Jurnal Pendidikan dan Teknologi Indonesia (JPTI)
Vol. 2, No. 2, Februari 2022, Hal. 81-91
DOI: https://doi.org/10.52436/1.jpti.141
p-ISSN: 2775-4227
e-ISSN: 2775-4219

Perancangan dan Implementasi Sistem Otentikasi OAuth 2.0 dan PKCE Berbasis Extreme Programming (XP)

Rahmat Kurniawan[*1]

[1]Universitas Mercubuana Yogyakarta, Indonesia
Email: [1]rahmatkurniawan339@gmail.com

Abstrak

Perusahaan XYZ adalah sebuah perusahaan swasta yang menyediakan sebuah produk *Hospital Management System (HMS)* yang bersifat *subscription*. Dimana lebih dari satu rumah sakit menggunakan produk ini dengan satu *server* yang terpusat. Permasalahan yang ingin diuji pada penilitian ini adalah adalah bagaimana merancang sistem otentikasi dan otorisasi untuk sistem *HMS multitenant* sehingga sistem tidak bisa digunakan oleh pihak yang tidak terdaftar. Tujuan penelitian ini adalah merancang dan menerapkan prosedur otentikasi dengan mekanisme otentikasi *OAuth 2.0* dan *PKCE* pada aplikasi *HMS multitentant* dengan melibatkan suatu *server* dan *client* dalam melakukan proses otentikasi untuk mempermudah proses otentikasi pada tiap *tenant*. Pada penelitian ini akan melibatkan tiga aplikasi. Satu aplikasi sebagai *middleware* dimana terdapat halaman *sign-in* berbasis *OAuth 2.0*. Dan dua aplikasi lagi sebagai *client* dan *server*. Pada penelitian ini dilakukan proses pengembangan sistem menggunakan metode *Extreme Programming (XP)*. Hasil dari penelitian ini berupa sistem *login* atau sistem otorisasi dan otentikasi yang secara nyata dapat memenuhi kebutuhan perusahaan XYZ. Sistem *login* ini memiliki kelebihan dimana mempermudah perusahaan XYZ untuk mengatur client yang terintegrasi dengan sistem ini dan bagi *client* mudah untuk menimplementasikannya.

Kata kunci: *Authorization Server, OAuth 2.0, Otentikasi dan Otorisasi. PKCE.*

Design and Implementation of Authentication System OAuth 2.0 and PKCE Based on Extreme Programming (XP)

Abstract

XYZ Company is a private company that provides a subscription Hospital Management System (HMS) product. Where more than one hospital uses this product with one centralized server. The problem to be tested in this research is how to design an authentication and authorization system for a multitenant HMS system so that the system cannot be used by unregistered parties. The purpose of this study is to design and implement authentication procedures with OAuth 2.0 and PKCE authentication mechanisms on multitentant HMS applications by involving a server and client in the authentication process to facilitate the authentication process for each tenant. This research will involve three applications. One application as middleware where there is a sign-in page based on OAuth 2.0. And two more applications as client and server. In this study, the system development process was carried out using the Extreme Programming (XP) method. The result of this research is a login system or authorization and authentication system which can actually meet the needs of XYZ company. This login system has the advantage that it makes it easier for XYZ company to manage clients that are integrated with this system and for clients it is easy to implement it.

Keywords: *Authentication and Authorization, Authorization Server, OAuth 2.0, PKCE.*

1. PENDAHULUAN

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 manapun dan kapanpun sehingga sebuah aplikasi web rentang terhadap pencurian data terutama untuk data *user* [1]. 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* [2]. *OAuth 2.0* sendiri sudah digunakan di hampir seluruh dunia, dari penyedia skala besar seperti Facebook dan Google bahkan di sebuah perusahaan *startup* [3]. *OAuth 2.0* memungkinkan aplikasi pihak ketiga untuk mendapatkan akses

p-ISSN: 2775-4227
e-ISSN: 2775-4219

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 [4].

Pada implementasi *OAuth 2.0* ada proses *authorization code*. Dimana diproses ini memberikan *token* kepada *public client* [5]. 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 kerahasian 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.

