2. TechEx2019 - Trust Management

Trust Management & OP configuration

Section Topics

- SAML IdP Metadata vs. OpenID Provider Configuration
- OIDC Client Information
- Trust relationship establishment between Shibboleth OP and test RP

SAML IdP Metadata vs. OpenID Provider Configuration

SAML IdP Metadata

- Example IdP metadata: https://idp.csc.fi/idp/shibboleth
  - By default, the contents are fetched from /opt/shibboleth-idp/metadata/idp-metadata.xml
  - The file is generated by the installation script and is **NOT** updated afterwards automatically
- Example federation metadata: https://haka.funet.fi/metadata/haka-metadata.xml
  - Signed by the federation operator (trusted 3rd party)

OpenID Provider Configuration

- OpenID Connect Discovery spec - responds to two questions
  1. How the OP configuration is found?
     - https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderConfig
       - Webfinger protocol exploited for finding out the issuer name from the resource identifier
       - URL format: **<issuer>/.well-known/openid-configuration**
  2. What are the contents of the OP configuration?
     - https://openid.net/specs/openid-connect-discovery-1_0.html#ProviderMetadata
- Two ways to publish OP configuration in Shibboleth OP: https://github.com/CSCfi/shibboleth-idp-oidc-extension/wiki/DiscoveryAndOPConfiguration#openid-provider-metadata
  1. Static file - example found at */opt/shibboleth-idp/static/.well-known/openid-configuration*
  2. Static file with dynamic claims - example found at */opt/shibboleth-idp/flows/oidc/discovery* (URL-endpoint *../idp/profile/oidc/discovery*)

Exercises

Exercise 2.1 - Google's OP configuration

Google OP issuer name is 'https://accounts.google.com'

1. What is the endpoint URL for the openid-configuration?

   Hints, Tips and Result

   ```
   https://accounts.google.com/.well-known/openid-configuration
   ```

2. Check the contents of Google's openid-configuration

   Hints, Tips and Result


```
{
 "issuer": "https://accounts.google.com",
 "authorization_endpoint": "https://accounts.google.com/o/oauth2/v2/auth",
 "token_endpoint": "https://oauth2.googleapis.com/token",
 "userinfo_endpoint": "https://www.googleapis.com/oauth2/v3/userinfo",
 "revocation_endpoint": "https://oauth2.googleapis.com/revoke",
 "jwks_uri": "https://www.googleapis.com/oauth2/v3/certs",
 "response_types_supported": [
  "code",
  "token",
  "id_token",
  "code token",
  "code id_token",
  "token id_token",
  "code token id_token",
  "none"
 ],
 "subject_types_supported": [
  "public"
 ],
 "id_token_signing_alg_values_supported": [
  "RS256"
 ],
 "scopes_supported": [
  "openid",
  "email",
  "profile"
 ],
 "token_endpoint_auth_methods_supported": [
  "client_secret_post",
  "client_secret_basic"
 ],
 "claims_supported": [
  "aud",
  "email",
  "email_verified",
  "exp",
  "family_name",
  "given_name",
  "iat",
  "iss",
  "locale",
  "name",
  "picture",
  "sub"
 ],
 "code_challenge_methods_supported": [
  "plain",
  "S256"
 ]
}
```

Exercise 2.2 - Shibboleth OP Configuration

Everybody has a Shibboleth OP instance running on a virtual machine with public IP. The OP issuer name is https://IP_ADDRESS

- What is the endpoint URL for the openid-configuration?

  Hints, Tips and Result

  ```
  https://IP_ADDRESS/.well-known/openid-configuration
  ```

- What is the contents of the well-known endpoint?

