B2Gv1 - Identity - Tech Details

August 9th, 2012

Overview

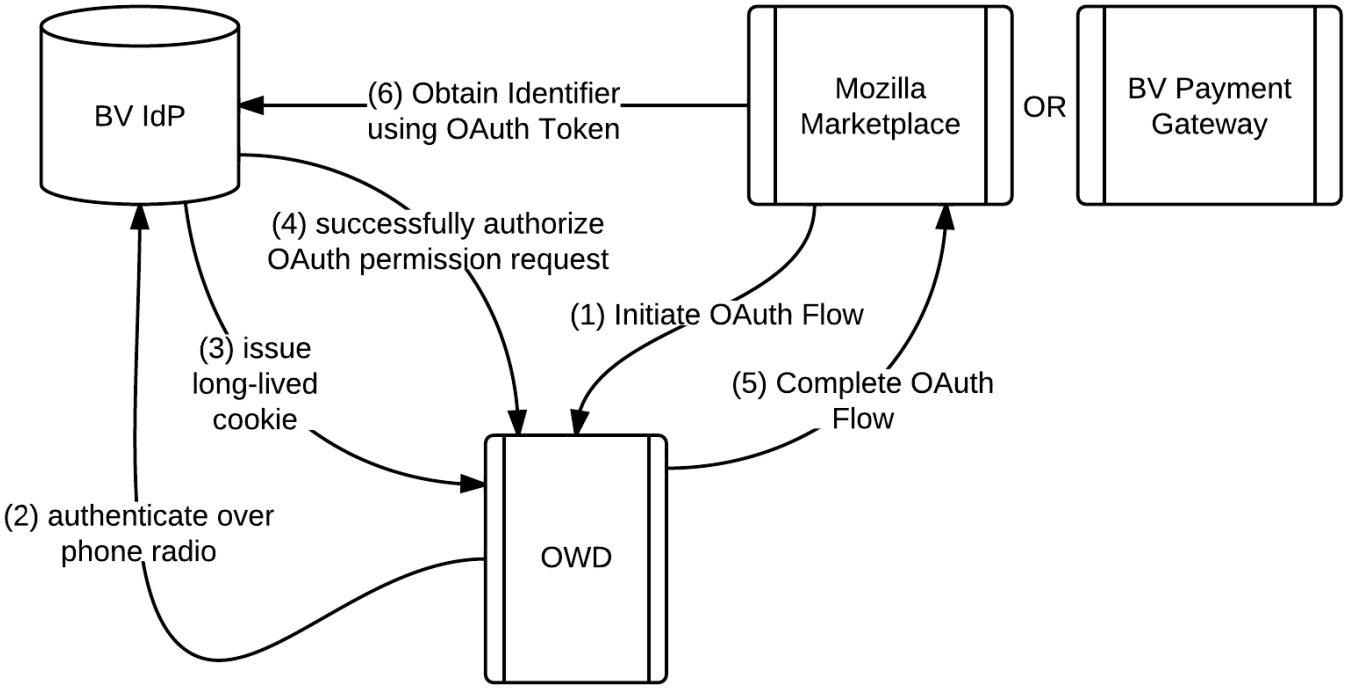
Identity needs for B2Gv1 are scoped specifically to:

- authentication to the Marketplace with a stable but opaque identifier
- authentication to BlueVia Payment Provider using identifier that enables carrier billing.

Initially, we anticipated these features to be implemented using Mozilla Persona. However, the changes required to the core Persona protocol are significant, and not easily testable. Thus, we propose using OAuth 2.0 instead.

The BlueVia IdP is implemented as an OAuth Provider, with at least one API call to provide approved OAuth Consumers a stable, opaque identifier. The Marketplace and Payment Provider Relying Parties are implemented as OAuth Consumers against the BlueVia OAuth Provider. The stable, opaque identifier can be made specific to the OAuth consumer (i.e. “directed identifier”) if BlueVia chooses.

The BlueVia Payment Provider may, in addition to the OAuth authorization flow, establish a purchasing PIN to further protect the payment flow from device theft or “borrowing.” This should be handled by the payment processor, not by the IdP.



OAuth Version and Details

We use OAuth 2.0, as defined by IETF Draft 31:

http://tools.ietf.org/html/draft-ietf-oauth-v2-31

We use a specific profile of the available specifications that works much like Facebook's Server-Side Authentication flow:

https://developers.facebook.com/docs/authentication/server-side/

We differ from Facebook in the same places where the latest OAuth 2.0 draft does for "confidential" client setup:

1. `response_type=code` is provided as an argument to the OAuth dialog.
2. APP_SECRET and ACCESS_TOKEN travel in HTTP Authorization headers
3. return values are JSON, rather than form-url-encoded.

We choose to have fixed `redirect_uri`, rather than dynamic. Each OAuth Consumer is thus set up, in a registration phase (which for our purposes is done by system administrators before launch of B2Gv1), with:

- an `APP_ID` that is public.
- an `APP_SECRET` that must be kept secret.
- a `REDIRECT_URI`.

IMPORTANTLY, ALL REQUESTS ARE MADE TO SSL-PROTECTED HOSTS.