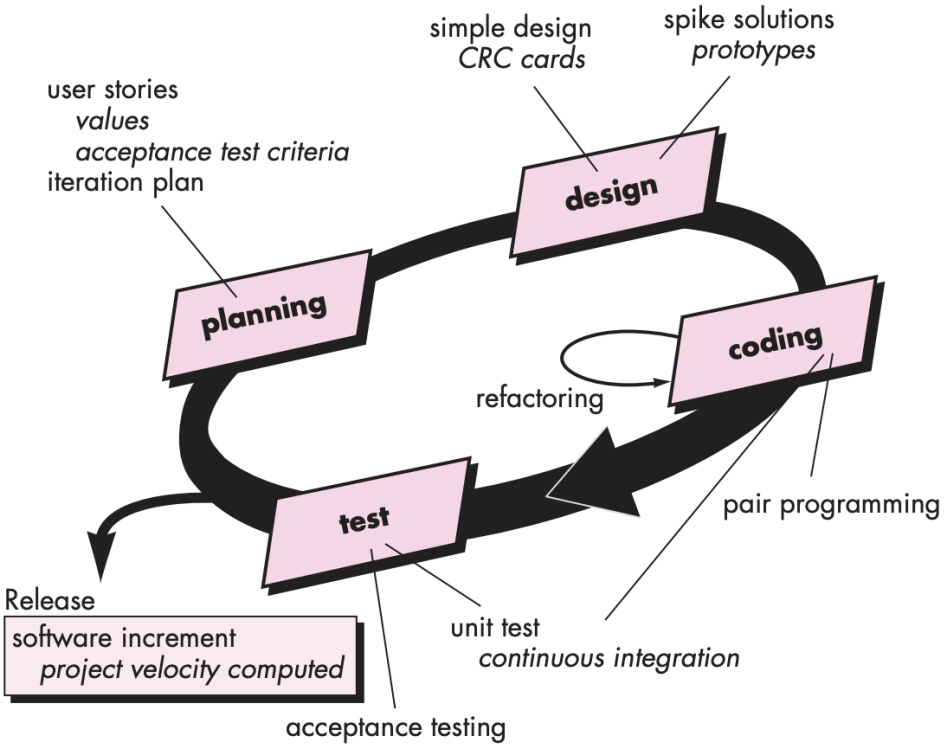
Solusi ini dapat diterima dan diperlukan saat aplikasi tidak dapat membuat permintaan lintas *domain* sehingga mustahil untuk menyelesaikan proses *authorization code*. Namun sekarang *CORS* didukung secara luas, jadi aplikasi bisa membuat request langsung kepada *token endpoint*.

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 kerahasian data sambil mengizinkan *server* otorisasi untuk memverifikasi bahwa aplikasi yang meminta *token* akses sama dengan yang memulai proses *OAuth* [6].

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.

2. METODE PENELITIAN

Pada penelitian ini menggunakan metodologi penilitian Extreme Programming karena pada XP lebih menekankan pada praktik rekayasa perangkat lunak daripada manajemen proyek. Dan XP juga lebih fokus pada engineer daripada project manager. Berikut tahapan-tahapan metodologi dalam penelitian ini, seperti pada Gambar 1.



Gambar 1 Proses Extreme Programming (XP) [7]

p-ISSN: 2775-4227
e-ISSN: 2775-4219

Penjelasan tahap-tahap yang tertera pada Gambar 1 tersebut, dapat dilihat pada Tabel 1.

