JWT AUTHENTICATION UNTUK KEAMANAN *WEB SERVIS E-COMMERCE* (CV JASTRA CARD)

Muhammad Roihan[1)], Andi Nurkholis[2)], Donaya Pasha[3)], Wawan Koswara[4)], Ernando Rizki Dalimunthe[5)]

[1,3,4] Teknologi Informasi, Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia
[2]Informatika, Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia
[5]Teknik Elektro, Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia
[1,2,3,4, 5]Jl. H. ZA. Pagar Alam No. 90-11 Kedaton, Bandarlampung
Email: [1]muhammad_roihan@teknokrat.ac.id, [2]andinurkholis@teknokrat.ac.id, [3]donayapasha@teknokrat.ac.id, [4]wawan_koswara2023@teknokrat.ac.id, [5]ernando_rizki_dalimunthe@teknokrat.ac.id

Abstrak

Pelayanan sistem e-commerce CV. Jastra Card kurang aman, terutama untuk transaksi data antar jaringan. Penelitian ini bertujuan untuk memberikan keamanan dan distribusi yang efektif pada web service e-commerce yang menggunakan JSON Web Token dengan algoritma HMAC-SHA 512 serta dikombinasikan dengan metode refreshing token authentication pada CV Jastra Card, yang merupakan salah satu cara untuk melindungi kerahasiaan dan integrasi data. Penelitian ini menggunakan metode pengembangan aplikasi cepat yang mencakup fase rencana, desain pengguna, pembangunan, dan cutover. Hasil penelitian ini menunjukkan bahwa token web JSON dapat melindungi autentikasi akun pengguna saat menggunakan framework Django yang berbasis Python untuk mengembangkan sistem web service. Karena refreshing token memiliki kemampuan untuk memberikan akses selama masa hidup token dan memiliki kemampuan untuk memblokir token yang sudah expired, penggunaan refreshing token lebih efektif daripada akses token biasa.

Kata Kunci: *Keamanan, JSON Web Token, Rapid Application Development, Django.*

1. Pendahuluan

Pada era digital saat ini, keamanan menjadi masalah yang sangat diperhatikan. Ini memiliki dampak yang signifikan terutama pada sektor bisnis [1]. Sebuah situs web harus memiliki keamanan untuk melindungi privasi dan data pengguna [2]. Dengan perkembangan teknologi yang semakin meningkat saat ini, banyak metode keamanan telah dikembangkan, salah satunya adalah JSON Web Token, juga dikenal sebagai JWT [3]. JWT sangat populer karena mudah digunakan dan memiliki keamanan yang telah diuji [4]. Dengan menggunakan token untuk memverifikasi otorisasi akun pengguna, JWT dapat mengamankan transaksi data antar jaringan melalui proses autentikasi [5].

Karena kemampuan mereka untuk memudahkan pertukaran data antara berbagai platform, layanan web sekarang sangat penting untuk melakukan integrasi pada sebuah sistem [3]. RESTful (REST) adalah web *service* yang sangat populer saat ini karena ukuran pesannya lebih kecil daripada web *service* berbasis SOAP (*Simple Object Access Protocol*). Namun, REST kurang aman sehingga rentan terhadap peretasan [4]. Menggunakan JSON Web Token (JWT) untuk pengamanan dan distribusi yang efektif adalah cara untuk melindungi kerahasiaan dan integrasi data [3].

Saat ini, CV Jastra Card memiliki layanan e-commerce, tetapi belum menerapkan keamanan khusus, terutama untuk layanan web-nya. Hal ini mengganggu kepercayaan pelanggan terhadap pemesanan online, seperti kartu undangan, kartu nama, buku, dan media cetak lainnya [6]. Untuk mengatasi masalah ini, keamanan akun pengguna harus diterapkan.

