Native SSO for Mobile Apps

George Fletcher
OATH

Nat Sakimura
NRI

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.

1. 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 OAuth 2.0 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. On most mobile platforms, mobile apps signed by the same vendor certs can share information via the system "keychain".

This document profiles the OAuth 2.0 token exchange spec allowing mobile apps to share identity (SSO) between apps produced and signed by the same vender (i.e. signed with the same vendor cert).

2. 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.

3. Roles

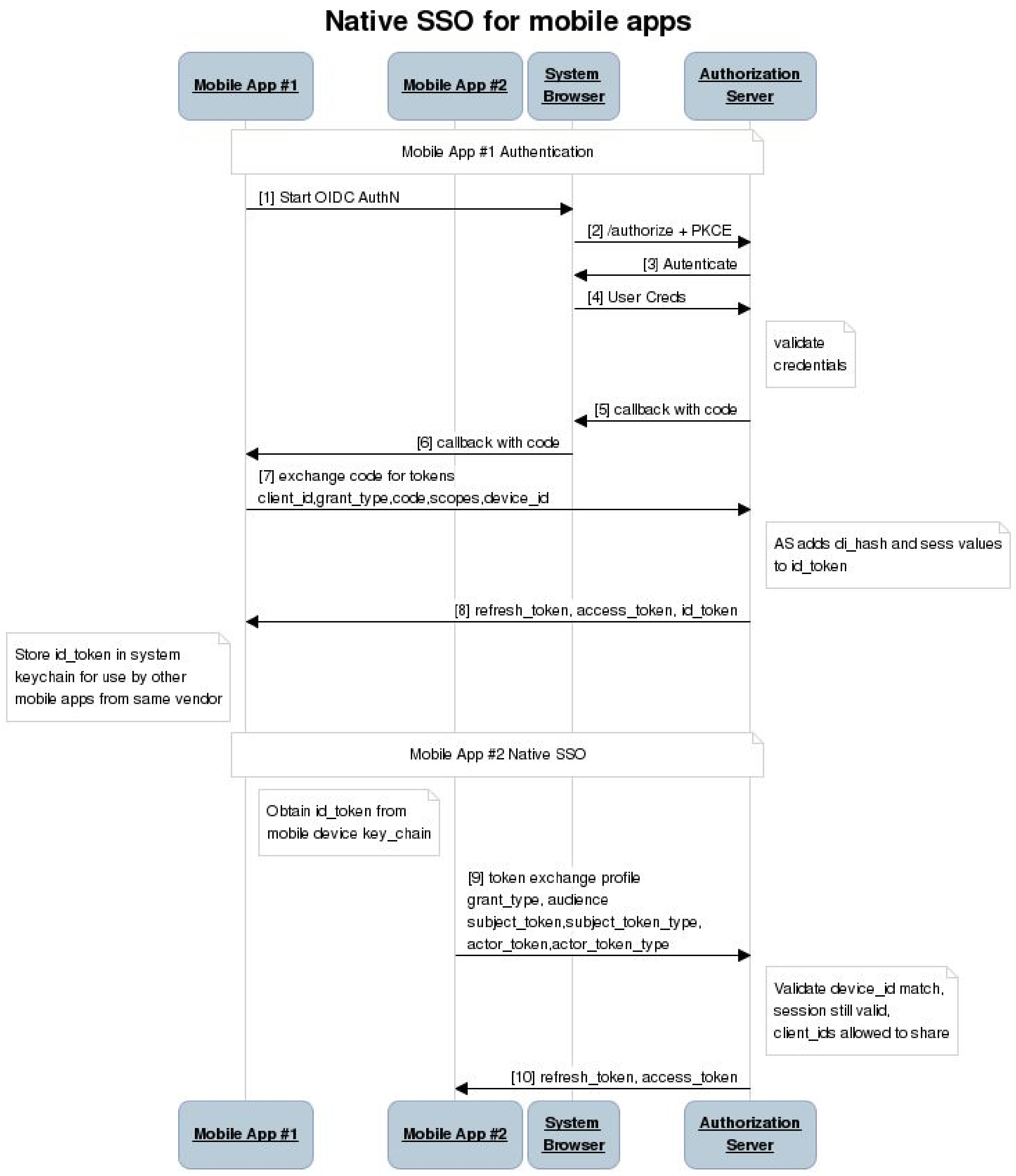
This specification defines the following roles:

- 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 [RFC6749])

4. Abstract Flow

The following is the abstract flow of the flow.