Tabel 1 Tahapan Penelitian

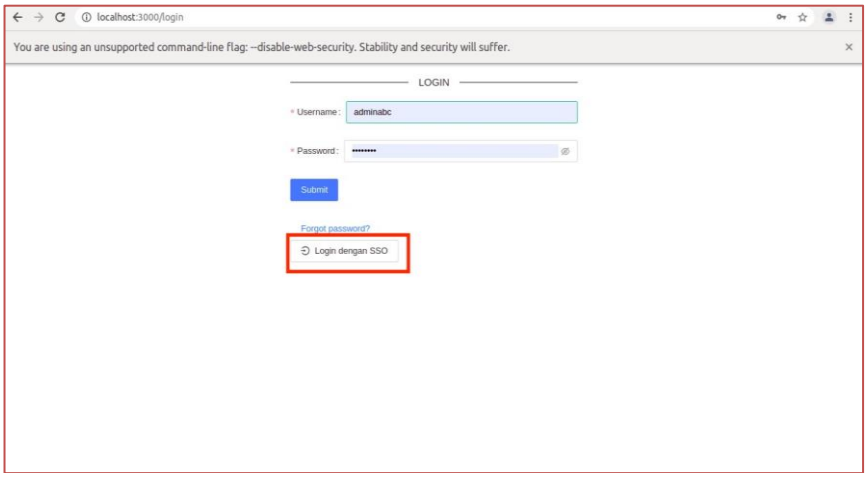
Tahap	Keterangan
Planning	Pada tahap ini peneliti mengumpulkan informasi dari berbagai sumber di internet. Juga membaca dokumentasi yang ada pada *RFC*. Serta melihat implementasi dari website lain yang menggunakan *OAuth 2.0*. Mengumpulkan informasi dan proses-proses yang perlu diimplementasi agar fitur otentikasi *OAuth 2.0* dan *PKCE* berjalan dengan sesuai dan memenuhi kebutuhan untuk produk *multitenant*
Design	Pada tahap ini peneliti mencoba merancang kebutuhan sistem yang dibutuhkan. Rancangan sistem dilakukan peneliti berupa rancangan sistem *database*, rancangan sistem tampilan proyek, dan rancangan alur program yang akan dibaut. Rancangan ini akan digunakan untuk proses penelitian ini pada fitur otentikasi *OAuth 2.0* dan *PKCE*
Coding	Pada tahapan ini peneliti mencoba mengimplementasikan rancangan yang dibuat pada tahap sebelumnya. Dimana meliputi proses otentikasi pada *client*, proses autorisasi untuk akun yang mencoba *login*, dan penukaran *code* yang didapat dengan *token*.
Test	Tahap ini peneliti melakukan beberapa pengunjian aplikasi dengan memberikan memberapa data dan skenario. Seperti menggunakan *client* yang sudah terdaftar pada sistem dan *client* yang tidak terdaftar atau *client* sudah dihapus datanya dari *server*.
Software Increment	Pada tahap ini adalah penelitian untuk mendemonstrasikan aplikasi yang sudah dibuat ke perusahaan yang akan memakai fitur ini.

3. HASIL DAN PEMBAHASAN

3.1. Hasil Penelitian

Implementasi dan perancangan sistem *login* merupakan hasil dari perancangan yang telah dilakukan sebelumnya. Berikut adalah hasil dari implementasi sistem yang telah dibuat.

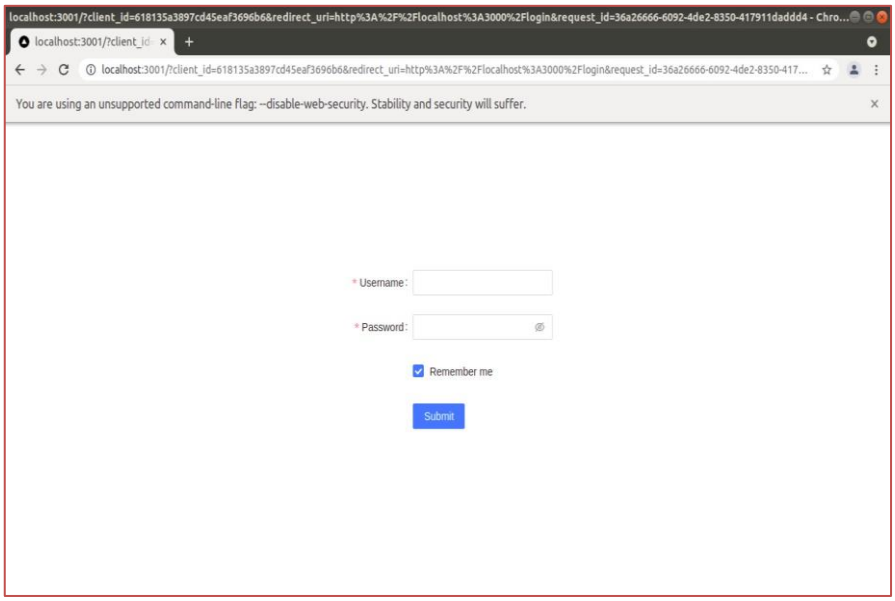
Pada Gambar 2 merupakan halaman *client* yang mengintegrasikan webnya dengan sistem *login* yang dirancang sebelumnya. Terdapat satu tombol yang digunakan untuk melakukan aksi untuk melakukan proses otentikasi dari *client* ke *server*. Jika *client* tersebut terdaftar maka otomatis akan diarahkan kehalaman *login SSO*.



