Flexible, Risk Aware Authentication System - Project's Preliminary Report

Diogo Matos - 102848, David Araújo - 93444

March 15, 2024

Table of Contents

Introduction

The objective of this report is to present an architecture and description of the services, general authentication and authorization flows, and a general risk tracking model. This serves as the basis for the creation of the applications and the Identity Provider (IdP).

The IdP

The goal of the IdP is to identify and authenticate users and limit their access to resources using OAuth 2.0.

Service/application registration in the IdP should be configured through a configuration file without the need for any additional code, within the capabilities of the IdP.

Supported authentication methods

Supported authentication methods: - Password; - One Time Password via email; - TOTP/HOTP via a smartphone with support for any authentication application (e.g. Google Authenticator, Aegis); - Portuguese citizen card.

An authorization token is the output of a successful login. This token is a random string and has a configurable expiration time after creation. The token can be updated if requested within its lifetime.

Authentication risk tracking

The IdP will be able to **utilize context to determine the extent of the authentication flow required** to ensure user authentication. The following will be supported: - Access IP history; - Number of logins in the last X hours; - Number of login failed attempts in the last X hours. - Login timestamp

Since the applications that use the IdP have access to more data to measure client-specific risk, an explicit risk value (two-levels) can be provided in the request, which will determine the required authentication flow.

Authorization

In terms of authorization, the **IdP will only limit the access to the resource servers**, extra authorization mappings are stored and applied by the resource servers itself. **The access token returned by the IdP is a JWT with enough user-specific generic information** (e.g. group, role) and it's properly signed by the IdP. The set of rules in the IdP that limits the access of a given user to a resource server is defined in the service configurations in the IdP and should be based in the user's profile.

Clients

The clients of the IdP are the services that need the IdP for authentication and authorization. Each client is identified by; - *client_id* - random string. - *client_secret* - random string. - *issued_at* - timestamp. - *expired_at* - timestamp.

Also, has attached to it the following metadata: - *client_name* - string - *client_uri* - string (URI) - *grant_types* - string. We will only support the 'authorization_code' option since we want the user to go through the authentication process. - *redirect_uris* - list of strings. - *auth_methods* - list of authentication methods. - *auth_flow* - structure that defines the flow of the authentication.

Each one is defined in a configuration file, but not all the data is the this file, but will be given to the user when the IdP boots.

Endpoints

Authentication Endpoint

- URL: `/oauth/authorize`
- Method: GET
- Arguments: client_id (str), client_secret (str)
- Returns: redirect to registered redirect Url with authorization_code (str)
- Description: This endpoint is used by clients to initiate the OAuth 2 authentication process. When a client wants to authenticate a user, it redirects the user's browser to this endpoint. The IdP presents the user with a login page. After successful authentication, the user is redirected back to the client application with an authorization code (a random unique string).

Resource Access Authorization Endpoint

- URL: `/oauth/token`
- Method: POST
- Arguments: authorization_code (str), client_id (str), client_secret (str), scope (str)
- Returns: accessToken (JWT), refreshToken (str)
- Description: This endpoint is used by clients to exchange the authorization code obtained from the authentication process for an access token and a refresh token. Clients send a POST request to this endpoint, and the IdP validates the authorization token and returns an access token along with an refresh token.

Authentication Revoke Endpoint

- URL: `/oauth/revoke`
- Method: POST
- Arguments: access_token (str), client_id (str), client_secret (str)
- Returns: status
- Description: The services, when the user logs off or something happens that must finish the current session, the services (the clients) should send a request to this endpoint in order to revoke the access token.

Registration Endpoint

- URL: `/registration`
- Method: POST
- Argument: client_id (str), client_secret (str), ... tbd
- Return: status
- Description: We're talking about services (see next section) that shall not be accessible by everyone without a first intent from the bank. So, the user registry should be integrated within authorized services. This endpoint shall receive from the service very little information related with the bank and basic user information. The user will receive an email with a URL that will send him/her to a IdP page where the registration process will be finished (e.g. password and MFA setup).

Use case - Bank services

Description

Authentication and authorization are paramount in banking services. The various parties involved in typical processes may have access to private information and/or perform critical operations. Excessive user access or identity theft can lead to significant economic impacts. Another aspect that is clearly evident in the case of banks is the conflict of interests, where an employee with sufficient privileges can benefit themselves or others associated with them.

