Adopting OAuth 2.0 for First-Party Applications

Building the Authentication Layer

Janak Amarasena
WSO2

The Authentication Layer

- Handling authentication in an API centric manner
- Being able to handle multiple authentication options
  - Ex: Passkeys, Email OTP, Magic links, Social Login, etc
- Preferably having a generic way
  - Allowing apps to handle different authentication options easily

The Authentication API

An interactive API that can handle user authentication in an API centric manner

Expectations:

- Describe what data is needed to proceed with user authentication
- Provide info for the app to build the UI representations

A passkey sample;

Generalizing the Authentication Requirements

- A generic api able to handle any authentication mechanism
- Describe what data is needed to proceed with user authentication
- Provide info for the app to build the UI representations

Generalized representation (Passkey and Email OTP)

Passkey


```
{
 ...
  "authenticators" : [
    {
      "authenticatorId" : "RklET0F1dGhlbnRpY2F0b3I6TE9DQUw" ,
      "authenticator" : "Passkey",
      "idp": "LOCAL",
      "metadata" : {
        "i18nKey": "authenticator.Fido" ,
        "promptType" : "INTERNAL_PROMPT" ,
        "additionalData" : {
          "challengeData ": "eyJyZXF1ZXN0SWQiOiJoZVRORElCbHFWMD…"
        }
      },
      "requiredParams" : [
        "tokenResponse"
      ]
    }
  ]
 ...
}
```


challengeData is base64 encoded json data containing the challenge. Base64 encoding is done as a workaround to maintain the generic API contract.

Email OTP


```
{
 ...
  "authenticators" : [
    {
      "authenticatorId" :
"ZW1haWwtb3RwLWF1dGhlbnRpY2F0b3I6TE9DQUw" ,
      "authenticator" : "Email OTP",
      "idp": "LOCAL",
      "metadata" : {
        "i18nKey": "authenticator.email.otp" ,
        "promptType" : "USER_PROMPT",
        "params" : [
          {
            "param": "otp",
            "type": "STRING",
            "order": 1,
            "i18nKey": "otp.param",
            "displayName" : "OTP",
            "confidential" : true
          }
        ]
      },
      "requiredParams" : [
        "otp"
       ]
     }
   ]
 ...
}
```

Submitting the data;

application/json

```
{
 "auth_session" : "6e6880d8-0e9d-45a6-bc4f-6efd048957c4" ,
 "selectedAuthenticator" : {
   "authenticatorId" : "RklET0F1dGhlbnRpY2F0b3I6TE9DQUw" ,
   "params": {
     "tokenResponse" : "eyJyZXF1ZXN0SWQiOiJTV2NEZmpCcEZMO..."
   }
 }
}
```

application/x-www-form-urlencoded

```
auth_session=6e6880d8-0e9d-45a6-bc4f-6efd048957c4 &
authenticatorId=RklET0F1dGhlbnRpY2F0b3I6TE9DQUw &
params.tokenResponse=eyJyZXF1ZXN0SWQiOiJTV2NEZmpCcEZMO...
```

Generalizing the Authentication Requirements cont.

The app would need to handle any authenticator in one of three ways

- Obtain user input (ex: Email OTP)
  - API response contained metadata to obtain user input
- Perform a redirection (ex: Federated login)
  - API response contains a constructed URL the app could directly use
- Interact with the device platform (ex: Passkeys: Call platform credential APIs)
  - API response contains data the app needs to use for the system call

```
promptType -> USER_PROMPT | REDIRECTION_PROMPT | INTERNAL_PROMPT
```

```
{
 ...
 "authenticators" : [
   {
    ...
     "metadata" : {
       ...
       "promptType": "INTERNAL_PROMPT",
       "additionalData" : {
         "challengeData" : "eyJyZXF1ZXN0SWQiOiJC..."
       }
 ...
```

Handling Federated Login

- Redirect to the federated Identity Provider (redirect mode)
  - Callback to the application
  - Send the code and state to the server
- Leveraging platform SDKs for better UX (native mode)
  - App handles the complete authentication
  - Shares id_token with the server
  - Security considerations
    - Trusting the assertion sent from the app
      - Signature validation
      - Audience validation
      - Nonce check
      - If nonce isn’t supported, the server can use iat and accept the assertion only briefly after issuance

Redirect mode and native mode;

Redirect mode


```
{
 "authenticatorId" : "R29vZ2xlT0lEQ0F1dGhlbnRpY2F0b3I6R29vZ2xl" ,
 "authenticator" : "Google",
 "idp": "Google",
 "metadata" : {
   "i18nKey": "authenticator.google" ,
   "promptType": "REDIRECTION_PROMPT",
   "additionalData" : {
     "state": "465551eb-81aa-40ee-bdae-c11fa5c5f18b" ,
     "redirectUrl" :
"https://accounts.google.com/o/oauth2/auth..."
   }
 },
 "requiredParams" : [
   "code",
   "state"
 ]
}
```