Studi sebelumnya telah banyak menggunakan berbagai teknik untuk membangun Web Service dengan JWT sebagai keamanan. Studi pertama menggunakan JWT untuk mengukur autentikasi dan persetujuan dengan algoritma hashing HMAC SHA-512 untuk sistem keamanan web service (RESTful API). Studi tersebut melakukan uji coba untuk membandingkan kinerja algoritma HMAC SHA-256 dan HMAC SHA-512, dan menemukan bahwa algoritma HMAC SHA-512 lebih baik dalam proses eksekusi. Selanjutnya, penelitian kedua menyelidiki penggunaan JWT untuk autentikasi pada intropera bilitas arsitektur berbasis RESTful Web Service. Desain interoperabilitas digunakan sebagai arsitektur sistem yang dibuat dalam penelitian tersebut [7]. Studi ketiga juga membahas pembuatan RESTful Web Service menggunakan JWT yang menggunakan HMAC-SHA 512 pada arsitektur 64-bit [4]. Studi ini menunjukkan bahwa algoritma HMAC SHA-512, yang memiliki arsitektur 64-bit, mengirimkan data dengan kecepatan dan besarnya 50% lebih cepat daripada algoritma lain. Selanjutnya, penelitian keempat memeriksa penggunaan kriptografi JWT untuk menerapkan keamanan API [8]. Studi lain membatasi penggunaan SHA-256 pada JWT. Penelitian

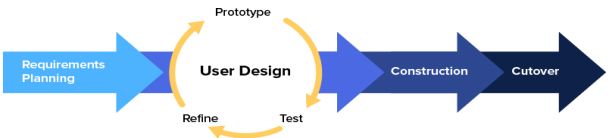
terakhir menggunakan JWT untuk menerapkan keamanan API di framework Django [9]. Sebuah sistem yang dapat mengintegrasikan atau bertukar data dari berbagai sistem telah dikembangkan dalam penelitian ini. Studi menunjukkan bahwa menggunakan HMAC SHA-512 memiliki keuntungan dalam proses pengamanan dan eksekusi.

Tujuan dari penelitian ini adalah untuk menerapkan JSON Web Token menggunakan Algoritma HMAC-SHA 512 dan metode refreshing token authentication dalam sistem keamanan Web Service E-commerce CV Jastra Card. Metode pengembangan Rapid Application Development (RAD) digunakan dalam penelitian ini, yang mencakup tahapan rencana, desain pengguna, pembangunan, dan cutover. Bahasa pemrogramman Phyton digunakan dalam framework Django untuk membuat sistem web service. Studi ini menunjukkan bahwa token web JSON dapat melindungi autentikasi akun pengguna. Karena refreshing token memiliki kemampuan untuk memberikan akses selama masa hidup token dan memiliki kemampuan untuk memblokir token yang sudah expired, penggunaan refreshing token lebih efektif daripada akses token biasa.

2. Tinjauan Pustaka

2.1. Rapid Application Development (RAD)

Rapid Application Development (RAD) adalah metode pengembangan perangkat lunak yang bersifat incremental yang menekankan siklus pengembangan berdasarkan pembuatan prototype, iterasi (berulang), dan feedback berulang-ulang, yang memungkinkan proses pengembangan yang cukup cepat. RAD lebih fokus pada keinginan pengguna [10].



Gambar 1. Metode Pengembangan RAD [10]

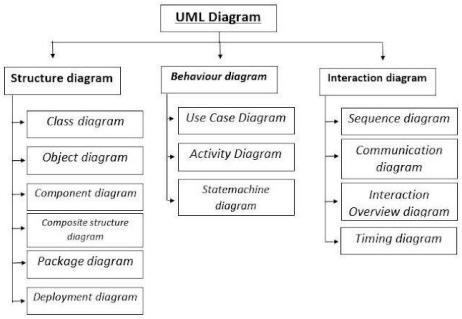
2.2. Access Token

Token akses terdiri dari string unik yang terdiri dari angka, huruf, dan simbol. Masing-masing pemilik akun layanan web service memiliki akses token yang berbeda. Ini adalah kunci yang dibutuhkan oleh pengirim layanan untuk memastikan apakah pengirim memiliki hak akses pada layanan web service. Token akses dimaksudkan untuk meningkatkan keamanan web service sehingga hanya orang yang memiliki hak akses yang dapat mengaksesnya [11].

2.3. JavaScript Object Notation (JSON)

Javascript Object Notation (JSON) adalah format data yang digunakan untuk berbagi dan menyimpan data antara client dan server. Filenya hanya berisi teks dan berekstensi.json, format teks yang berasal dari bagian dari bahasa pemrograman JavaScript dan tidak bergantung pada bahasa pemrograman apa pun karena menggunakan bahasa pemrograman yang umum digunakan oleh programmer. Oleh karena itu, JSON adalah bahasa pertukaran data yang ideal [3].

