StarlingX can be configured to use a remote Windows Active Directory server to authenticate users of the Kubernetes API, through the oidc-auth-apps application.

The Kubernetes cluster must first be configured with a few extra parameters for the kube-apiserver. This is done by adding the following to localhost.yml during bootstrap:

apiserver\_oidc:

client\_id: stx-oidc-client-app

issuer\_url: https://<oam-floating-ip>:30556/dex

username\_claim: email

groups\_claim: group

The values of the username\_claim and groups\_claim parameters could vary for different user and group configurations within your Windows Active Directory server.

Note that for now, the only way to specify these parameters is through bootstrap. Please make sure to enter the correct values here or a reinstall would be required.

*We probably need to put a short note about this in our Installation Guide, in the bootstrap section, to prevent people from forgetting this if required.*

The oidc-auth-apps application is a system managed application that is packaged in the ISO and uploaded by default. To use the oidc-auth-apps application for authentication, the user must first configure and deploy the oidc-auth-apps application.

Procedure:

1) create a secret with the certificate and key (local-dex.tls) to be used by the oidc-auth-apps as well as a secret with the CA that signed this certificate (dex-client-secret) for the client. The certificate should be signed by a CA trusted by the system. If the certificate is signed by a CA that is not trusted by default, the user can make the system trust the CA by specifying it during bootstrap by specifying ssl\_ca\_cert in localhost.yml, or through "system certificate-install -m ssl\_ca ..." after bootstrap.  
  
( the commands below assume the cert and key pem files for creating these secrets are in /home/sysadmin/ssl/ )

kubectl create secret tls local-dex.tls --cert=ssl/dex-cert.pem --key=ssl/dex-key.pem -n kube-system

kubectl create secret generic dex-client-secret --from-file=/home/sysadmin/ssl/dex-ca.pem  -n kube-system

2) specify user overrides for oidc-auth-apps.

system helm-override-update oidc-auth-apps dex kube-system --values /home/sysadmin/dex-overrides.yaml

The only mandatory section is the "connectors" section. This will vary for different Windows Active Directory deployments. Refer to the upstream dex documentation for this:

https://github.com/dexidp/dex/blob/master/Documentation/connectors/ldap.md

An example dex-overrides.yaml file:

config:

connectors:

- type: ldap

name: OpenLDAP

id: ldap

config:

host: <windows-active-directory-server-IP>:389

insecureNoSSL: true

insecureSkipVerify: true

bindDN: cn=Administrator,cn=Users,dc=corp,dc=starlingx,dc=com

bindPW: <windows-active-diretory-administrator-password>

usernamePrompt: Username

userSearch:

baseDN: ou=Users,ou=Sysadmin,dc=corp,dc=starlingx,dc=com

filter: "(objectClass=user)"

username: sAMAccountName

idAttr: sAMAccountName

emailAttr: sAMAccountName

nameAttr: displayName

3) apply the application:

system application-apply oidc-auth-apps

Using the setup

Assuming there is a user called "testuser" set up in the Windows Active Directory deployments

On StarlingX, bind Kubernetes RBAC role(s) to this user; for example, give this user admin privileges by creating the following deployment file and deploy with "kubectl apply -f filename"

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: testuser-rolebinding

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- apiGroup: rbac.authorization.k8s.io

kind: User

name: testuser

Set up a kubernetes user to authenticate through dex; this can be done locally on controller-0 or remotely on a workstation

# setup cluster if you haven’t already  
 kubectl config set-cluster mystxcluster –server=https://<oam-floating-ip>:6443

kubectl config set-context testuser@mystxcluster --cluster=mystxcluster --user=testuser  
  
 # put the cert for the CA that signed the dex certificate to ./ssl/dex-ca.pem

kubectl config set-credentials testuser \  
 --auth-provider=oidc \  
 --auth-provider-arg=idp-issuer-url=https://<oam-floating-ip>:30556/dex \  
 --auth-provider-arg=client-id=stx-oidc-client-app \  
 --auth-provider-arg=client-secret=St8rlingX\* \  
 --auth-provider-arg=idp-certificate-authority-data=$(base64 -w 0 ./ssl/dex-ca.pem) \  
 --auth-provider-arg=extra-scopes="offline\_access openid profile email groups"

**Obtaining Authentication Token**

The authentication token can be obtained using the oidc-auth-apps OIDC client web interface using a browser or by using the oidc-auth shell script.

