Plan B Documentation

Release 1.0

Henning Jacobs

March 06, 2016

Contents

CHAPTER 1

Introduction

Plan B provides three infrastructure components to do *Service To Service Authentication* with JWT OAuth Bearer tokens:

Provider OAuth2 Authorization Server and OpenID Connect Provider issuing JWT tokens.

Token Info OAuth2 tokeninfo validation endpoint for JWT tokens.

Revocation Service REST service to manage token revocation lists.

1.1 Provider

The Plan B Provider issues signed JSON Web Tokens (JWT) as response to Access Token Requests with a valid Resource Owner Password Credentials Grant.

An example token creation response (`/oauth2/access_token` endpoint):

```
{
    "access_token": "eyJraWQiOiJ0ZXN0a2V5LWVzMjU2IiwiYWxnIjoiRVMyNTYifQ.eyJzdWIiOiJ0ZXN0MiIsInNjb3Bl
    "id_token": "eyJraWQiOiJ0ZXN0a2V5LWVzMjU2IiwiYWxnIjoiRVMyNTYifQ.eyJzdWIiOiJ0ZXN0MiIsInNjb3BlIjpb
    "token_type": "Bearer",
    "expires_in": 28800,
    "scope": "cn",
    "realm": "/services"
}
```

The JWT header contains the ID of the signing key and the used algorithm:

```
{
    "kid": "testkey-es256",
    "alg": "ES256"
}
```

The JWT payload contains the user ID (`sub` claim), realm and granted scopes:

```
{
    "sub": "test2",
    "scope": [
        "cn"
    ],
    "iss": "B",
    "realm": "/services",
    "exp": 1457319814,
```

```
    "iat": 1457291014
}
```

1.2 Token Info

The Plan B Token Info validates signed JWT tokens using the right public key (exposed by the Provider) and checks the token against any revocation lists.

The `/oauth2/tokeninfo` validation response contains details about the OAuth2 access token:


```
{
    "access_token": "eyJraWQiOiJ0ZXN0a2V5LWVzMjU2IiwiYWxnIjoiRVMyNTYifQ.eyJzdWIiOiJ0ZXN0MiIsInNjb3Bl
    "cn": true,
    "expires_in": 28292,
    "grant_type": "password",
    "open_id": "eyJraWQiOiJ0ZXN0a2V5LWVzMjU2IiwiYWxnIjoiRVMyNTYifQ.eyJzdWIiOiJ0ZXN0MiIsInNjb3BlIjpbIn
    "realm": "/services",
    "scope": ["cn"],
    "token_type": "Bearer",
    "uid": "test2"
}
```


1.3 Revocation Service

The Plan B Revocation Service manages token revocation lists and provides them to Token Info.

CHAPTER 2

Getting Started

All three Plan B components can be easily built and started locally. You will need:

- Java 8 JDK for Provider and Revocation Service
- Go 1.6 for Token Info
- Cassandra 2.1 for Provider and Revocation Service (can be started as Docker container)

Follow the README instructions in the respective repositories to build the components:

- https://github.com/zalando/planb-provider
- https://github.com/zalando/planb-tokeninfo
- https://github.com/zalando/planb-revocation

Start Cassandra and create the necessary keyspaces for Provider and Revocation:

```
$ docker run --name cassandra -d -p 9042:9042 cassandra:2.1
$ docker run -i --link cassandra:cassandra --rm cassandra:2.1 cqlsh cassandra < provider/schema.cql
$ docker run -i --link cassandra:cassandra --rm cassandra:2.1 cqlsh cassandra < revocation/schema.cql
```

You can use the `generate-load-test-data.py` script to generate a test key pair and test credentials:

```
$ ./provider/scripts/generate-load-test-data.py > test-data.cql
$ docker run -i --link cassandra:cassandra --rm cassandra:2.1 cqlsh cassandra < test-data.cql
```

This will create a bunch of test service users and clients, e.g.:

- username "test0" and password "test0"
- client ID "test0" and client secret "test0"

Now start the Provider (default port 8080) and Token Info (default port 9021).

Let's first check that our OpenID Provider runs and contains at least one test key pair:

```
$ curl http://localhost:8080/.well-known/openid-configuration # should contain jwks_uri
$ curl http://localhost:8080/oauth2/connect/keys # should return at least one public key
```

We can create a new JWT token with cURL:

```
$ curl -u test0:test0 \
  -d 'grant_type=password&username=test0&password=test0' \
  http://localhost:8080/oauth2/access_token?realm=/services
```

Afterwards we can validate the JWT using the Token Info:

```
$ TOKEN=... # insert "access_token" value from above
$ curl -H "Authorization: Bearer $TOKEN" http://localhost:9021/oauth2/tokeninfo
```

CHAPTER 3

Service To Service Authentication

The main driver for Plan B's development was enabling "Service to Service" authentication with OAuth Bearer tokens.

Application A (client) needs to authenticate itself, so that application B (resource server) can grant access:

- Service user and OAuth client for application A are created in the Plan B Provider
- A's user and client credentials are distributed (e.g. via S3 buckets), so that A can read them
- Application A gets a new OAuth access token (JWT) from the Plan B Provider
- The client A calls the desired HTTP endpoint of application B, passing the JWT token in the `Authorization` header ("Bearer {token}")
- The resource server B validates the token by calling the Token Info endpoint
- Token Info will verify the JWT signature and returns a JSON token info structure with A's service user ID in the `uid` property

For this to work, both applications need to know about the authentication infrastructure:

- A needs to know its service user (username and password) and client credentials (client ID and secret)
- A needs to know the OAuth2 access token URL (`OAUTH2_ACCESS_TOKEN_URL` environment variable) to create JWT tokens, e.g. https://planb-provider.example.org/oauth2/access_token
- B needs to know the OAuth2 token info URL (`TOKENINFO_URL` environment variable), e.g. https://planb-tokeninfo.example.org/oauth2/tokeninfo

CHAPTER 4

Credential Rotation

It's good practice to rotate all credentials regularly:

- OAuth service user passwords should be rotated frequently, e.g. every two hours.
- OAuth client credentials (client ID and secret for confidential clients) might be rotated every month.

The Provider allows rotating both user and client credentials via its `/raw-sync/` REST API.

To rotate an application's user credentials, do:

- Add a new password by sending a POST request to `/raw-sync/users/{realm}/{id}/password`.
- Distribute the new password (e.g. via a S3 Mint Bucket).
- Wait at least 10 minutes to give the application time to pick up the new password. Both old and new password will be active during this grace period.
- Commit the new password by overwriting the user's passwords via a PATCH request to `/raw-sync/users/{realm}/{id}`.

CHAPTER 5

Revocations

Plan B allows revoking JWT tokens via three different revocation types:

`TOKEN` Revoke single JWT tokens.

`CLAIM` Revoke all JWTs having a specific claim value.

`GLOBAL` Revoke all JWTs issued before a certain date.

CHAPTER 6

Monitoring

6.1 Cassandra

The Cassandra cluster should be closely monitored using the usual Cassandra JMX beans, e.g.:

- Number of UP/DOWN nodes
- Number of Read/Write requests per second
- Read/Write latency

6.2 Provider

The Provider exposes metrics on its management port 7979 with path /metrics.

6.3 Token Info

The Token Info exposes metrics on its metrics port 9020 with path /metrics.