2.4. *Unified Modelling Languange* (UML)



Gambar 2. Jenis-jenis UML [12]

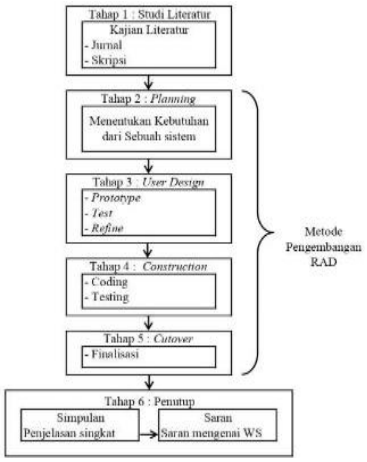
Bahasa *Unified Modelling Language* (UML), yang dibangun berdasarkan grafik dan gambar, digunakan untuk memvisualisasikan, menspesifikasikan, membangun, dan mendokumentasikan sistem pengembangan *software* berbasis objek-orientasi (OO). UML adalah sistem real-time yang berfokus pada pembuatan model konseptual yang kemudian diproses secara bertahap. Diagram Pengelolaan, Diagram Aktif, dan Diagram Komunikasi adalah beberapa contoh diagram struktural kelompok yang umum digunakan dalam proses desain sistem [12].

3. Metode Penelitian

Menurut beberapa tinjauan yang telah dijelaskan, dapat disimpulkan bahwa JSON Web Token dapat mengamankan sebuah web service. Untuk membedakan penelitian ini dengan penelitian sebelumnya, penelitian ini akan menggunakan pengaktifan autentikasi token pada JWT sebagai metode pengamanan yang lebih optimal [13].

3.1. Tahapan Penelitian

Fokus penelitian ini adalah tahapan metode pengembangan sistem yang menerapkan tahapan *Rapid Application Development* (RAD) dalam proses pengembangan sistem. Beberapa tahapan yang dilakukan dalam tahapan ini adalah sebagai berikut.



Gambar 3. Tahapan Penelitian

3.2. *Object* Penelitian

Dalam proses pengembangan sistem, perusahaan harus bekerja sama, menurut penelitian ini. Salah satu perusahaan yang mencetak undangan adalah CV Jastra Card, yang menjadi subjek penelitian ini. Kantor perusahaan berada di Teluk Baetung, Bandar Lampung.

3.3. Metode Pengumpulan Data

Studi pustaka adalah proses mengumpulkan data dengan membaca, mencatat, mengutip, dan menggunakan data dari buku, jurnal, dan sumber online lainnya.

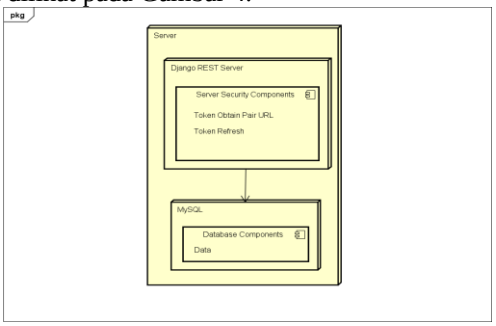
Wawancara adalah metode pengumpulan data di mana orang yang diwawancarai dan narasumber diberi pertanyaan yang telah disiapkan untuk dijawab.

Direksi CV Jastra Card diwawancarai untuk mendapatkan informasi tentang sistem yang akan dibangun.

Untuk pembangunan sistem, penelitian ini menggunakan metode pengembangan aplikasi cepat (RAD). Metode ini dianggap lebih mudah, tidak memakan waktu yang lama, dan mudah dibuat [10].

Perancangan sistem harus memenuhi kebutuhan pengguna dan memberikan gambaran yang jelas. *Unifield Modeling Languange* digunakan untuk tahap ini.

