Usage of custom private key for JWT Client Authentication

TOC

Abstract

UAA supports JWT client authentication for outbound communication according to OIDC standard, see https://openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication, method private_key_jwt.

The creation of a JWT needs a private key. UAA supports to maintain private keys already. The next section describes the current state for this.

This scenario is typically used where UAA acts as proxy but the authentication is done in another OIDC complaint provider.

Current support of referencing a private key

The connection to another OIDC provider can be setup with configurations like

a) uaa.yml
b) REST API , e.g. https://docs.cloudfoundry.org/api/uaa/version/77.2.0/index.html#oauth-oidc

Examples

a)

```yaml
oauth:
  providers:
    oidc.proxy:
      type: oidc1.0
      passwordGrantEnabled: true
      discoveryUrl: https://awstest.accounts400.ondemand.com/.well-known/openid-configuration
      issuer: https://awstest.accounts400.ondemand.com
      scopes:
        - openid
        - email
        - profile
      linkText: Login with OIDC-Proxy
      relyingPartyId: e9f2ac13-e1a9-44fd-ba09-b9ce950dd20e
      groupMappingMode: AS_SCOPES
      jwtclientAuthentication:
        alg: RS512
        kid: xxxx
      attributeMappings:
        user_name: email
        external_groups:
          roles
      showLinkText: true
```

The section jwtClientAuthentication in uaa.yml can contain detailed configuration for the creation of the JWT. The sub elements under jwtClientAuthentication allow to have different settings in the JWT according to the standard settings. Standard settings for the client JWT are

- Option alg, defaults to RS256
- Option kid, detaults to activeKeyId setting in uaa.yml
- Option iss, defaults to relyingPartyId setting in oauth provider entry
- Option aud, defaults to tokenUrl setting in oauth provider entry

The sub element kid is helpful to have a different key for JWT Client Authentication compared to the active Key for JWT signing. The key separation allows to rotate these keys with a different schedule.

Future needs for private key

There are 2 major requirements for the JWT creation.

I: Custom private

There are situations, where the creation of the client JWT should be done based in custom/own private key. This key should be not part of the token keys of a UAA zone. The reason for it is, that keys from a zone are published with the /token_keys endpoint.

Therefore, the proposal is, that there are 2 new sub elements for jwtClientAuthentication

- key
- cert

II: No DB storage

The private key for JWT client authentication should not be persistent in the DB, e.g. table identity_provider, column config. There are 2 reasons for this:

1. UAA DB tables are not protected with encryption, a private key is visible there for all DB admins, or in UAA backups
2. There could be many (> 10.000) IDPs which use the same private key for the OIDC proxy. A key rotation would update many (>10.000) entries in DB.

Therefore there is a proposal for a YAML reference mechanism

```yaml
oauth:
  providers:
    oidc.proxy:
      type: oidc1.0
      passwordGrantEnabled: true
      discoveryUrl: https://awstest.accounts400.ondemand.com/.well-known/openid-configuration
      issuer: https://awstest.accounts400.ondemand.com
      scopes:
        - openid
        - email
        - profile
      linkText: Login with OIDC-Proxy
      relyingPartyId: e9f2ac13-e1a9-44fd-ba09-b9ce950dd20e
      groupMappingMode: AS_SCOPES
      jwtclientAuthentication:
        alg: RS512
        key: ${default.jwt.client.key}
        cert: ${default.jwt.client.cert}
      attributeMappings:
        user_name: email
        external_groups:
          roles
      showLinkText: true
```

The reference points to another section in the uaa.yml, but many entries for oauth providers can point the same section in uaa.yml and in the DB of UAA the configuration will have an entry like

```
{
    "emailDomain": null,
    "additionalConfiguration": null,
    "providerDescription": null,
    "externalGroupsWhitelist": [],
    "attributeMappings": {
        "external_groups": "roles",
        "user_name": "email"
    },
    "addShadowUserOnLogin": true,
    "storeCustomAttributes": true,
    "authUrl": null,
    "tokenUrl": null,
    "tokenKeyUrl": null,
    "tokenKey": null,
    "userInfoUrl": null,
    "logoutUrl": null,
    "linkText": "Login with OIDC-Proxy",
    "showLinkText": true,
    "clientAuthInBody": false,
    "skipSslValidation": false,
    "relyingPartyId": "e9f2ac13-e1a9-44fd-ba09-b9ce950dd20e",
    "scopes": [
        "openid",
        "email",
        "profile"
    ],
    "issuer": "https://awstest.accounts400.ondemand.com",
    "responseType": "code",
    "userPropagationParameter": null,
    "groupMappingMode": "AS_SCOPES",
    "pkce": true,
    "performRpInitiatedLogout": true,
    "discoveryUrl": "https://awstest.accounts400.ondemand.com/.well-known/openid-configuration",
    "passwordGrantEnabled": true,
    "setForwardHeader": false,
    "jwtClientAuthentication": {
        "alg": "RS512",
        "key": "${default.jwt.client.key}",
        "cert": "${default.jwt.client.cert}"
    },
    "additionalAuthzParameters": {}
}
```


The PR https://github.com/cloudfoundry/uaa/pull/2771 is a proposal for the requirements.