Roles/Agents and respective platforms

In this project, we're going to implement three platforms, each accessible to a set of users with a given role. The following are the roles (the agents): - *Client* - bank client that has a active account in the bank; - *Officer* - bank employee that works in one of the banks branch providing costumer support; - *Account Manager* - bank employee responsible to manage a set of client accounts.

From those three roles we get three distinct services, respectively: - *Home Banking* - *Bank Officer Portal* - *Account Manager Portal*

Operations

A bank operation is vast and complex, so we've selected a set of operations that will be available to the agents through their respective platforms. The following features will be implemented (note that their labels will be used later): - Home banking: - View account movements, account balances and account general information. An user can have multiple accounts. - Transfer money to another account; - Make payments; - Change personal information. - Bank Officer Portal: - View all accounts on the local branch. It can see all information related to the bank itself and high-level balances; - Make a loan request to a given client from a fixed set of pre-defined plans. - Request a new account creation (pending acceptance by the selected manager) - Account management portal: - View all information about the accounts under management; - Accept loan requests (Must bide bank rate policies) - Accept account management - Delete accounts - Change account branch

Authentication

For each agent, an authentication flow must be followed. Each of the diagrams presented in this sub-section represents the highest-risk level scenario.

Home Banking

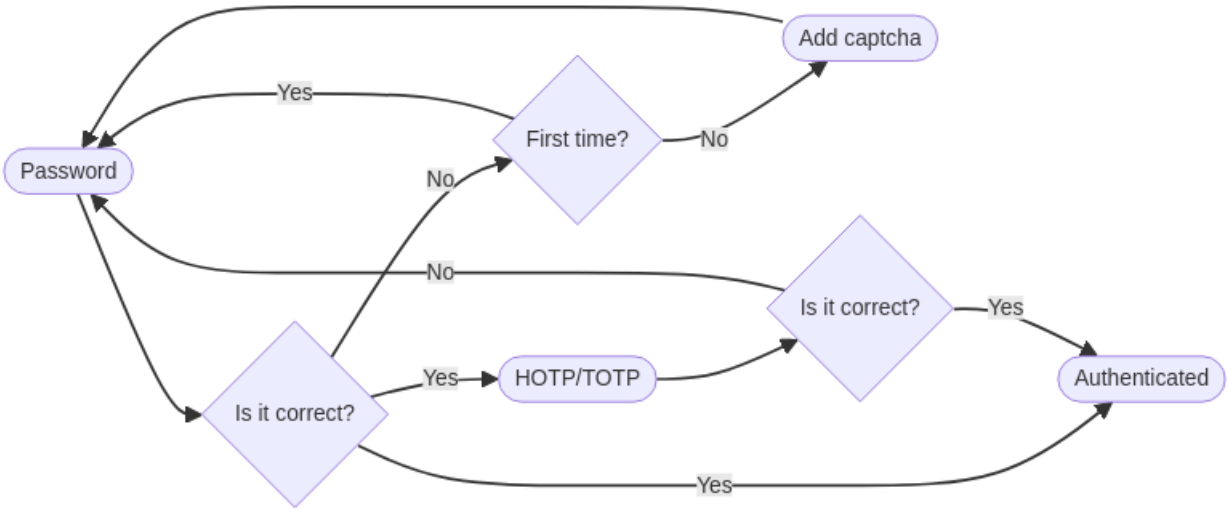

Figure 1: Home Banking authentication flow.

In normal cases the password is enough for the user to login, the full flow is followed when: - The IP from which the user is trying to login was never seen before; - Number of failed logins in the las 24 hours exceeds 4.

Bank Officer Portal

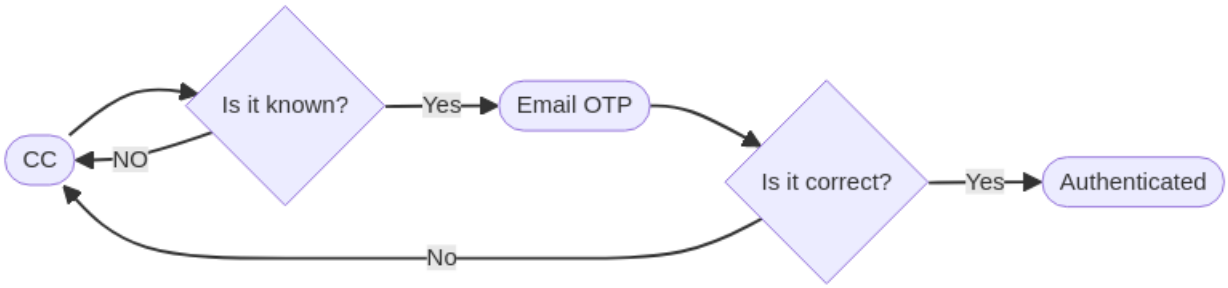

Figure 2: Bank Officer Portal authentication flow.