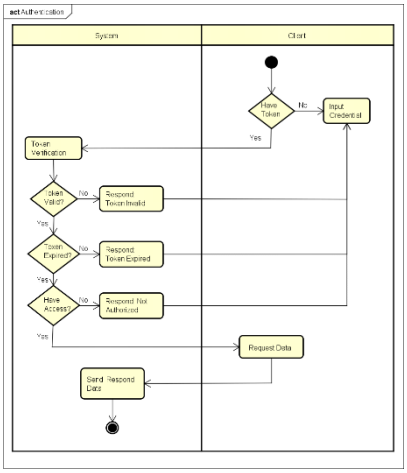
Dalam proses eksekusi sistem, konfigurasi komponen dikenal sebagai deployment diagram. Dalam penelitian ini, komponen server yang akan menerapkan keamanan bernama JWT dijelaskan. Diagram yang dibuat dapat dilihat pada Gambar 4.



Gambar 4. Deployment Diagram Back-End

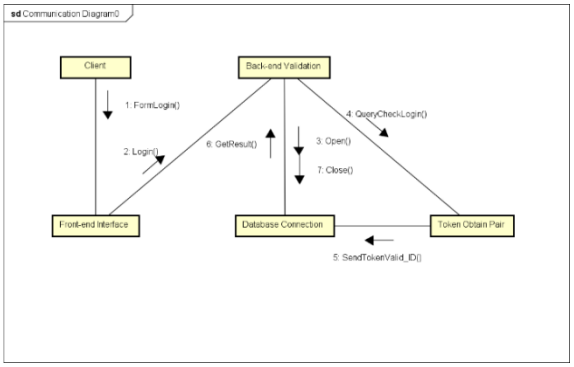
3.4. Activity Diagram

Activity diagram menggambarkan proses-proses yang terjadi dalam sistem dengan mengurutkan aliran aktivitas dari rancangan sistem atau aliran kerja yang akan dijalankan. Mereka juga digunakan untuk mengekelompokkan atau mendefinisikan aliran tampilan sistem. Gambar 5 menunjukkan diagram proses aktivitas sistem yang akan dikembangkan.



Gambar 5.. *Activity Diagram Authentication*

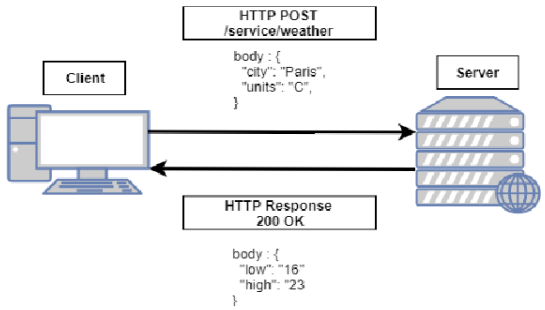
Diagram komunikasi ini menjelaskan bagaimana objek saling terhubung dan menjalankan proses sesuai dengan tugas yang akan dilakukan oleh pengguna. Proses login pengguna dapat digambarkan pada diagram komunikasi ini.



Gambar 6. *Communication Diagram Login*

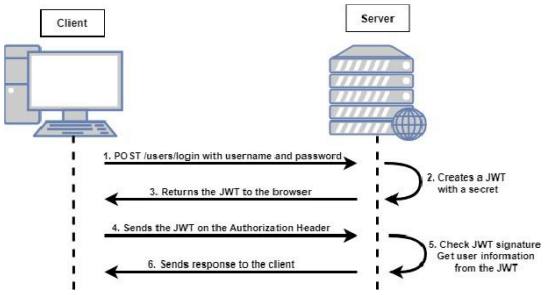
3.5. Arsitektur RESTful API

Dengan menggunakan arsitektur RESTful API, data dapat dikirim ke sistem menggunakan protokol HTTP seperti GET, POST, DELETE, dan Update. Data yang diubah diidentifikasi dengan Uniform Resource Locator (URL), yang berfungsi sebagai interface untuk mengubah sumber daya. Format pengiriman sumber daya dalam arsitektur REST dapat mencakup XML, HTML, JSON, dan format lainnya. dengan protokol HTTP/HTTPS yang tidak memiliki batas. Gambar 7 menunjukkan contoh pertukaran data.



Gambar 7. *Arsitektur RESTful API* [14]

Metode autentikasi menggunakan JWT melibatkan penyediaan data *credential client* kepada server untuk diidentifikasi. Setelah proses autentikasi maaka berhasil, sebuah token diberikan kepada klien untuk melakukan permintaan untuk sumber daya yang tersedia pada server. Gambar 8 menunjukkan mekanisme ini.