Gambar 2. Tempilan *client* dan tombol *SSO*

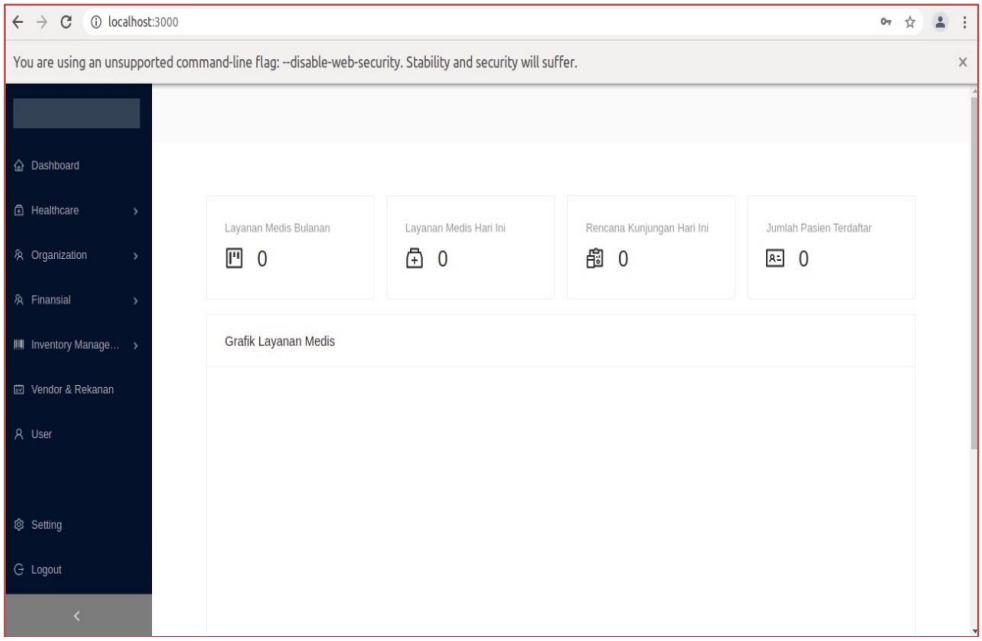
p-ISSN: 2775-4227
e-ISSN: 2775-4219

Tampilan halaman *login SSO* merupakan halaman terpusat yang akan digunakan oleh setiap *client* untuk melakukan pengecekan terhadap akun yang digunakan oleh user nantinya. Tampilan halaman *login SSO* bisa dilihat pada Gambar 3.



Gambar 3. Tampilan form l*ogin* SSO

Pada Gambar 4 ini merupakan contoh halaman dimana *user* sudah berhasil melakukan *login*.



Gambar 4. Halaman *client* berhasil *login*

3.2. Contoh Sub-Bab Kedua

a. Implementasi Sistem di *Client* Implementasi Sistem

Implementasi untuk menggunakan SSO ini pada sisi client hanya perlu menambahkan beberapa baris kode program untuk prosess otentikasi client dan pertukaran code yang nantinya akan diganti dengan access token.

Bisa dilihat Gambar 5 client perlu membuat code_challenge_method, code_verifier, dan state yang nantinya ini digunakan untuk tahap pertukaran token.

p-ISSN: 2775-4227
e-ISSN: 2775-4219

```
const loginWithSSO = async () => {
  const code_challenge_method = 'sha256';
  const code_verifier = randomstring.generate(128);
  const state = randomstring.generate(9);

  const base64Digest = crypto.createHash(code_challenge_method)
    .update(code_verifier).digest('base64');

  const code_challenge = base64url.fromBase64(base64Digest);

  const params = {
    code_challenge,
    code_challenge_method: 'S256',
    state,
    response_type: 'code',
    redirect_uri: "YOUR_CALLBACK_URL",
  };

  // CALL API OAUTH 2 /authorize
  const result = await authorizeuser(params);
  if(result.status === 200) {
    // digunakan untuk pertukaran token
    saveToLocalStorage("code_verifier", code_verifier);

    const url = result.request.responseURL;
    window.location = url;
  }
};
```

Gambar 5. Otentikasi client

Bisa dilihat pada Gambar 6 adalah merupakan contoh pertukaran *code* yang untuk yang sudah berhasil *login* ditukar dengan *access token*. Dimana *access token* ini akan digunakan untuk tiap *request.*

```
const exchangeOAuth2Token = async (code) => {
  const code = params.query.code;
  try {
    const params= {
      code,
      code_verifier: getFromLocalStorage("code_verifier"),
      grant_type: "authorization_code",
    }
    await loginWithOauth2(params);
  } catch (err) {
    console.log(err);
};
```

Gambar 6. Pertukaran *access token*

Jadi *client* yang menggunakan *SSO* ini tidak perlu membuat banyak *code* untuk *login* seperti *form* untuk *username* dan *password*, serta halaman khusus untuk *login*. Hanya perlu menambahkan beberapa baris kode program

b. Pengujian Sistem

Disini peneliti menggunakan 5 *credential client* yang akan digunakan untuk proses pengujian sistem. Hasil penelitian dapat dilihat pada Tabel 2.