  Hints, Tips and Result


```
{
   "issuer":"https://192.168.0.150",
   "authorization_endpoint":"https://192.168.0.150/idp/profile/oidc/authorize",
   "registration_endpoint":"https://192.168.0.150/idp/profile/oidc/register",
   "token_endpoint":"https://192.168.0.150/idp/profile/oidc/token",
   "userinfo_endpoint":"https://192.168.0.150/idp/profile/oidc/userinfo",
   "jwks_uri":"https://192.168.0.150/oidc/keyset.jwk",
   "response_types_supported":[
      "code",
      "id_token",
      "token id_token",
      "code id_token",
      "code token",
      "code token id_token"
   ],
   "subject_types_supported":[
      "public",
      "pairwise"
   ],
   "grant_types_supported":[
      "authorization_code",
      "implicit",
      "refresh_token"
   ],
   "id_token_encryption_alg_values_supported":[
      "RSA1_5"
   ],
   "id_token_encryption_enc_values_supported":[
      "A128CBC-HS256"
   ],
   "id_token_signing_alg_values_supported":[
      "RS256",
      "RS384",
      "RS512",
      "HS256",
      "HS384",
      "HS512",
      "ES256"
   ],
   "userinfo_encryption_alg_values_supported":[
      "RSA1_5"
   ],
   "userinfo_encryption_enc_values_supported":[
      "A128CBC-HS256"
   ],
   "userinfo_signing_alg_values_supported":[
      "RS256",
      "RS384",
      "RS512",
      "HS256",
      "HS384",
      "HS512",
      "ES256"
   ],
```

```
   "request_object_signing_alg_values_supported":[
      "none",
      "RS256",
      "RS384",
      "RS512",
      "HS256",
      "HS384",
      "HS512",
      "ES256",
      "ES384",
      "ES512"
   ],
   "token_endpoint_auth_methods_supported":[
      "client_secret_basic",
      "client_secret_post",
      "client_secret_jwt",
      "private_key_jwt"
   ],
   "claims_parameter_supported":true,
   "request_parameter_supported":true,
   "request_uri_parameter_supported":false,
   "require_request_uri_registration":false,
   "display_values_supported":[
      "page"
   ],
   "scopes_supported":[
      "openid",
      "profile",
      "email",
      "address",
      "phone",
      "offline_access"
   ],
   "response_modes_supported":[
      "query",
      "fragment",
      "form_post"
   ],
   "claims_supported":[
      "aud",
      "iss",
      "sub",
      "iat",
      "exp",
      "acr",
      "auth_time",
      "email",
      "email_verified",
      "address",
      "phone",
      "phone_number_verified",
      "name",
      "family_name",
      "given_name",
      "middle_name",
      "nickname",
      "preferred_username",
      "profile",
      "picture",
      "website",
      "gender",
      "birthdate",
      "zoneinfo",
      "locale",
      "updated_at"
   ]
}
```

OIDC Client Metadata

OIDC Client Information

Static configuration example from Google:

- https://developers.google.com/identity/protocols/OpenIDConnect

Dynamic registration:

- https://openid.net/specs/openid-connect-registration-1_0.html#ClientMetadata

There's no direct equivalent to SAML SP Metadata (or EntityDescriptor/EntitiesDescriptor) on OIDC

- Similar federation metadata signing logic cannot be directly applied to OIDC
- See OIDC Federation draft
  - https://openid.net/specs/openid-connect-federation-1_0.html
  - Applies to OP discovery + dynamic client registration phases - different from the SAML R&E model

Trust relationship establishment between Shibboleth OP and test RP

OIDC Client Information

https://github.com/CSCfi/shibboleth-idp-oidc-extension/wiki/MetadataConfiguration

Exercises

Exercise 2.3 - Add a trusted RP

- Try OIDC flow with an RP that is not (yet) trusted
  - The OIDC sequence can be started with https://IP_ADDRESS:8443/protected/ endpoint - You should end up into an error screen at Shibboleth OP. The logs should show that the client is not trusted.

    Hints, Tips and Result

    ```
    [vagrant@gn43-oidcshibop-devel ~]$ tail -f /opt/shibboleth-idp/logs/idp-process.log
    2018-12-05 11:08:32,258 - 81.197.147.73 - WARN [org.geant.idpextension.oidc.profile.impl.
    OIDCMetadataLookupHandler:122] - Message Handler:  No client information returned for test_rp
    2018-12-05 11:08:32,264 - 81.197.147.73 - DEBUG [org.geant.idpextension.oidc.profile.impl.
    InitializeRelyingPartyContext:170] - Attaching RelyingPartyContext for rp test_rp
    2018-12-05 11:08:32,275 - 81.197.147.73 - WARN [net.shibboleth.idp.profile.impl.
    SelectProfileConfiguration:117] - Profile Action SelectProfileConfiguration: Profile http://csc.
    fi/ns/profiles/oidc/sso/browser is not available for RP configuration shibboleth.
    UnverifiedRelyingParty (RPID test_rp)
    2018-12-05 11:08:32,302 - 81.197.147.73 - DEBUG [org.geant.idpextension.oidc.profile.impl.
    AbstractBuildErrorResponseFromEvent:123] - Profile Action
    BuildAuthenticationErrorResponseFromEvent: No mapped event found for
    InvalidProfileConfiguration, creating general invalid_request
    2018-12-05 11:08:32,305 - 81.197.147.73 - DEBUG [org.geant.idpextension.oidc.profile.impl.
    AbstractBuildErrorResponseFromEvent:133] - Profile Action
    BuildAuthenticationErrorResponseFromEvent: Error response not formed
    ```