In normal cases the Citizens Card (CC) is enough for the user to login, the full flow is followed when: - The IP from which the user is trying to login was never seen before. Ensuring that if the CC is stolen and the access is not made from the bank's officer computer. - Every 1 week.

No logins in the platform can be made on a weekend.

Account Manager Portal

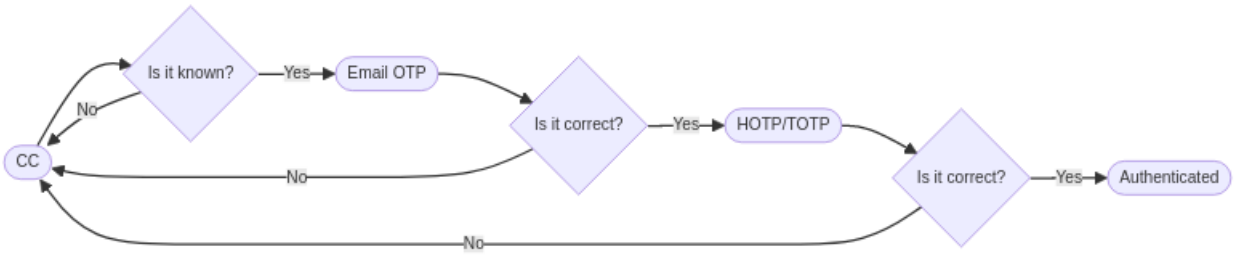

Figure 3: Account Manager Portal authentication flow.

The CC and the Email OTP are always requires. MFA fatigue can be ignored in this case because the user is too critical. The HOTP/TOTP is only required when: - It's a weekend or every day from 7pm to 7am - The IP from which the user is trying to login was never seen before; - In the last 24 hours the second step in the authentication (Email OTP) failed.

Authentication revocation

Using the `auth/revoke` endpoint, all the services can revoke a session token. This system will me used to prevent token stealing. If the service detects that the client ip attached to the authentication token (a.k.a. session token) is not the one being used currently, it must revoke the authentication token. All the three services must have this security measure implemented.

Authorization

For each agent, distinct authorization criteria must be followed. Where possible, access control will be enforced by the IdP itself. In cases where this is not possible, the resource server will perform access control based on the information it stores and the access token provided by the client.

All the data needed for all the three applications to function, will be stored in the following resource servers: - *Account information server* - stores and provides all the information about the bank's accounts. - *Payment server* - enables clients to perform, upon request, money transfers and payments. - *Loan server* - manages loan-related information.

Access control criteria can be directly mapped from the following subsections.

Access Control

	View account movements	View account balances	Make Payments	Request Account Creation	Transfer Money	Update Client Info	View Account Info	Request Loan	Accept Loan Request	Delete Account	Change Account Branch
Client	x	x	x	x	x	x	x	x	-	-	-
Bank Officer	-	x	-	x	-	-	x	x	-	-	-
Account Manager	x	x	-	x	-	-	x	x	x	x	x

Have access: **x** | Doesn't have access: **-**

More limitations: - A *bank officer* can't perform any action on accounts that he/she is titular; - An *account manager* can't manage account where he/she is titular; - An account deletion cannot be fully performed without explicit consent from the *client*. The client shall verify the action inside the Home Banking platform; - A *client* when performing either a payment or a transfer, must present a set of digits corresponding to four random-generated index of a multi-channel code that was given to the user when the account was created via email; - The *account manager* has a margin to adjust the loan request before accepting it, but it always must follow the bank's rate policies.

Information Flow Control

Regarding account management, information flow control is paramount since read/write operations should not be allowed to all agents at every action they take part in. We can implement the Bell-La Padula MLS Model to specify controls where it is not possible to read up or write down the hierarchical tree.

In actions such as *Request Account Creation* and *Request Loan*, the information follows the same pattern.

