Abstract

This document describes a current best practice that allows a mobile app to share the identity/authentication obtained by a different mobile app where both apps are issued by the same vendor.

Introduction

As the industry moves to a more mobile oriented environment, vendors need a way to share identity across the multiple mobile apps they deploy. While the current OAuth2 best practice allows for SSO across any mobile app by sharing the session cookies in the system browser, this has risks such as a user clearing their system browser of cookies (possibly as requested by a customer care agent) or using private browsing on iOS and Android. On most mobile platforms, mobile apps signed by the same vendor certs can share information via the system "keychain (Account Manager on Android)".

This document profiles the OAuth2 token exchange spec allowing mobile apps to share identity (SSO) between apps produced and signed by the same vendor (i.e. signed with the same vendor cert).

Notational Conventions

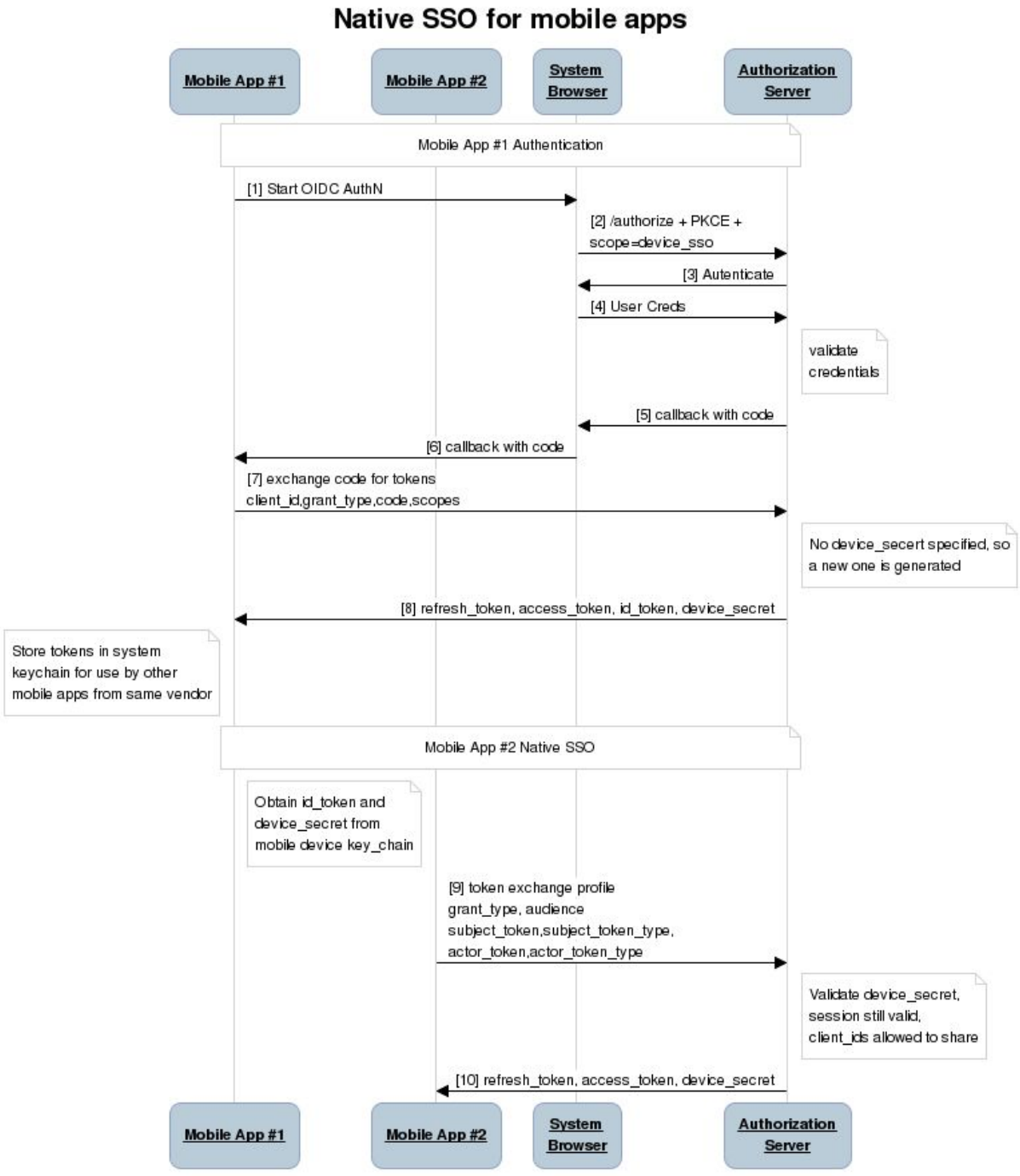
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Roles

This specification defines the following roles:

- Mobile App -- the mobile client that obtains authentication from the user leveraging RFC 8252 (OAuth 2.0 for Native Apps).
- Authorization Server -- as defined in RFC 6749 (The OAuth 2.0 Authorization Framework)

Abstract Flow



Steps [1] - [8] are the standard OpenID Connect authorization_code flow with the following extensions. In step 2, the ‘device_ssp’ scope is specified signifying that the client is requesting a device_secret to be returned when the code is exchanged for tokens.