Tabel 2. Hasil Pengujian *Client*

No	Data	Input	Output yang diharapkan	Hasil pengujian
1	{ "_id" : ObjectId("61a76 b6bf04e26652cb 172f3"),	POST /identity/v1/oauth2/authorize HTTP/1.1 Host: localhost:8081 Connection: keep-alive Content-Length: 220 Pragma: no-cache Cache-Control: no-cache sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96", "Google	Response code 302	HTTP/1.1 302 Found Location: http://localhost:3001?client_id=61a76b6b f04e26652cb172f3&redirect_uri=http%3 A%2F%2Flocalhost%3A3000%2Flogin& request_id=02da84b5-2b59-4f6c-8012- 1b004e7692a9 Date: Sun, 12 Dec 2021 07:20:50 GMT Content-Length: 0

p-ISSN: 2775-4227
e-ISSN: 2775-4219

No	Data	Input	Output yang diharapkan	Hasil pengujian
	"tenant" : "abcmedic", "name" : "Frontend ABC - 1", "description" : "Frontend ABC", "clientType" : "public", "isActive" : true, "permissions" : null, "redirectUri" : "http://localhost: 3000/login", "createdAt" : ISODate("2021-12-01T12:32:43.934Z"), "createdBy" : ObjectId("000000000000000000000") }	Chrome";v="96" Accept: application/json, text/plain, */* Content-Type: application/x-www-form-urlencoded Authorization: Basic NjFhNzZiNmJmMDRlMjY2NTJjYjE3MmYzOg== sec-ch-ua-mobile: ?1 User-Agent: Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Mobile Safari/537.36 sec-ch-ua-platform: "Android" Origin: http://localhost:3000 Sec-Fetch-Site: same-site Sec-Fetch-Mode: cors Sec-Fetch-Dest: empty Referer: http://localhost:3000/ Accept-Encoding: gzip, deflate, br Accept-Language: en-US,en;q=0.9		Access-Control-Allow-Origin: http://localhost:3000 Access-Control-Allow-Methods: GET, PUT, POST, DELETE, HEAD, OPTIONS
2	{ "_id" : ObjectId("61a76b6bf04e26652cb172f4"), "tenant" : "abcmedic", "name" :	POST /identity/v1/oauth2/authorize HTTP/1.1 Host: localhost:8081 Connection: keep-alive Content-Length: 220 Pragma: no-cache Cache-Control: no-cache sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96", "Google Chrome";v="96" Accept: application/json, text/plain, */* Content-Type: application/x-www-form-urlencoded Authorization: Basic NjFhNzZiNmJmMDRlMjY2NTJjYjE3MmY0Og== sec-ch-ua-mobile: ?1 User-Agent: Mozilla/5.0 (Linux; Android	Response code 302	HTTP/1.1 302 Found Location: http://localhost:3001?client_id=61a76b6bf04e26652cb172f4&redirect_uri=http%3A%2F%2Flocalhost%3A3000%2Flogin&request_id=2e17f0ec-6968-4d8c-98e9-b17326c4d0cb Date: Sun, 12 Dec 2021 07:22:50 GMT Content-Length: 0 Access-Control-Allow-Origin: http://localhost:3000 Access-Control-Allow-Methods: GET, PUT, POST, DELETE, HEAD, OPTIONS