Native SSO for mobile apps
Mobile App #1
Mobile App #2
System Browser
Authorization Server
Mobile App #1 Authentication
[1] Start OIDC AuthN
[2] /authorize + PKCE
[3] Autenticate
[4] User Creds
validate credentials
[5] callback with code
[6] callback with code
[7] exchange code for tokens client_id,grant_type,code,scopes,device_id
AS adds di_hash and sess values to id_token
[8] refresh_token, access_token, id_token
Store id_token in system keychain for use by other mobile apps from same vendor
Mobile App #2 Native SSO
Obtain id_token from mobile device key_chain
[9] token exchange profile grant_type, audience subject_token,subject_token_type, actor_token,actor_token_type
Validate device_id match, session still valid, client_ids allowed to share
[10] refresh_token, access_token

Steps [1] - [8] are the standard OpenID Connect authorization_code flow with the following extensions. In step 7, the additional device_id parameter is specified containing a unique identifier that represents the device.

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 identifier.

Step [10] returns the SSO generated refresh and access tokens for Mobile App #2.

5. 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.

5.1 Token Request [OIDC]

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_id -- REQUIRED. This is string represents a unique identifier for the specific device on which the mobile app is running. It is recommended that the first app installed on the device by the vendor generate a strong random identifier to represent the device rather than relying on any device identifiers that might change.

5.2 Token Response [OIDC]

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

1. Add a ‘di_hash’ claim to the id_token to represent a function of the device identifier.
    a. di_hash -- REQUIRED. Device Identifier hash value. Its value is the base64url encoding of the left-most half of the hash of the octets of the ASCII representation of the device_id value, where the hash algorithm used is the hash algorithm used in the alg Header Parameter of the ID Token's JOSE Header. For instance, if the alg is RS256, hash the device_id value with SHA-256, then take the left-most 128 bits and base64url encode them. The di_hash value is a case sensitive string.

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.

6. Native SSO Token Exchange profile

This section profiles the OAuth2 Token Exchange spec [OTE] 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.

6.1 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.
- 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_identifier of the mobile application making the request. The device_identifier MUST be presented per the definition of the urn:x-oath:params:oauth:token-type:device-id token identifier described below.
- Actor_token_type -- REQUIRED. This value MUST be: urn:x-oath:params:oauth:token-type:device-id
- 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-id
  - This token type identifier is used to describe the token specified in the

actor_token parameter.

- The urn:x-oath:params:oauth:token-type:device-id token is a simple string containing the value of the mobile app's device_identifier.

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

6.2 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 OAuth 2.0 /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-e
xchange
&audience=https%3A%3F%3Flogin.aol.com&subject_token=<id_token
>&subject_token
_type=urn%3Aietf%3Aparams%3Aoauth%3Atoken-type%3Aid-token&act
or_token=95tw
df3w4y6wvftw35634t&actor_token_type=urn%3Ax-oath%3Aparams%3Ao
auth%3Atoken
-type%3Adevice-id
```

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

6.3 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_id from the actor_token against the 'di_hash' claim in the id_token. If the calculated left-side hash of the device_id does not match the 'di_hash' claim in the id_token the AS must return an error of 'invalid_grant'. See RFC 6749 Section 5.2 [RFC6749] .

2. Check that the session identifier in the id_token is still valid. The AS MUST take the 'sid' claim from the id_token and verify that it is still valid for the user identified by the 'sub'

claim in the id_token. If the session is no longer valid, the AS MUST return an error of ‘invalid_grant’.

3. 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. In order to make this check the AS takes 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.

4. 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’.

Provided the above criteria is met, the AS will issue a normal Token Response object containing a refresh_token and access_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.

6.4 Profiled Token Exchange Response

This 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.

In the case of any errors, the AS MUST return a valid OAuth 2.0 Error response as described in Section 2.2.2 of the Token Exchange [OTE] specification.

```
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"

}
```

5. Conclusion

While the current OAuth 2.0 best practice [BCP212] 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. This degrades the user experience. By utilizing the methods proposed in this document, users can enjoy better user experience and the vendors having less support cost, which seems to be a Win-Win.

Having said that, the method introduces a very different flow to [RFC6749] and the [RFC8252]. As a result, a separate through security analysis is sought.

References

[RFC6749] Hardt, D.: The OAuth 2.0 Authorization Framework, IETF, RFC6749, <
https://tools.ietf.org/html/rfc6749>

[OTE] Jones, Nadalin, Bradley, Mortimore: OAuth 2.0 Token Exchange, <
https://tools.ietf.org/html/draft-ietf-oauth-token-exchange-12>

[RFC8252] Dennis, W. and J. Bradley: OAuth 2.0 for Native Apps, IETF, RFC8252/BCP212, <
https://tools.ietf.org/html/bcp212>

[BCP212] Dennis, W. and J. Bradley: OAuth 2.0 for Native Apps, IETF, RFC8252/BCP212, <
https://tools.ietf.org/html/bcp212>

[OIDC] Sakimura, N., Bradley, J., Jones, M., et al. :OpenID Connect Core, OIDF,
<http://openid.net/specs/openid-connect-core-1_0.html>