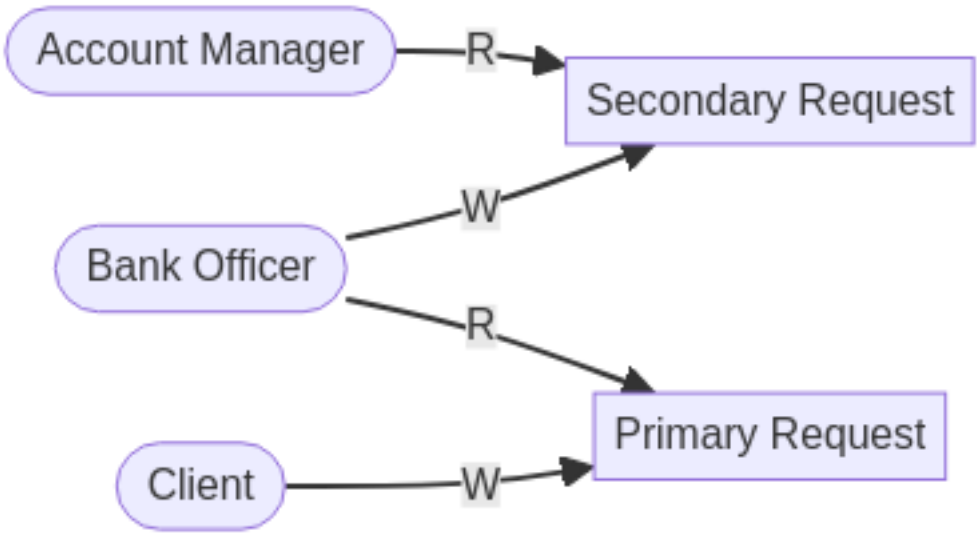

Figure 4: Request Account Creation and Request Loan information flow.

When performing other actions such as *Change Account Branch*, the information flows from the top management to the bottom, and to ensure integrity of the action, the most appropriate model to implement is the Biba Integrity Model which ensures that data can only be written down and read above.

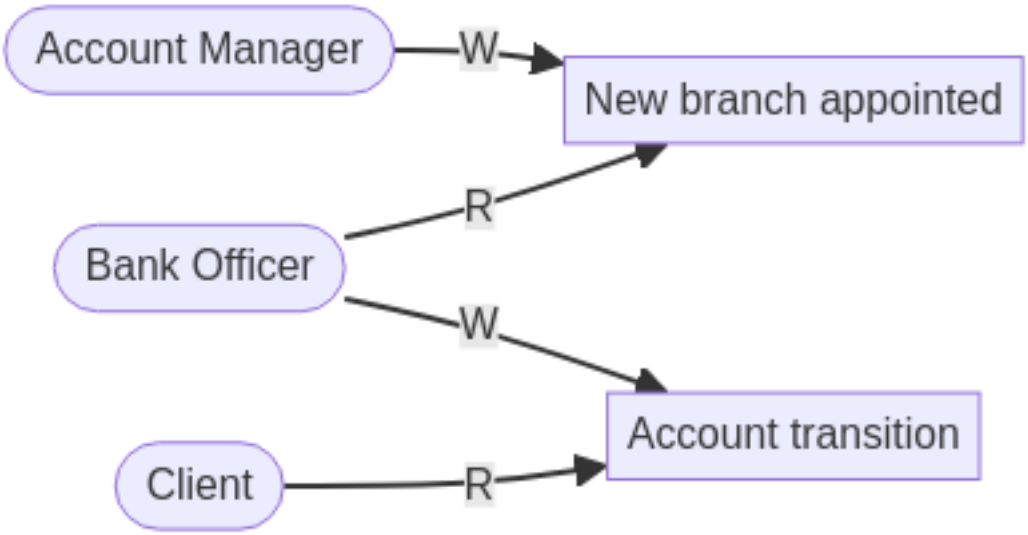

Figure 5: Change Account Branch information flow.

Authentication/authorization flow

The IdP does not enforce any structure; it requires two requests, one for the authorization token and one for the resource access token. To keep the access tokens in the backend, and since there's no need for user interaction to obtain the access token, it can be easily done by the application backend after user authentication. This way, the user has access to an authorization token that, by itself, does not grant access to the resource servers. Therefore, the application is able to add more resource access restrictions.