p-ISSN: 2775-4227
e-ISSN: 2775-4219

No	Data	Input	Output yang diharapkan	Hasil pengujian
	"Frontend ABC - 2", "description" : "Frontend ABC", "clientType" : "public", "isActive" : true, "permissions" : null, "redirectUri" : "http://localhost: 3000/login", "createdAt" : ISODate("2021-12-01T12:32:43.934Z"), "createdBy" : ObjectId("000000000000000000000000") }	6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Mobile Safari/537.36 sec-ch-ua-platform: "Android" Origin: http://localhost:3000 Sec-Fetch-Site: same-site Sec-Fetch-Mode: cors Sec-Fetch-Dest: empty Referer: http://localhost:3000/ Accept-Encoding: gzip, deflate, br Accept-Language: en-US,en;q=0.9		
3	{ "_id" : ObjectId("61a76b6bf04e26652cb172f5"), "tenant" : "abcmedic", "name" : "Frontend ABC - 3", "description" : "Frontend ABC",	POST /identity/v1/oauth2/authorize HTTP/1.1 Host: localhost:8081 Connection: keep-alive Content-Length: 220 Pragma: no-cache Cache-Control: no-cache sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96", "Google Chrome";v="96" Accept: application/json, text/plain, */* Content-Type: application/x-www-form-urlencoded Authorization: Basic NjFhNzZiNmJmMDRlMjY2NTJjYjE3MmY1Og== sec-ch-ua-mobile: ?1 User-Agent: Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Mobile Safari/537.36 sec-ch-ua-platform: "Android" Origin: http://localhost:3000 Sec-Fetch-Site: same-site	Response code 401 - Unauthorized	HTTP/1.1 401 Unauthorized Content-Type: application/json Date: Sun, 12 Dec 2021 07:25:27 GMT Content-Length: 62 Access-Control-Allow-Origin: http://localhost:3000 Access-Control-Allow-Methods: GET, PUT, POST, DELETE, HEAD, OPTIONS

p-ISSN: 2775-4227
e-ISSN: 2775-4219

No	Data	Input	Output yang diharapkan	Hasil pengujian
	"clientType" : "public", "isActive" : false, "permissions" : null, "redirectUri" : "http://localhost:3000/login", "createdAt" : ISODate("2021-12-01T12:32:43.934Z"), "createdBy" : ObjectId("000000000000000000000000") }	Sec-Fetch-Mode: cors Sec-Fetch-Dest: empty Referer: http://localhost:3000/ Accept-Encoding: gzip, deflate, br Accept-Language: en-US,en;q=0.9		
4	{ "_id" : ObjectId("61a76b6bf04e26652cb172f6"), "tenant" : "abcmedic", "name" : "Frontend ABC - 4", "description" : "Frontend ABC", "clientType" : "public",	POST /identity/v1/oauth2/authorize HTTP/1.1 Host: localhost:8081 Connection: keep-alive Content-Length: 220 Pragma: no-cache Cache-Control: no-cache sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96", "Google Chrome";v="96" Accept: application/json, text/plain, */* Content-Type: application/x-www-form-urlencoded Authorization: Basic NjFhNzZiNmJmMDRlMjY2NTJjYjE3MmY2Og== sec-ch-ua-mobile: ?1 User-Agent: Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Mobile Safari/537.36 sec-ch-ua-platform: "Android" Origin: http://localhost:3000 Sec-Fetch-Site: same-site Sec-Fetch-Mode: cors Sec-Fetch-Dest: empty Referer: http://localhost:3000/ Accept-Encoding: gzip, deflate, br Accept-Language: en-US,en;q=0.9	Response code 302	HTTP/1.1 302 Found Location: http://localhost:3001?client_id=61a76b6bf04e26652cb172f6&redirect_uri=http%3A%2F%2Flocalhost%3A3000%2Flogin&request_id=29d03c3d-d5f1-426f-a39d-5fe14da36a76 Date: Sun, 12 Dec 2021 07:27:47 GMT Content-Length: 0 Access-Control-Allow-Origin: http://localhost:3000 Access-Control-Allow-Methods: GET, PUT, POST, DELETE, HEAD, OPTIONS

