required application identity annotations to represent the Azure resource include `subscription-id` and `resource-group`. Additionally, you can add an extra level of granularity by providing the user or system assigned identity for the Azure instance.

If the Azure resource is assigned a *user-assigned-identity*, provide the *user assigned identity* defined for the resource. A host with just `subscription-id` and `resource group` or with the addition of a user assigned identity in its annotations can be used for several Azure VMs. If you prefer to use the system-assigned-identity for the Azure instance, in the host annotations provide its *Object ID.*

NOTE: These fields can be retrieved from the Azure portal.

NOTE: At this time, you will not be able to provide **both** assigned-identities for the same host.

1. Copy the following host policy

```
- !policy
  id: azure-apps
  body:
    - !group

    - &hosts
      - !host
        id: azureVM
        annotations:
          authn-azure/subscription-id: <subscription_id>
          authn-azure/resource-group: <resource_group>

    - !grant
      role: !group
      members: *hosts

- !grant
  role: !group conjur/authn-azure/<service-id>/apps
  member: !group azure-apps
```

   If you would prefer to use a *user assigned identity* for your Azure resource, just add `authn-azure/user-assigned-identity: <user assigned identity>`.

   If instead you would like to provide the system assigned identity for the Azure resource, add `authn-azure/system-assigned-identity: <Object ID>`.
2. Save the policy as a .yml file, **authn-azure-<*service-id*>-hosts.yml**
3. Load the policy file into root: **conjur policy load root authn-azure-<*service-id*>-hosts.yml**

Provide the following:

NOTE: These fields can be retrieved from the Azure portal.

Term	Explanation
Subscription ID	An agreement with Microsoft to be able to use Microsoft services.
Resource group	An accumulation of related Azure resources that are gathered under a group

Object ID (***Optional*** and dependent on assigned-identity)	An ID that identities an object when using a system-assigned-identity
user assigned identity (***Optional*** and dependent on assigned-identity)	A readable name given to an Azure resource upon enabling a user-assigned-identity

Define secrets and access for Azure instances

Detail the group of hosts that will be able to access your DAP secrets and give that group the proper privileges.

An example policy snippet is provided below where all members of the `consumers` group are granted permissions on the `test-variable` DAP secret.

```
- !variable test-variable


- !group
  id: consumers

- !permit
   role: !group consumers
   privilege: [ read, execute ]
   resource: !variable test-variable

- !grant
  role: !group consumers
  member: !group azure-apps
```

Enable the Azure Authenticator

Now that we have configured the Azure Authenticator and Hosts that will authenticate to DAP using the Azure Authenticator, we need to now enable the Azure Authenticator in the DAP cluster. You can do so by enabling it in the DAP master which will automatically cascade to all DAP Followers. Before doing so, retrieve the currently enabled authenticator and append the Azure authenticator to it.

Run the following in the DAP master:

```
$ evoke variable list CONJUR_AUTHENTICATORS
```

Then, append "authn-azure/<service-id>" to the list and run:

```
$ evoke variable set CONJUR_AUTHENTICATORS <comma-separated-list>
```

Provide the following:

Term	Explanation	Example value
comma-separate-list	All authenticators you would like to be enabled	authn,authn-azure/prod

For example, If

```
$ evoke variable list CONJUR_AUTHENTICATORS
```

has the output

```
$ CONJUR_AUTHENTICATORS=authn,authn-oidc/okta
```

You should run

```
$ evoke variable set CONJUR_AUTHENTICATORS=authn,authn-oidc/okta,authn-azure/<service-id>
```

Retrieve an Azure access token

To retrieve an Azure token for authenticating with DAP, log into the Azure VM and run the following command:

Token type	Command
system-assigned ID	curl http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https%3A%2F%2Fmanagement.azure.com%2F -H Metadata:true -s | jq -r '.access_token'
user-assigned ID	curl http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&client_id=**<client_id>**&resource=https%3A%2F%2Fmanagement.azure.com%2F -H Metadata:true -s | jq -r '.access_token' Where <client_id> can be found in the Managed Identity page in the portal

Note: Both tokens can be used when authenticating a host that has only the subscription-id and the resource-group in its Application Identity.

Azure Authenticator REST API

Once the Azure Authenticator is configured, your Azure instance can send an authentication request.

The request should be defined as follows:

URI

```
POST https://<DAP-server-hostname>/authn-azure/<service-id>/<account>/<host_id>/authenticate
```

Request

Header	`Content-Type: application/x-www-form-urlencoded`
Body	The body must include the Azure access token for Azure instance `jwt: "eyJhbGciOiJSUzI1NiIs......uTonCA"`

URI parameters

Term	Explanation
`DAP-server-hostname`	Server URL for DAP master
`service-id`	The ID of your Azure Authenticator webservice
`account`	The organization account name
`host-id`	The DAP identity for the Azure instance defined in the `**Define Azure Authenticator Policy**` section. This value should be the full hostname and should include url encoding for `/`. In our example: `host%2Fazure-apps%2FazureVM`

Example REST request

```
POST https://ec2-0-000-000-00.eu-central-1.compute.amazonaws.com:8443/authn-azure/prod/cucumber/host%2Fazure-
apps%2FazureVM/authenticate
```

Troubleshooting the Azure Authenticator

Unable to authenticate using Azure Authenticator?

We have developed an Authenticator Status feature in order to be able to check the status of Azure Authenticator. Using this tool, you will be able to identify if the authenticator configuration was successful and if not, the reasons for failure.

Limitations

We **do not** officially support DAP running inside of Azure

We only officially support Azure VMs and **not** Azure Functions, App Services, etc.