Fall 2019 Network Security Project Proposal

1. Group:

   - Prateek Narendra

   - Nishant Shankar

   - Kai-Chieh Huang

   - Ravinder Singh

2. Title: DevAuth

3. Problem:

OAuth 2.0 has seen increasing adoption across a wide variety of websites. It's primarily used to authorize client application access to data on third-party websites. Most of the times this data is present on social networking platforms or big tech providers, like Google and Facebook. OAuth is convenient: it solves the problem of password reuse by allowing the use of the same account across different websites, but that creates another issue. The main problem in this infrastructure is reliance on these third-party data holders and identity providers to relay information to client applications. We seek to address the problem of our reliance on corporate ownership of personal data. Can we selectively allow client applications access to data without them going through third parties? In this document, we describe our proposal to solve this problem.

4. Context:

CIBA, a new specification written by MODRNA Working Group of OpenID Foundation, which uses a different code flow called decoupled flow. In CIBA, client applications will call the APIs from the resource server. The devices used for authentication and consent confirmation are decoupled. This enables the client application and end-user to be two physically separated devices. Though it involves authentication, here we use a similar flow but for authorization.

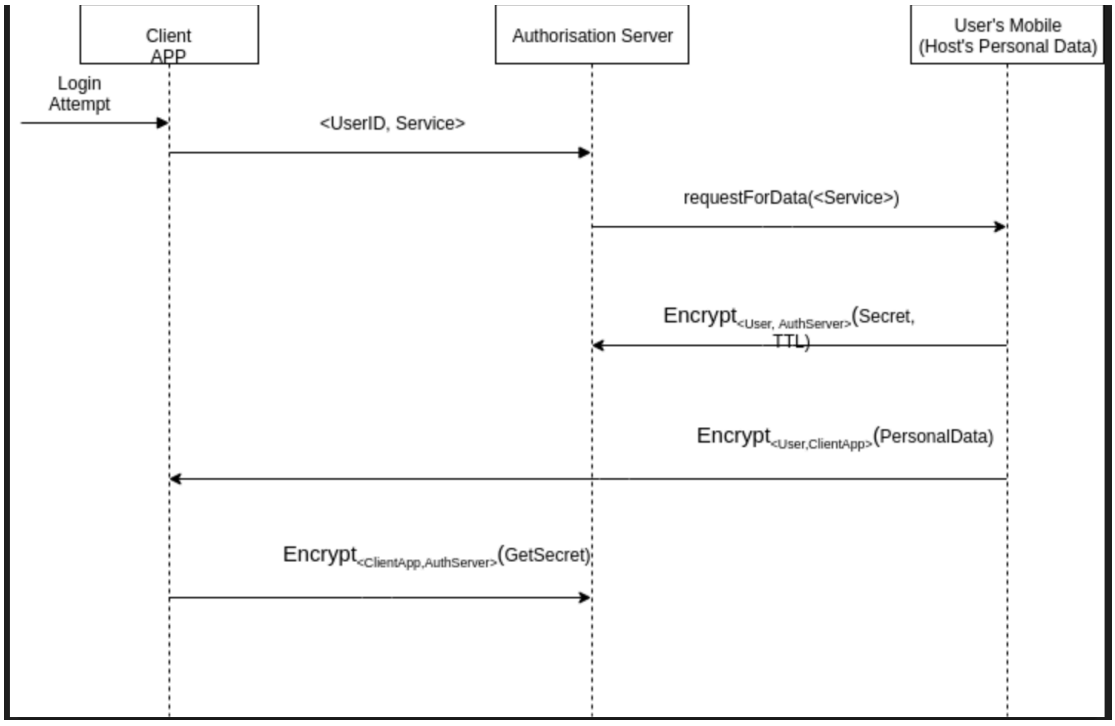
5. Approach

   - User enters his/her ID issued by the Orchestrator Service into the Client App
   - Client App sends a request to the Orchestrator Service specifying they would like the data of the user (The user would've registered

their phone on this service with the app of the Orchestrator Service)

- The Orchestrator Service responds 'OK' to the Client App. If the User ID was correct, the Orchestrator would send a notification to the User ID's phone. This notification has the callback URL of the Client App and what data it is requesting.
- If the user approves, the user will create a secret key for communication to the Client App. It would send this newly created key to the Orchestrator Service so that the Client App can decrypt and read what the Users' phone sends out.
- The user now sends the data that the client requested encrypted with the key that it chose.
- The Client App then asks the key from Orchestrator Service Server. The key is encrypted with the shared secret between the Orchestrator Service and Client App.
- The Client App can then decrypt what the User sent from the phone.

The way this system differs from the past work is that sensitive user data is currently stored on resource servers run by companies that are no longer being considered trustworthy. This system puts the user in direct ownership of the data and can choose whether he/she wants to share it with a 3rd party server.



6. Evaluation
    - Vulnerability assessments: We'll consider multiple threat models in network security and assess our system's ability to handle different types of attacks.
    - Seamless: Usability is important for ensuring ease of use, and our protocol must be seamless to use.
7. Scope
    - Evaluation between different OAuth providers
    - Building the Orchestrator Service
    - Building a sample Client App which leverages our new flow to get user data from the system.
    - Building the Orchestrator Services app for the phone
    - Evaluate that the system is protected from multiple attack vectors