Using state to correlate the callback to the app

Native mode


```
{
 "authenticatorId" : "R29vZ2xlT0lEQ0F1dGhlbnRpY2F0b3I6R29vZ2xl ",
 "authenticator" : "Google",
 "idp": "Google",
 "metadata" : {
   "i18nKey": "authenticator.google" ,
   "promptType": "INTERNAL_PROMPT",
   "additionalData" : {
     "clientId": "2242226c3b0ahu68kg7..." ,
     "nonce": "880dc83b-97b1-4793-9bc6-af05a1f54240" ,
     "scope": "openid email profile"
   }
 },
 "requiredParams" : [
   "idToken"
 ]
}
```

Dealing with Multiple Authentication Options

- Letting the app know there are multiple authentication options available for the user

stepType -> MULTI_OPTIONS_PROMPT | AUTHENTICATOR_PROMPT


```
"nextStep" : {
   "stepType": "MULTI_OPTIONS_PROMPT",
   "authenticators" : [
     {
       "authenticatorId" : "RklET0F1dGhlbnRpY2F0b3I6TE9DQUw" ,
       "authenticator" : "Passkey",
       "idp": "LOCAL",
       "metadata" : {
         "i18nKey": "authenticator.Fido"
       }
     },
     {
       "authenticatorId" :
"ZW1haWwtb3RwLWF1dGhlbnRpY2F0b3I6TE9DQUw" ,
       "authenticator" : "Email OTP",
       "idp": "LOCAL",
       "metadata" : {
         "i18nKey": "authenticator.email.otp"
       }
     }
   ]
 }
```


Submitting the selection


```
{
 "authSession" : "6e6880d8-0e9d-45a6-bc4f-6efd048957c4" ,
 "selectedAuthenticator" : {
   "authenticatorId" : "RklET0F1dGhlbnRpY2F0b3I6TE9DQUw"
 }
}
```

Supporting Localization

- i18n keys for everything to be displayed in the UI
- Providing majority used language by default (in our case English)
- Providing context data to be used by the UI

```
{
 ...
  "authenticatorId" :
"ZW1haWwtb3RwLWF1dGhlbnRpY2F0b3I6TE9DQUw" ,
  "authenticator": "Email OTP",
  "idp": "LOCAL",
  "metadata": {
    "i18nKey": "authenticator.email.otp",
    "promptType" : "USER_PROMPT",
    "params": [
      {
        "param": "otp",
        "type": "STRING",
        "order": 1,
        "i18nKey": "otp.param",
        "displayName": "OTP",
 ...
}
```

```
{
 ...
 "messages": [
   {
     "type": "INFO",
     "messageId": "msg_email_otp_sent" ,
     "message": "OTP sent to joh****@abc.com",
     "i18nKey": "message.msg_email_otp_sent" ,
     "context": [
       {
         "key": "email",
         "value": "joh****@abc.com"
       }
     ]
   }
 ]
}
```

Endpoint Discovery

- Allowing links to be discoverable

```
{
 ...
 "links": [
   {
     "rel": "self",
     "href": "https://abc.com/authorize-challenge" ,
     "method": "POST"
   },
   {
     "rel": "sign-up",
     "href": "https://abc.com/sign-up" ,
     "method": "POST"
   },
   {
     "rel": "auth-enroll",
     "href": "https://abc.com/auth-enroll" ,
     "method": "POST"
   }
 ]
}
```

Why We Would Need a Complex API

- Need to handle the complexities at some level
  - Let the app deal with this?
    - Provide documentation and let the apps write the full logic
      - Too much burden on the apps
  - SDKs?
    - Replicate the logic over many platforms
  - API?
    - Absorb the complexities centrally
    - Less burden on the app
    - Maintains consistency globally
- SDKs are definitely needed, but as a thin layer to wrap the API
- SDKs/Apps built at the granular level of the API is able to handle dynamic changes to the login flow

Get Developer Feedback Early On

- Write a app capturing different authentication options while designing the authentication layer
  - Getting early feedback helped us a lot
  - A nice API might not always be easy to implement for the app
  - Multiple iterations of the Authentication API

Login Experience with the Authentication API

- Authentication flow integrated to an Android native application
- Demo flows:
  - Basic Authentication + TOTP
  - Sign in with Google (using platform SDK)
  - Passkey
- Login experience: https://www.youtube.com/watch?v=dLfmBAF5iBA