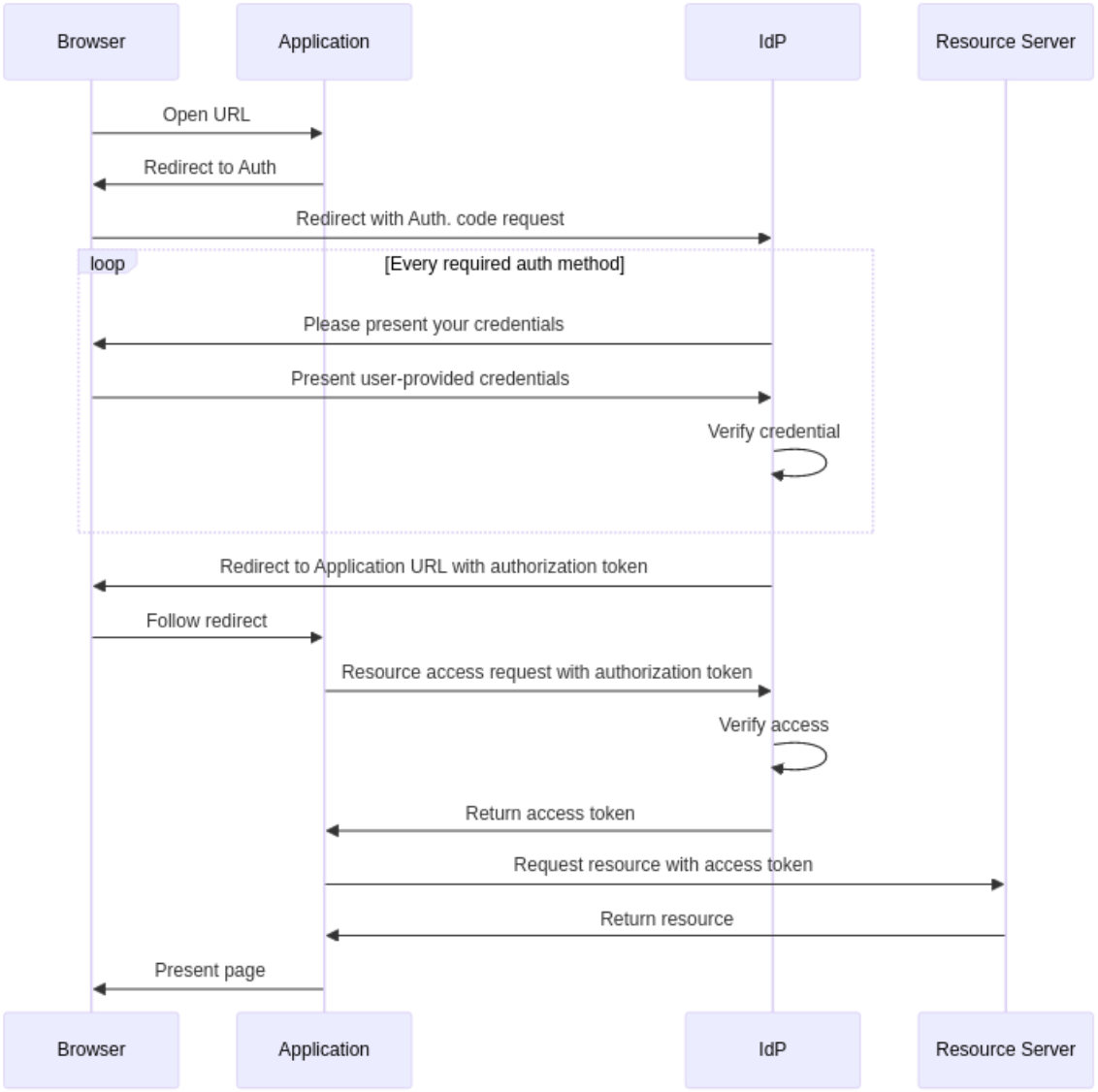

Figure 6: Authentication/authorization flow.

Conclusions

In this preliminary report lays the groundwork for the development of a flexible, risk-aware authentication system. Additionally, it presents a use case based on three distinct bank services that allow some basic, yet critical, operations. By defining the agents (roles), operations (and their flow), and access control criteria, the report provides a base for the authentication and authorization characteristics to be implemented.

Further refinement will be necessary to validate the effectiveness and reliability of the proposed system. This will follow an agile workflow that will be developed as the project progresses.