BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan perkembangan internet, hampir semua sistem atau informasi dapat diakses melalui *internet*. Sehingga memungkinkan seseorang untuk mengakses suatu informasi dari mana saja. Namun karena bisa di akses dari mana pun dan kapan pun sehingga sebuah aplikasi web rentang terhadap pencurian data terutama untuk *data user*. Untuk mengatasi masalah ini suatu aplikasi web disarankan untuk menggunakan suatu protokol keamanan saat melakukan proses *login*. Protokol keamanan itu disebut *OAuth 2.0*.

OAuth 2.0 merupakan protokol keamanan yang digunakan dalam proses otentikasi *user*. *OAuth 2.*0 sendiri sudah digunakan di hampir seluruh dunia, dari penyedia skala besar seperti Facebook dan Google bahkan di sebuah perusahaan *startup*. *OAuth 2.0* memungkinkan aplikasi pihak ketiga untuk mendapatkan akses terbatas ke layanan *HTTP*, baik atas nama *resource owner* dengan mengatur interaksi persetujuan antara pemilik sumber daya dan layanan *HTTP*, atau dengan mengizinkan aplikasi pihak ketiga untuk mendapatkan akses atas namanya sendiri (Krusen, 2020).

Pada implementasi *OAuth 2.0* ada proses *authorization code*. Dimana diproses ini memberikan *token* kepada *public client*. Sayangnya, proses ini bergantung pada aplikasi yang menyediakan *client secret* dalam permintaan terakhir untuk *access token*. Untuk jenis aplikasi tertentu, itu membuat celah terhadap kebocoran kerahasiaan data dan hal ini tidak dapat dihindari. Karena ini adalah klien publik, tidak ada cara untuk menjamin keamanan rahasia yang digunakan untuk pertukaran *token*. Menggunakan *implicit flow* bisa menyelesaikan masalah itu tetapi menambah risiko mengekspos *token* akses di *URI*, yang dimana membuat proses ini rentan terhadap intersepsi aplikasi berbahaya (Krusen, 2020).

Solusi ini dapat diterima dan diperlukan saat aplikasi tidak dapat membuat permintaan lintas *domain* sehingga mustahil untuk menyelesaikan proses *authorization code*. Namun sekarang *CORS* didukung secara luas, jadi aplikasi bisa membuat request langsung kepada *token* endpoint (Krusen, 2020).

Dan tanpa adanya masalah *cross-origin*, klien publik bisa memanfaatkan *authorization code* dengan menggunakan tambahan *PKCE*, dimana untuk menggantikan *client secret* dengan *string* yang dibuat secara dinamis. Dengan begitu, *PKCE* menghilangkan risiko kebocoran kerahasiaan data sambil mengizinkan *server* otorisasi untuk memverifikasi bahwa aplikasi yang meminta *token* akses sama dengan yang memulai proses *OAuth*.

Untuk itu salah satu solusi untuk menghindari kebocoran data saat proses otentikasi yang diharapkan dapat diterapkan dan dilakukan dengan cara menambahkan *PKCE* pada *OAuth 2.0*. Berdasarkan latar belakang yang telah disampaikan, maka dibuatlah suatu penelitian dengan merancang sistem *login* yang menggunakan metode otentikasi *OAuth 2.0* dengan menambahkan *PKCE* pada proses *login* sebuah *website*.

1.2 Perumusan Masalah

Dari penjelasan latar belakang di atas, maka rumusan masalah yang dapat didefinisikan dalam penelitian ini adalah sebagai berikut :

1. Bagaimana merancang dan membangun sebuah sistem *login* yang bisa dipakai pada banyak *client*?
2. Bagaimana merancang dan membangun sistem otorisasi *client* yang tidak mudah dipakai oleh sembarang orang ?
3. Bagaimana menguji proses otorisasi *client* pada sistem yang dirancang?

1.3 Tujuan Penelitian

Tujuan yang ingin di capai dari penelitian ini antara lain :

1. Merancang dan membangun sistem *login* yang bisa dipakai oleh banyak *client*.
2. Menciptakan sistem pemeriksaan yang memungkinkan *server* memverifikasi permintaan otorisasi berasal dari *client* yang terdaftar.

3. Mengetahui seberapa besar tingkat keberhasilan dan kegagalan pada rancangan sistem *login* yang dibuat untuk memvalidasi *client*.

1.4 Manfaat Penelitian

Diharapkan dengan adanya penelitian ini dapat diambil beberapa manfaat dari penelitian ini antara lain :

1. Memahami bagaimana merancang sistem *login* yang bisa dipakai oleh banyak *client.*
2. Memahami bagaimana membuat sistem otorisasi *client* yang hanya digunakan oleh *client* yang terdaftar.
3. Memahami dan menambah wawasan apa saja penyebab kegagalan yang terjadi pada proses otorisasi *client*.

1.5 Batasan Masalah

Beberapa batasan masalah dalam penelitian ini adalah sebagai berikut :

1. Pengujian dilakukan pada satu *server*.
2. Pengujian fokus pada proses otorisasi *client*.
3. Pengujian dilakukan pada *browser chromium* (versi 98.0.4726).