Zeppelin authentication

Zeppelin uses Shiro as the authentication framework. Shiro provides many authentication methods:

- File
- Ldap/AD
- PAM
- Knox SSO

To active one of the above authentication method, you need to edit the **/usr/hdp/current/zeppelin-server/conf/shiro.ini**

The shiro.ini has four main sections

- [users] : login and password for file authentication
- [main] : configuration for ldap/ad authentication
- [roles] : Define roles
- [urls] : Define authorization policy for the defined roles

1. File authentication

Edit the [users] section of shiro.ini file

```
# general form
<user_name> = <user_pwd>, role
# example of list of users with their password allowed to access Zeppelin.
admin =
$shiro1$SHA-256$500000$XwFNnp9qlwtKC3lObKqGQQ==$0z8kQBHgyLIX4NLlIWC1oSOaF6e7
1nho9QRhZedGeTM=, admin
pliu =
$shiro1$SHA-256$500000$0qkTih9/Wph2rRCOrK12mA==$Dx7PnsW2FNjg8Nsk0OXCBkuyuySc
wHqLT/xE79kBTGA=, admin
jdarmont =
$shiro1$SHA-256$500000$O7qXLbAUkKcc1r2tilJprQ==$es19ME5urGPMVBqBc6L2etDsVwW6
FXl9hwQi5U4IPZ8=, admin
sloudcher =
$shiro1$SHA-256$500000$W/YhSJxuB+t+wfguBALlyg==$Tp5wmGEDYgOdS8cnC8t+zLEUDfUM
3rgxMu6AUhZlmVU=, admin
```

The hash of the password is generated by using the shiro-tools-hasher.jar which can be downloaded from https://repo1.maven.org/maven2/org/apache/shiro/tools/shiro-tools-hasher/1.4.0/

```
# downlad the jar file and run the following command
java -jar shiro-tools-hasher-1.3.2.jar -p

# type your password here
Password to hash:
Password to hash (confirm):
```

```
$shiro1$SHA-256$500000$W/YhSJxuB+t+wfguBALlyg==$Tp5wmGEDYgOdS8cnC8t+zLEUDfUM3rgxMu6AUhZlmVU=
```

Note: Shiro only accepts a salted hash of a password.

2. LDAP authentication

Shiro provides two possible way to connect to an OpenLDAP server

- LdapGroupRealm
- LdapRealm

2.1 LdapGroupRealm config

The ldapGroupRealm is very easy to configure, but it has limited flexibility with mapping of ldap groups to users and for authorization for user groups. Below is a config example

```
[main]
ldapRealm = org.apache.zeppelin.realm.LdapGroupRealm
# search base for ldap groups (only relevant for LdapGroupRealm):
ldapRealm.contextFactory.environment[ldap.searchBase] = dc=udl,dc=org
ldapRealm.contextFactory.url = ldap://lin01.udl.org:389
ldapRealm.userDnTemplate = uid={0},ou=Users,dc=udl,dc=org
ldapRealm.contextFactory.authenticationMechanism = simple
```

2.2 LdapRealm config

The LdapRealm allows for mapping of ldapgroups to roles and also allows for role/group based authentication into the zeppelin server. Sample configuration for this realm is given below.

```
[main]
ldapRealm=org.apache.zeppelin.realm.LdapRealm

ldapRealm.contextFactory.authenticationMechanism=simple
ldapRealm.contextFactory.url=ldap://lin01.udl.org:389
ldapRealm.userDnTemplate=uid={0},ou=Users,dc=udl,dc=org
# Ability to set ldap paging Size if needed default is 100
ldapRealm.pagingSize = 200
ldapRealm.authorizationEnabled=true
ldapRealm.contextFactory.systemAuthenticationMechanism=simple
ldapRealm.searchBase= dc=udl,dc=org
ldapRealm.userSearchBase = dc=udl,dc=org
ldapRealm.groupSearchBase = ou=groups,dc=udl,dc=org
ldapRealm.groupObjectClass=groupofnames
# Allow userSearchAttribute to be customized
ldapRealm.userSearchAttributeName = sAMAccountName
```

```
ldapRealm.memberAttribute=member
# force usernames returned from ldap to lowercase useful for AD
ldapRealm.userLowerCase = true
# ability set searchScopes subtree (default), one, base
ldapRealm.userSearchScope = subtree;
ldapRealm.groupSearchScope = subtree;
ldapRealm.memberAttributeValueTemplate=cn={0},ou=Users,dc=udl,dc=org
ldapRealm.contextFactory.systemUsername=uid=guest,ou=Users,dc=udl,dc=org
ldapRealm.contextFactory.systemPassword=S{ALIAS=ldcSystemPassword}
# enable support for nested groups using the LDAP_MATCHING_RULE_IN_CHAIN
operator
ldapRealm.groupSearchEnableMatchingRuleInChain = true
# optional mapping from physical groups to logical application roles
ldapRealm.rolesByGroup = LDN_USERS: user_role, NYK_USERS: user_role,
HKG_USERS: user_role, GLOBAL_ADMIN: admin_role
# optional list of roles that are allowed to authenticate. Incase not
present all groups are allowed to authenticate (login).
# This changes nothing for url specific permissions that will continue to
work as specified in [urls].
ldapRealm.allowedRolesForAuthentication = admin_role,user_role
ldapRealm.permissionsByRole= user_role = *:ToDoItemsJdo:*:*, *:ToDoItem:*:*;
admin_role = *
securityManager.sessionManager = $sessionManager
securityManager.realms = $ldapRealm
```