OAuth 2.0 for FiPA Example

- MFA with Passkey + Email OTP
  - (Refer next pages)

Flow with Passkey + Email OTP

Request #1: Initiation

```
POST /authorize-challenge HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

client_id=XWRkRNkJDeTiR5MwHdXROGiJka&
login_hint=johnd@abc.com
```

Response

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{
  "error": "insufficient_authorization",
  "auth_session": "cf06ae73-9158-4d14-86fb-0bbfabcb0d45",
  "nextStep": {
    "stepType": "AUTHENTICATOR_PROMPT",
    "authenticators": [
      {
        "authenticatorId": "RklET0F1dGhlbnRpY2F0b3I6TE9DQUw",
        "authenticator": "Passkey",
        "idp": "LOCAL",
        "metadata": {
          "i18nKey": "authenticator.Fido",
          "promptType": "INTERNAL_PROMPT",
          "additionalData": {
            "challengeData":
"eyJytSWmFWemdCSDNFU09uZkRDWUcwZlh5dXo1eVZuRURKdUdkaHZfZDU4IiwicHViJl
ZGVudGlhbFJlcXVlc3RPcHR7ImNoYWxsZW5nZSI6Ikx4MFRGLWYtanlrZGlSdEdaTVdCe
mw0VUZObi1sYlAxVUxYelVYazRlbGciLCJycElkIjoibG9jYWxob3N0IiwiZXh0Z..."
          }
        },
        "requiredParams": [
          "tokenResponse"
        ]
      }
    ]
  },
```

challengeData is base64 encoded json data

```
  "links": [
    {
      "rel": "self",
      "href": "https://server.example.com/authorize-challenge",
      "method": "POST"
    }
  ]
}
```

Request #2: Passkey authentication data submission

```
POST /authorize-challenge HTTP/1.1
Host: server.example.com
Content-Type: application/json

{
  "auth_session": "cf06ae73-9158-4d14-86fb-0bbfabcb0d45",
  "selectedAuthenticator": {
    "authenticatorId": "RklET0F1dGhlbnRpY2F0b3I6TE9DQUw",
    "params": {
      "tokenResponse":
"eyJyZXF1ZXN0SWQiOiJTV2NEZmpCcEZMOXdzc2YxYk92WnFwS0VtZVM4SXBocWJnb3Bm
ZXV6SEVBIiwiY3JlZGVudGlhbCI6eyJpZCI6IlFwX05WeDluS1pvLUJpRVNyZV9rMGFzT
mVBRSIsInJlc3BvbnNlIjp7ImF1dGhlbnRpY2F0b3JEYXRhIjoiU1pZTjVZZ09qR2gwTk
JjUFpIWmdXNF9rcnJtaWhqTEhtVnp6dW9NZGwyTWRBQUFBQ..."
    }
  }
}
```

tokenResponse is base64 encoded json data

Response

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{
  "error": "insufficient_authorization",
  "auth_session": "cf06ae73-9158-4d14-86fb-0bbfabcb0d45",
  "nextStep": {
    "stepType": "AUTHENTICATOR_PROMPT",
    "authenticators": [
      {
        "authenticatorId": "ZW1haWwtb3RwLWF1dGhlbnRpY2F0b3I6TE9DQUw",
```

```
        "authenticator": "Email OTP",
        "idp": "LOCAL",
        "metadata": {
          "i18nKey": "authenticator.email.otp",
          "promptType": "USER_PROMPT",
          "params": [
            {
              "param": "otp",
              "type": "STRING",
              "order": 1,
              "i18nKey": "otp.param",
              "displayName": "OTP",
              "confidential": true
            }
          ]
        },
        "requiredParams": [
          "otp"
        ]
      }
    ]
  },
  "messages": [
    {
      "type": "INFO",
      "messageId": "msg_email_otp_sent",
      "message": "OTP sent to joh****@abc.com",
      "i18nKey": "message.msg_email_otp_sent",
      "context": [
        {
          "key": "email",
          "value": "joh****@abc.com"
        }
      ]
    }
  ]
  "links": [
    {
      "rel": "self",
      "href": "https://server.example.com/authorize-challenge",
      "method": "POST"
    }
  ]
}
```

Request #3: OTP submission

```
POST /authorize-challenge HTTP/1.1
Host: server.example.com
Content-Type: application/json

{
  "auth_session": "cf06ae73-9158-4d14-86fb-0bbfabcb0d45",
  "selectedAuthenticator": {
    "authenticatorId": "RklET0F1dGhlbnRpY2F0b3I6TE9DQUw",
    "params": {
      "otp": "D98AQF"
    }
  }
}
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "authorization_code": "5f1b2c2a-1436-35a5-b8e4-942277313287"
}
```