p-ISSN: 2775-4227
e-ISSN: 2775-4219

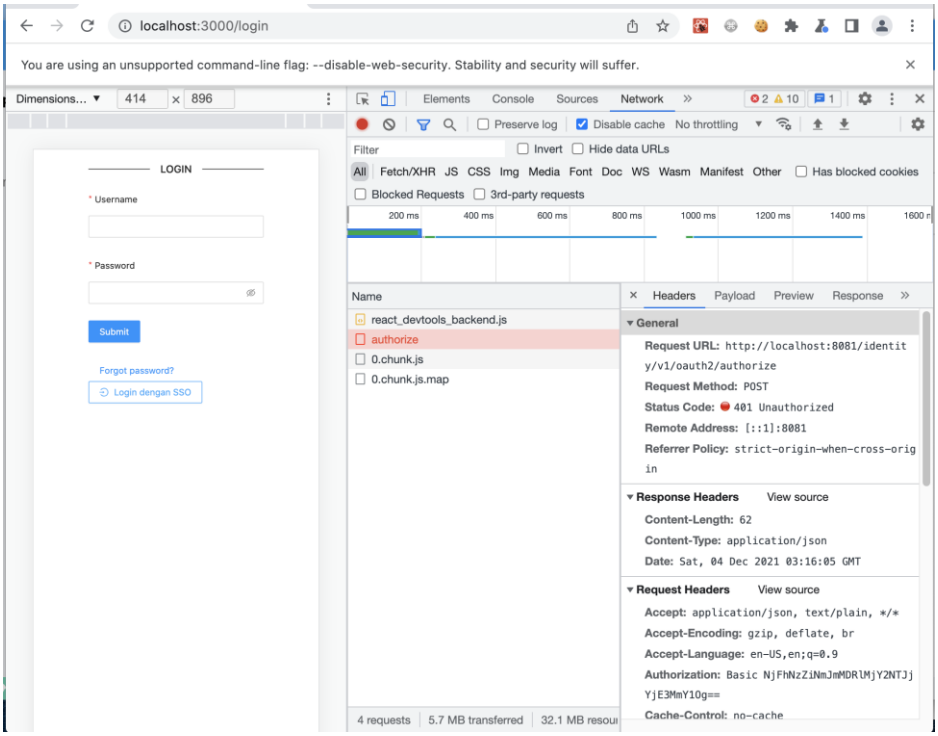
No	Data	Input	Output yang diharapkan	Hasil pengujian
	"isActive" : true, "permissions" : null, "redirectUri" : "http://localhost:3000/login", "createdAt" : ISODate("2021-12-01T12:32:43.935Z"), "createdBy" : ObjectId("000000000000000000000000") }			
5	{ "_id" : ObjectId("61a76b6bf04e26652cb172f7"), "tenant" : "abcmedic", "name" : "Frontend ABC - 5", "description" : "Frontend ABC", "clientType" : "public", "isActive" : true, "permissions" : null,	POST /identity/v1/oauth2/authorize HTTP/1.1 Host: localhost:8081 Connection: keep-alive Content-Length: 220 Pragma: no-cache Cache-Control: no-cache sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96", "Google Chrome";v="96" Accept: application/json, text/plain, */* Content-Type: application/x-www-form-urlencoded Authorization: Basic NjFhNzZiNmJmMDRlMjY2NTJjYjE3MmY3Og== sec-ch-ua-mobile: ?1 User-Agent: Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Mobile Safari/537.36 sec-ch-ua-platform: "Android" Origin: http://localhost:3000 Sec-Fetch-Site: same-site Sec-Fetch-Mode: cors Sec-Fetch-Dest: empty Referer: http://localhost:3000/ Accept-Encoding: gzip, deflate, br Accept-Language: en-US,en;q=0.9	Response code 401 - Unauthorized	HTTP/1.1 401 Unauthorized Content-Type: application/json Date: Sun, 12 Dec 2021 07:28:58 GMT Content-Length: 62 Access-Control-Allow-Origin: http://localhost:3000 Access-Control-Allow-Methods: GET, PUT, POST, DELETE, HEAD, OPTIONS

p-ISSN: 2775-4227
e-ISSN: 2775-4219

No	Data	Input	Output yang diharapkan	Hasil pengujian
	"redirectUri" : "http://localhost:3010/login",			
	"createdAt" : ISODate("2021-12-01T12:32:43.935Z"),			
	"createdBy" : ObjectId("000000000000000000000")			
	}			

Kita bisa lihat pada Tabel 2 terdapat dua pengujian yang gagal yaitu pada nomor 3 dan 5. Dan yang lainnya berhasil.

Pada pengujian nomor 3 itu terdapat *field isActive* dimana nilai dari *field* itu adalah *false*. Jika *client* memiliki nilai *isActive* adalah *false*, maka *client* tersebut bisa dibilang sudah tidak bisa melakukan atau menggunakan sistem SSO ini. Bisa dilihat pada Gambar 7 *client* akan mendapatkan *error* 401 *unauthorized* pada saat melakukan permintaan ke *server*. Jadi *server* tidak akan meneruskan atau menolak proses yang seharusnya dilakukan.