- Statically add an RP as trusted (RP config already done)
  - Verify that you have the following element in ***/opt/shibboleth-idp/conf/oidc-metadata-providers.xml***
    - The element enables loading the OIDC metadata file from the configured location

- **Hints, Tips and Result**

```
    <util:list id="shibboleth.oidc.ClientInformationResolvers"
        value-type="org.geant.idpextension.oidc.metadata.resolver.ClientInformationResolver">
        <ref bean="ExampleFileResolver" />
        <ref bean="ExampleStorageClientInformationResolver" />
    </util:list>

    <bean id="ExampleFileResolver"
        class="org.geant.idpextension.oidc.metadata.impl.FilesystemClientInformationResolver"
        p:id="ExampleFileResolver1"
        p:remoteJwkSetCache-ref="shibboleth.oidc.RemoteJwkSetCache">
        <constructor-arg>
            <bean class="java.io.File" id="ExampleFile">
                <constructor-arg type="String" value="/opt/shibboleth-idp/metadata/oidc-client.
json" />
            </bean>
        </constructor-arg>
    </bean>
```

- Add client metadata to the location defined in previous configuration ***/opt/shibboleth-idp/metadata/oidc-client.json***

```
nano /opt/shibboleth-idp/metadata/oidc-client.json

[
  {
    "scope":"openid phone",
    "redirect_uris":["https://demorp.example.com/redirect_uri"],
    "client_id":"demo_rp",
    "client_secret":"topsecret",
    "response_types":["code"],
    "grant_types":["authorization_code","implicit","refresh_token"]
  }
]
```

- Check the RP-side configuration first from ***/etc/httpd/conf.d/auth_openidc.conf***. RP has been preconfigured for these training exercises. Note that pre-configured settings are at the bottom of the file and the file includes a lot of additional settings.

Hints, Tips and Result

```
OIDCClientID test_rp
OIDCClientSecret testSecret1234
OIDCProviderMetadataURL https://IP_ADDRESS/.well-known/openid-configuration
OIDCProviderIssuer https://IP_ADDRESS
OIDCOAuthSSLValidateServer Off
OIDCSSLValidateServer Off
OIDCRedirectURI https://IP_ADDRESS:8443/protected/redirect_uri
OIDCCryptoPassphrase secret
OIDCResponseType "code"
OIDCScope "openid profile email address phone"
```

- Pick up the settings for *client_id*, *client_secret*, *redirect_uris* and *scope* from the RP configuration.

- Also add *"response_types":["id_token","code"]* and *"grant_types":["authorization_code","implicit","refresh_token"] claims and* apply them to ***/opt/shibboleth-idp/metadata/oidc-client.json***. Either replace the existing configuration for client *demo_rp*, or add an additional configuration, as instructed by the wiki (MetadataConfiguration).

  Hints, Tips and Result

  ```
  [root@gn43-oidcshibop-devel vagrant]# cat /opt/shibboleth-idp/metadata/oidc-client.json
  [
    {
      "scope":"openid info profile email address phone",
      "redirect_uris":["https://IP_ADDRESS:8443/protected/redirect_uri"],
      "client_id":"test_rp",
      "client_secret":"testSecret1234",
      "response_types":["id_token","code"],
      "grant_types":["authorization_code","implicit","refresh_token"]
    }
  ]
  ```

- Reload the OIDC client metadata by reloading service **shibboleth.ClientInformationResolverService**.

  Hints, Tips and Result

  ```
  [root@gn43-oidcshibop-devel shibboleth-idp]# /opt/shibboleth-idp/bin/reload-service.sh -id
  shibboleth.ClientInformationResolverService
  ```

- You should now get the login screen when starting the login sequence at: https://IP_ADDRESS:8443/protected/. Use username **teppo** with password **testaaja**. Check the logs during the login sequence.