Gambar 8. Mekanisme Authorisasi Token JWT [14]

4. Hasil dan Pembahasan

4.1. Hasil Perancangan dan Pembahasan

Dalam bagian ini, langkah-langkah apa yang dilakukan dan aplikasi apa yang digunakan untuk membuat sistem RESTapi dijelaskan. Pembuatan RESTapi menggunakan metode autentikasi JWT menggunakan framework Rest Django yang digunakan dengan bahasa pemrograman Python. Komputer yang digunakan untuk pembuatan RESTapi ini memiliki spesifikasi perangkat keras (*hardware*) dan perangkat lunak (*software*), seperti yang ditunjukkan dalam tabel 1 berikut:

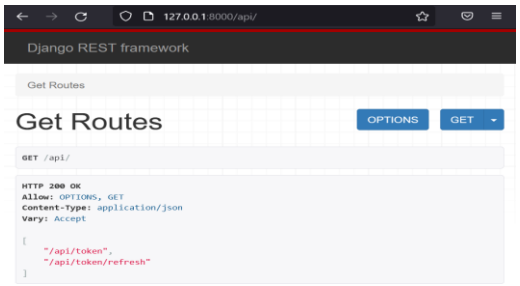
Tabel 1. Spesifikasi *Hardware* dan *Software* yang Digunakan

Spesifikasi	**Keterangan**
OS	Windows 10
Processor	Intel Celeron
Memori RAM	2GB DDR 3
Software	1. Visual Studio Code 2. Postman 3. XAMPP 4. SQLyog

Setelah spesifikasi *hardware* dan *software* dipenuhi, tahap selanjutnya adalah tahap pembuatan menggunakan perangkat lunak yang telah disiapkan, berikut beberapa tahapan penting dalam pembuatan aplikasi.

4.2. Hasil URL API Pada Django *REST Framework*

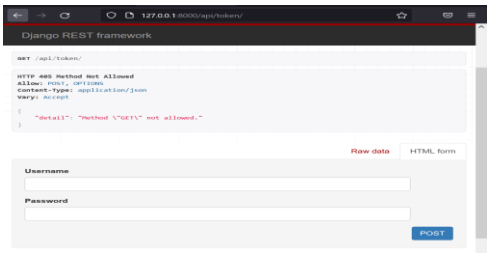
Setelah framework dipasang, langkah berikutnya adalah membuat pola URL pada urls.py. Pola pertama yang digunakan adalah API yang menuju Localhost/api/, yang memberikan tanggapan pada URL apa pun yang tersedia pada sisi server, seperti yang ditunjukkan pada Gambar 9. Dua URL yang ditampilkan pada halaman API adalah "api/token" dan "/api/token/."



Gambar 9. Django *REST Framework*

4.3. Token Obtain Pair URL

Jika user menggunakan metode POST, server dapat mengirimkan tanggapan kepada user melalui URL ini. Metode POST ini dilakukan saat user masuk ke sisi Front-End atau Web-Client. Tampilan URL pada sisi server ditunjukkan pada Gambar 10.



Gambar 10. Token Obtain Pair

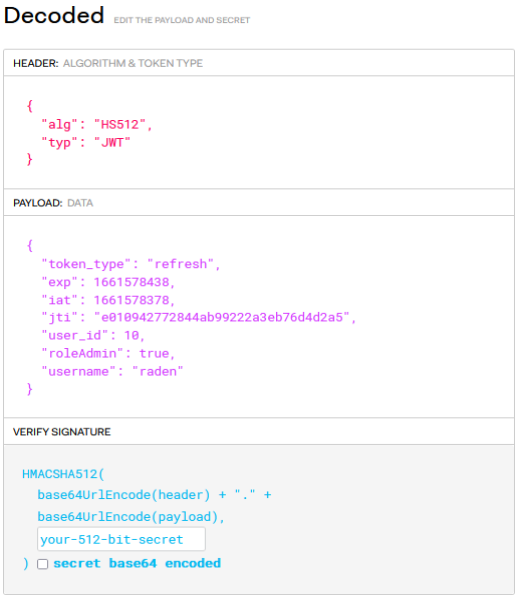
4.4. Halaman pada URL Token Refresh

Jika seseorang memiliki token, mereka dapat mengaksesnya melalui URL ini dengan mengirimkan refresh token. Tujuan dari refresh token ini adalah untuk memastikan bahwa token yang dimiliki user tetap berubah sesuai dengan konfigurasi client, sehingga orang yang mencurinya tidak dapat menggunakannya. Gambar 11 menunjukkan tampilan URL Update Token.