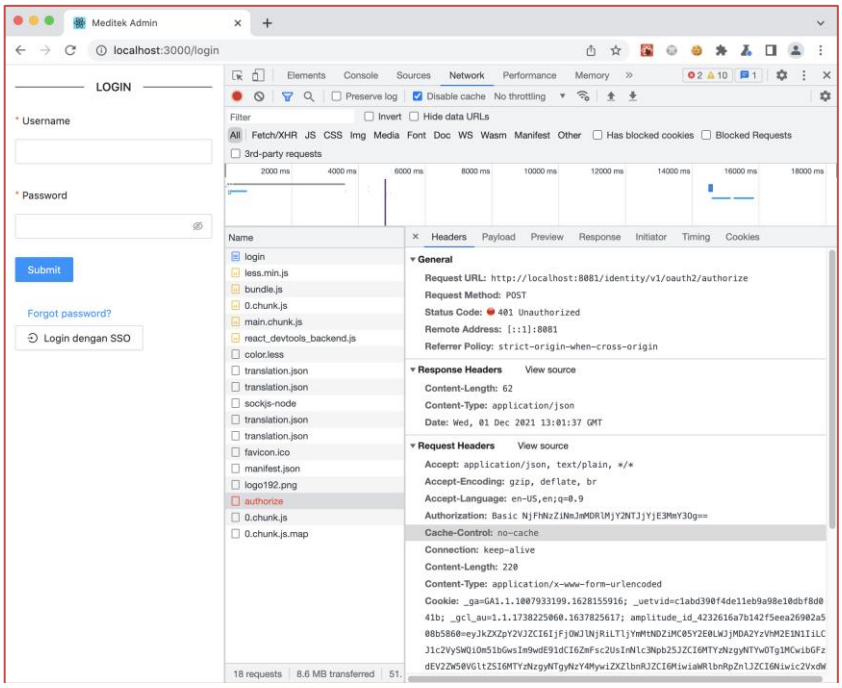


Gambar 7. Error 401 untuk *unregistered client*

Pada pengujian nomor 5 itu gagal mendapatkan *token*. Pada pengujian nomor 5 ini penguji mencoba melakukan *request* dari *domain* yang berbeda dengan nilai dari *redirectUri* yang disimpan

p-ISSN: 2775-4227
e-ISSN: 2775-4219

dalam *database server*. Bisa dilihat pada Gambar 8. Pada gambar tersebut permintaan ke *server* untuk otorisasi mendapatkan *response unauthorized* dengan *error code* 401. Itu membuktikan permintaan tersebut ditolak oleh *server*.



Gambar 8. Error dengan *domain* berbeda

4. KESIMPULAN

Dengan menggunakan otentikasi *OAuth 2.0* kita bisa dengan mudah menghapus hak akses untuk sebuah *client* tanpa harus mengubah *source code* di sisi *client*. Penerapan *OAuth 2.0* lebih aman jika dibandingkan menggunakan otorisasi *API* standar seperti *basic-auth API* ataupun *API Key*. Karena ada proses otentikasi *client* yang perlu dilakukan sebelum melakukan proses otentikasi *user*. *Client* tidak perlu membuat sistem *login* sendiri untuk memverifikasi akun pengguna.

DAFTAR PUSTAKA

[1] N. Sulisrudatin, "Analisa Kasus Cybercrime Bidang Perbankan Berupa Modus Pencurian Data Kartu Kredit," *Jurnal Ilmiah Hukum Dirgantara–Fakultas Hukum Universitas Dirgantara Marsekal Suryadarma* , vol. 9, 2018.

[2] Aminudin, "Implementasi Single Sign On Untuk Mendukung Interaktivitas Aplikasi E-Commerce Menggunakan Protocol Oauth," *Seminar Nasional Teknologi dan Rekayasa (SENTRA) SSN (Cetak) 2527-6042eISSN (Online) 2527-6050*, pp. 9-13, 2016.

[3] M. Elsera, "Implementasi Single Sign On Pada Web Menggunakan Protocol Oauth Facebook," *Buletin Utama Teknik*, vol. 16, no. 3, pp. 179-185, 2021.

[4] E. D. Hardt, "The OAuth 2.0 Authorization Framework," October 2012. https://datatracker.ietf.org/doc/html/rfc6749. (accessed december 2021)

[5] I. K. D. Senapartha, "Implementasi Single Sign-OnMenggunakan Google Identity, REST dan OAuth 2.0 Berbasis Scrum," *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 7, no. 2, pp. 307-320, 2021.

[6] T. Krusen, "PKCE: What and Why?," Dropbox, 4 December 2020. https://dropbox.tech/developers/pkce--what-and-why-. (accessed october 2021)

[7] R. S. Pressman, *Software Engineering: A Practitioner's Approach, Seventh Edition*, New York, The McGraw-Hill Companies, Inc, 2010