Setup

We agree on `APP_ID`'s for the Marketplace and BlueVia Payment Provider, e.g. `mozilla_marketplace` and `bv_payment_provider`. We then generate (and keep secret!) an `APP_SECRET` for each of these. We also ensure that the BlueVia IdP knows the `REDIRECT_URI`s for each of the two OAuth Consumers, Marketplace and Payment-Provider.

BlueVia Identity Provider (OAuth Provider)

The BlueVia Identity Provider manages user accounts and is an OAuth Provider with three important functions: Authorize, Obtain Access Token, and Get User Identifier.

Authorize

```
GET /oauth/request_code?response_type=code&
    client_id=<APP_ID>&scope=identity&state=<STATE>
```

Serves a web page (HTML & JS) meant for the user's browser, which prompts the user to authorize this app to access identity information. This is expected to be accessed by the user's browser after a redirect from the OAuth Consumer (identified by `APP_ID`).

Once the user authorizes this access (or if the access has already been granted), BlueVia IdP redirects the user's browser to <`REDIRECT_URI`>`?state=<STATE>&code=<CODE>`, where

- `REDIRECT_URI` as specified during the Setup phase for the corresponding `APP_ID`.
- `CODE` is a randomly generated, unguessable one-time-use code that BlueVia IdP should remember for 1 minute.
- `STATE` is the state passed into the request_code URI earlier.

Obtain Access Token

```
POST /oauth/access_token
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
```

```
grant_type=authorization_code&code=<CODE>
```

The Authorization HTTP header should match the corresponding `APP_SECRET`. IdP also checks that `CODE` is recent and valid against the `APP_ID`, and returns an access token and the access token's expiration time, in JSON format:

```
{
   "access_token": <ACCESS_TOKEN>,
   "expires_in": <NUM_SECONDS>
}
```

The expiration should be less than 5 minutes.

Get User Identifier

```
GET /id
Authorization: Basic DzABaHbN34Sdfcv234GSd3gs3
```

The Authorization HTTP header should contain a valid `ACCESS_TOKEN` that keys to a given user. The IdP returns the opaque identifier for the user corresponding to the given valid `ACCESS_TOKEN`:

```
{
   "user_id": <USER_ID>
}
```

Mozilla Marketplace and BlueVia Payment Processor (OAuth Consumers)

Marketplace and Payment-Processor are both OAuth Consumers, with `APP_ID`, `APP_SECRET`, and `REDIRECT_URI`. When they wish to begin authentication, e.g. when the user clicks "Login with BlueVia," they begin an OAuth flow:

1. prepare the `/oauth/request_code` URL with GET parameters:
   a. a new unique `STATE`, also stored in the Consumer's session.
   b. the `APP_ID`.
   c. `scope=identity`.
   d. `response_type=code`

2. the user returns to `REDIRECT_URI` with a few GET parameters. First check that `state` is equal to the value generated in Step 1. If not, there's session inconsistency and the flow should be aborted altogether.

3. if `error` is a GET parameter, then an error occurred at authorization time. Errors are described in: http://tools.ietf.org/html/draft-ietf-oauth-v2-31#section-4.2.2.1. The flow should abort here.

4. if `code` is a GET parameter, then authorization succeeded.

5. Obtain the access token by making a direct server-to-server POST request to `/oauth/access_token` with parameters:
    a. `code=<CODE>`
    b. `grant_type=authorization_code`

    And use the `APP_ID` and `APP_SECRET` to form an HTTP Basic Auth header in that request. The result is a JSON object, including the field `access_token`.

6. Use the access token to obtain the user's ID by issuing a GET request to: `/id` with an HTTP Basic Auth header formed using the `ACCESS_TOKEN`. Parse the resulting `user_id` from the JSON structure.

There is one ***difference*** between the two OAuth Consumers: because the marketplace is an app, it should initiate the OAuth flow in a popup using window.open(), so that the user can see the URL of the BlueVia IdP. The Payment Provider, on the other hand, should simply redirect, since it will already be served in the secure popup dialog.

STRETCH GOAL: Immediate Code Request

In the normal OAuth flow, an OAuth consumer cannot know ahead of time if the OAuth flow will complete automatically (because of a prior grant), or will pause and require the user to log in / agree to terms. It may be useful for UX purposes to try to automatically log the user in, but to know deterministically if that isn't possible.

For this purpose, we can add an extra capability in the BlueVia IdP:

```
GET /oauth/request_code?response_type=code
    &client_id=<APP_ID>&scope=identity&state=<STATE>&immediate=1
```

Note the extra parameter, `immediate=1`.

In that case, the IdP will **always** immediately redirect to `REDIRECT_URI`, guaranteeing no user interaction. If the user is logged into the IdP and has previously granted identity access to this OAuth consumer, then the `REDIRECT_URI` will receive an access token in the normal success path. In any other case, the `REDIRECT_URI` will receive `error=immediate_failed`, which simply means that an immediate grant is not possible, and that an explicit button should be shown to the user.

This immediate code request can be used in an IFRAME to prevent redirect flicker.