Step [9] invokes the /token endpoint with the token exchange profile passing the id_token obtained from the shared device storage, the client_id and the device secret.
Step [10] returns the SSO generated refresh and access tokens for Mobile App #2.

UML Text

Native App Authorization Extensions

The following sections describe the extensions required to the standard OIDC Authentication flow which will enable a second mobile app to share the authentication of the first mobile app where both mobile applications are signed by the same vendor certificates.

Authorization Request [OpenID Connect Core]

This specification defines a new scope value that is used to convey to the Authorization Server that when the code is exchanged for a token, a new `device_secret` will be returned in addition to the other tokens identified as part of the authorization request.

The new scope value is defined as `device_sso`. When this scope is present on the authorization request, when the code is exchanged for tokens, a new `device_secret` will be returned.

Device Secret

The device secret contains relevant data to the device and the current users authenticated with the device. The device secret is completely Opaque to the client and as such it is RECOMMENDED that the device secret be implemented as a JWE encrypted for the Authorization Server.

In the context of this extension the device secret may be shared between mobile apps that can obtain the value via the shared security mechanism (e.g. keychain on iOS). If a mobile app requests a device secret via the `device_sso` scope and a `device_secret` exists, then the client MUST provide the `device_secret` on the request to the /token endpoint to exchange code for tokens. The client MAY provide the `device_secret` to the /token endpoint during refresh token requests.

The exact construction of the `device_secret` is out of scope for this specification.

Token Request [OpenID Connect Core]

During a normal user authentication via the system browser, after the mobile app receives the code and state response from the Authorization Server, this spec defines the following additional parameters to the /token endpoint for the authorization_code grant_type.

- device_secret -- OPTIONAL. This token SHOULD be provided if the client requested the `device_sso` scope and the client already has a `device_secret` available. If no `device_secret` is specified, a new `device_secret` will be generated.

Token Response [OpenID Connect Core]

When the authorization server receives the `device_secret` value it MUST process the authorization_code grant type per the OIDC spec with the following additions applying to the id_token.

1. Add a 'ds_hash' claim to the id_token to represent a function of the device identifier.
    a. ds_hash -- REQUIRED. The ds_hash value provides a binding between the id_token and the issued `device_secret`. The exact binding between the ds_hash and device_secret is not specified by this profile. As this binding is managed solely by the Authorization Server, the AS can choose how to protect the relationship between the id_token and device_secret.
2. Add a session id to the id_token that represents the user's current authentication session.
    a. sid -- REQUIRED. A string that uniquely identifies this user's authentication "session". This value can be used in logout flows as well as the flow this spec is describing. For mobile apps where there is not explicit "browser authentication" this value SHOULD represent the underlying session associated with the refresh_token.l

Note that the implementation of this spec and the specification of the ds_hash and sid value MUST NOT leak any data that would provide a security advantage to an attacker who gains access to the id_token.

Token Response [OpenID Connect Core]

When the authorization server receives the `device_secret` it must validate the token. If the token is invalid it must be discarded and the request processed as if no `device_secret` was specified.

If the authorization request included the `device_sso` scope then the authorization server MUST return a `device_secret` in the response. The `device_secret` is returned in the "device_token" claim of the returned JSON data.

If no `devices_secret` is specified, then the AS MUST generate the token. If a `device_secret` is specified and is valid, the AS MUST update the `device_secret` as necessary..

Native SSO Token Exchange profile

This section profiles the OAuth2 Token Exchange spec and describes the processing rules used to exchange a previous authentication for new refresh and access tokens requested by a mobile app created by the same vendor as the first mobile app and both apps signed by the same developer key.

OAuth token exchange profile

The client MUST use the HTTP Basic Authentication method from RFC 6749 to authenticate the request to the token endpoint. Note that since the mobile app is using PKCE, the mobile app does not have a secret and will only specify the client_id in the HTTP Authorization header.

This profile defines the use of the following token exchange parameters.