**Using the oidc-auth shell script**

The oidc-auth script retrieves the ID token from Windows Active Directory using the OIDC client, and dex, and updates the Kubernetes credential for the user in the kubectl config file.

* On controller-0, oidc-auth is installed as part of the base installation, and ready to use.
* On a remote workstation using remote-cli container, oidc-auth is installed within the remote cli container, and ready to use. See Remote CLI section for details.
* On a remote host, using directly installed kubectl and helm. The following setup is required:
  + Install “Python Mechanize” module using the following command:

# sudo pip2 install mechanize

* + Get oidc-auth script from CENGN StarlingX mirror

Procedure

Run oidc-auth script in order to authenticate and update user credentials in kubectl config file with the retrieved token.

~(keystone\_admin)$ oidc-auth -c <OAM ip address> -u testuser

Password:

Login succeeded.

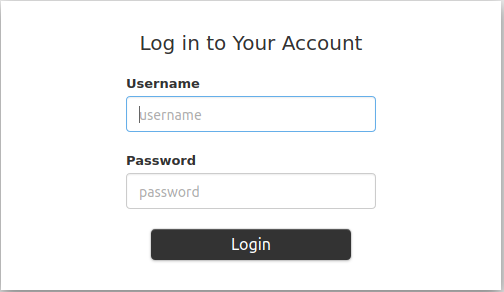
Updating kubectl config ...

User testuser set.

**Obtain authentication token via browser**

1. From a browser:

http://*oam-floating-ip-address*:*30555*



Enter username, password and click Login.

The ID token will be displayed as follows:

ID Token:

eyJhbGciOiJSUzI1NiIsImtpZCI6IjkwYTcyYmIwZTRjNTJhZDhiNGYxMmYxNzc3NTVmNDdmODc5M2ZkYTAifQ.eyJpc3MiOiJodHRwczovLzEwLjEwLjEwLjM6MzA1NTYvZGV4Iiwic3ViIjoiQ2dkbmQyRnBibVZ6RWdSc1pHRnciLCJhdWQiOiJzdHgtb2lkYy1jbGllbnQtYXBwIiwiZXhwIjoxNTgwODQ4NTkzLCJpYXQiOjE1ODA3NjIxOTMsImF0X2hhc2giOiJNU0YtNDBpOWVuM1QyVjdUMWdSZW5RIiwiZW1haWwiOiJnd2FpbmVzIiwiZW1haWxfdmVyaWZpZWQiOnRydWUsIm5hbWUiOiJHcmVnb3J5IEEuIFdhaW5lcyJ9.oNIabUhd5wx3tFCIuewtzsbYfx1OsrGXtEUEPL0l5Y944WE2c1HP6YUHWxvYTMw1\_Ldl-jx-koiYbiE8Eztgy9anfJqclUFa6xlxP666Z7AYxndsULylqzfT0dvySaddIEEYDffx7aH6g7q2PKZjMHFierRyqmCu8WTPRSNy3NymLmQaGGjUmFHqbvpEBgg\_ytpsDgbRIpk1EbyP63l79hBNlRvcffTRLi3LYYRaJLgSbx2tha43OX5rKxylF\_GrzZHaqxxT6MjIHKHagUrcqa054RwPWUHKyV26ErkMg6gN5uyMm462UtnW7jJucYrWBpbaWaj0U0OTWv\_1NnKlJw

Access Token:

jwcj46v3vmumpixr54wbyrstf

Claims:

{

  "iss": "<https://10.10.10.3:30556/dex>",

  "sub": "Cgdnd2FpbmVzEgRsZGFw",

  "aud": "stx-oidc-client-app",

  "exp": 1580848593,

  "iat": 1580762193,

  "at\_hash": "MSF-40i9en3T2V7T1gRenQ",

  "email": "user",

  "email\_verified": true,

  "name": "Username"

}

1. Set Kubernetes credential with the above ID token

~(keystone\_admin)]$ TOKEN=<ID token string>

~(keystone\_admin)]$ kubectl config setcredentials testuser --token $TOKEN

1. Switch to the context for this user

~(keystone\_admin)]$ kubectl config use-context testuser@mystxcluster

run "kubectl get pods --all-namespaces", this should be successful because we are now authenticated.