3. Apache Directory

```
activeDirectoryRealm = org.apache.zeppelin.realm.ActiveDirectoryGroupRealm
activeDirectoryRealm.systemUsername = userNameA
activeDirectoryRealm.systemPassword = passwordA
activeDirectoryRealm.hadoopSecurityCredentialPath =
jceks://file/user/zeppelin/conf/zeppelin.jceks
activeDirectoryRealm.searchBase = CN=Users,DC=SOME_GROUP,DC=COMPANY,DC=COM
activeDirectoryRealm.url = ldap://ldap.test.com:389
activeDirectoryRealm.groupRolesMap =
"CN=aGroupName,OU=groups,DC=SOME_GROUP,DC=COMPANY,DC=COM":"group1"
activeDirectoryRealm.authorizationCachingEnabled = false
activeDirectoryRealm.principalSuffix = @corp.company.net
```

Also instead of specifying systemPassword in clear text in shiro.ini administrator can choose to specify the same in “hadoop credential”. Create a keystore file using the hadoop credential commandline, for this the hadoop commons should be in the classpath

```
hadoop credential create activeDirectoryRealm.systempassword -provider
jceks://file/user/zeppelin/conf/zeppelin.jceks
```

Change the following values in the Shiro.ini file, and uncomment the line:

```
activeDirectoryRealm.hadoopSecurityCredentialPath =
```

```
jceks://file/user/zeppelin/conf/zeppelin.jceks
```

4. PAM

PAM authentication support allows the reuse of existing authentication moduls on the host where Zeppelin is running. On a typical system modules are configured per service for example sshd, passwd, etc. under /etc/pam.d/. You can either reuse one of these services or create your own for Zeppelin. Activiting PAM authentication requires two parameters: 1. realm: The Shiro realm being used 2. service: The service configured under /etc/pam.d/ to be used. The name here needs to be the same as the file name under /etc/pam.d/

```
[main]
 pamRealm=org.apache.zeppelin.realm.PamRealm
 pamRealm.service=sshd
```

5. Knox SSO

KnoxSSO provides an abstraction for integrating any number of authentication systems and SSO solutions and enables participating web applications to scale to those solutions more easily. Without the token exchange capabilities offered by KnoxSSO each component UI would need to integrate with each desired solution on its own.

To enable this, apply the following change in **conf/shiro.ini under [main] section**.

```
### A sample for configuring Knox JWT Realm
knoxJwtRealm = org.apache.zeppelin.realm.jwt.KnoxJwtRealm
## Domain of Knox SSO
knoxJwtRealm.providerUrl = https://domain.example.com/
## Url for login
knoxJwtRealm.login = gateway/knoxsso/knoxauth/login.html
## Url for logout
knoxJwtRealm.logout = gateway/knoxssout/api/v1/webssout
knoxJwtRealm.redirectParam = originalUrl
knoxJwtRealm.cookieName = hadoop-jwt
knoxJwtRealm.publicKeyPath = /etc/zeppelin/conf/knox-sso.pem
knoxJwtRealm.groupPrincipalMapping = group.principal.mapping
knoxJwtRealm.principalMapping = principal.mapping
# This is required if KNOX SSO is enabled, to check if
"knoxJwtRealm.cookieName" cookie was expired/deleted.
authc = org.apache.zeppelin.realm.jwt.KnoxAuthenticationFilter
```

6. Authorizations

6.1 Role definition

If you use file authentication, you can define different roles here and associate them with user login in the [user] section

```
[roles]
role1 = *
role2 = *
role3 = *
admin = *
```

If you use Ldap/AD, the roles are imported from the LDAP/AD server.

6.2 Policy rules

By default, anyone who defined in [users] can share Interpreter Setting, Credential, and Configuration information in Apache Zeppelin. Sometimes you might want to hide this information for your use case. Since Shiro provides URL-based security, you can hide the information by commenting or uncommenting these below lines in conf/shiro.ini.

```
[urls]
/api/interpreter/** = authc, roles[admin]
/api/configurations/** = authc, roles[admin]
/api/credential/** = authc, roles[admin]
```

The above config means only users with admin role can see Interpreter Setting, Credential and Configuration information. If you want to grant this permission to other users, you can change roles[ ] as you defined at [users] section.

Note: authc means user must be authenticated.

6.2.1 Multiple roles for one policy rule

```
[urls]
/api/interpreter/** = authc, roles[admin, role1]
```

Here, user with admin or role1 role can see interpreter settings.

6.2.2 Mix role and users

Sometime you just want to authorize one use not a group of users. You can do the following

```
# active user authorization
[main]
anyofrolesuser = org.apache.zeppelin.utils.AnyOfRolesUserAuthorizationFilter
```

```
[urls]

/api/interpreter/** = authc, anyofrolesuser[admin, user1]
```