- grant_type -- REQUIRED. The value MUST be urn:ietf:params:oauth:grant-type:token-exchange
- audience -- REQUIRED. This parameter defines the logical purview of the returned tokens. For the purposes of this profile, this value MUST be the issuer URI for the OpenID Provider that issued the id_token used in this profile.
- subject_token -- REQUIRED. This parameter MUST contain the id_token obtained by the first mobile app. The id_token is used in the same manner as id_token_hint to identify the user to SSO into the invoking mobile app.
- subject_token_type -- REQUIRED. This parameter MUST contain the value: urn:ietf:params:oauth:token-type:id_token
- actor_token -- REQUIRED. This value defines the actor making the request which in this case is the `device_secret` issued to the device of the mobile application making the request. The `device_secret` MUST be presented per the definition of the urn:x-oath:params:oauth:token-type:device-secret token identifier described below.
- Actor_token_type -- REQUIRED. This value MUST be: urn:x-oath:params:oauth:token-type:device-sso
- scope -- OPTIONAL. The scopes required by the requesting mobile application.

This profile also defines the following token type identifiers.

- urn:x-oath:params:oauth:token-type:device-secret
    - This token type identifier is used to describe the `device_secret` specified in the actor_token parameter.

Note also that for the purpose of this profile the requested_token_type parameter is expressly omitted.

Token Exchange request for Native SSO

When a mobile app wants to request “native SSO” (i.e. obtain refresh and access tokens for an already signed in user) it makes a standard OAuth2 /token endpoint request following the profile for Token Exchange defined above.

```
POST /token HTTP/1.1
Host: as.example.com
Authorization: Basic ZGZhZGYyMzUyNDU0Og
...
grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Atoken-exchange
&audience=https%3A%3F%3Flogin.aol.com&subject_token=<id_token>&subject_token
_type=urn%3Aietf%3Aparams%3Aoauth%3Atoken-type%3Aid-token&actor_token=95tw
df3w4y6wvftw35634t&actor_token_type=urn%3Ax-oath%3Aparams%3Aoauth%3Atoken
-type%3Adevice-sso
```

The client_id in this request is sent via the HTTP Basic Authentication method using the HTTP Authorization header.

Native SSO Processing Rules

When the authorization server receives a request at the token endpoint conforming to this profile it MUST perform the following checks before issuing any tokens.

1. Validate the `device_secret` to ensure the token is still valid. The format of this secret is defined by the Authorization server and is out of scope for this specification.
2. Verify that the binding between the id_token and the device_secret as defined in the extension to the /token response.
3. Verify that the session id in the id_token (‘sid’ claim) is still valid. If the session is no longer valid, the AS MUST return an error of ‘invalid_grant’.
4. Validate that the client requesting native SSO is authorized to do so. The AS SHOULD maintain a list of client_ids that can share user authentications. For example, the AS MAY take the ‘aud’ claim from the id_token and the client_id from the token request and ensures that both client_ids are allowed to share user authentications.
5. The AS SHOULD verify that the scopes requested by the client in the token request (either default scopes or explicitly specified in the optional ‘scope’ parameter) do NOT require explicit user consent. If any requested scopes require explicit user consent the AS SHOULD fail the request and return an error of ‘invalid_scope’.

Based on the AS definition of the device_secret, the AS may perform addition check to ensure the security of the request. Provided the above criteria is met, the AS will issue a normal Token Response object containing a refresh_token, access_token and id_token issued to the client_id of the mobile app making the request. The session associated with the new refresh_token SHOULD be the same as that used to verify the validity of the SSO exchange. If that session expires, all refresh_tokens associated with it MUST be invalidated.

Profiled Token Exchange Response

The Token Exchange response for this profile has the following characteristics:

- access_token -- REQUIRED. This response field contains the access token issued to the mobile client identified by the client_id sent in the Authorization header.
- issued_token_type -- REQUIRED. This value of this parameter MUST be: urn:ietf:params:oauth:token-type:access_token
- token_type -- REQUIRED. The value of this parameter MUST be “bearer”
- expires_in -- REQUIRED. Identifies when the access_token expires.
- scope -- OPTIONAL. Follows the token exchange spec definition.
- refresh_token -- REQUIRED. A refresh_token that the mobile app can use to obtain additional access_tokens when the access_token expires.
- device_secret -- REQUIRED. The AS MUST return either a new or updated `device_secret` in the response.

In the case of any errors, the AS MUST return a valid OAuth2 Error response as described in Section 2.2.2 of the Token Exchange spec.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  “Issued_token_type”: “urn:ietf:params:oauth:token-type:access_token”,
  "token_type":"bearer",
  "expires_in":3600,
  "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
  “device_sso_token”:”casdfgarfgasdfg…”
}
```