Gambar 11. Tampilan URL Token Refresh

4.5. Isi Dari Token

Token dibuat dengan algoritma hashing HMAC-SHA512 atau disingkat HS512. Token ini terdiri dari tiga bagian: header, payload, dan signature. Header menunjukkan jenis enkripsi dan algoritma yang digunakan. Karena isi payload sangat mudah untuk dilihat, sangat penting untuk tidak menampilkan data sensitif seperti password pengguna. Isi payload yang dikembangkan termasuk "token_type" sebagai penunjuk tipe token. "exp" menunjukkan kapan token akan habis masa pakainya, "iat" menunjukkan waktu di mana token dibuat, dan "exp" dan "iat" dihitung menggunakan kalkulasi timedelta yang tersedia di library Python untuk menentukan tanggal dan jam. Selain itu, ada "jti", yang menunjukkan identitas token; identitas ini berfungsi sebagai standar untuk melakukan blacklist token. Data pengguna "user_id", "usename", dan "roleAdmin" juga ditemukan. Bagian terakhir dari token, yang berfungsi sebagai kunci utama untuk token JWT, memiliki kunci rahasia yang hanya dimiliki oleh server. Peretas tidak akan dapat mengubah isi token tanpa tanda tangan. Gambar 12 menunjukkan isi token.



Gambar 12. Isi dari Token

4.6. Pengujian Program

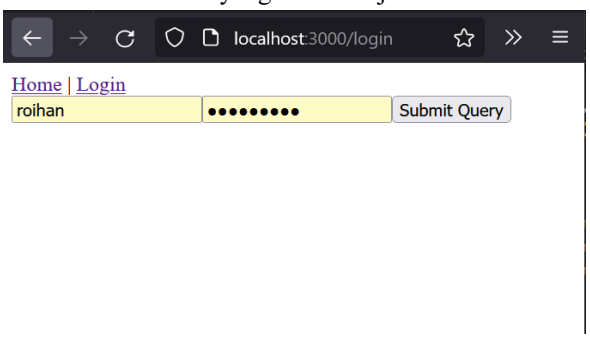
Pengujian dilakukan untuk menunjukkan hasil dari sistem yang telah dibuat. Pada tahap pengujian ini, beberapa langkah percobaan peretasan dilakukan untuk memberikan gambaran dan penjelasan alur kerja sistem REST dengan JWT. Selain itu, proses ini digunakan untuk mengetahui apakah sistem sudah dapat digunakan pada sisi Front-End atau bagian Client.

Dalam penelitian ini, percobaan peretasan dilakukan dalam dua (dua) cara: akses login tanpa akun dan mengubah isi data token. Ada pengujian tambahan yang menjelaskan proses refresh token.

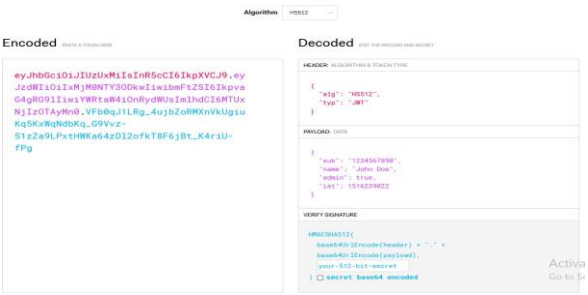
4.7. Percobaan Akses Login Tanpa Akun Yang Terverifikasi

Dalam penelitian ini, percobaan peretasan dilakukan dalam dua (dua) cara: akses login tanpa akun dan mengubah isi data token. Ada pengujian tambahan yang menjelaskan proses refresh token.

Eksperimen ini dilakukan di front-end dengan menggunakan server React JS di localhost:3000. Untuk mengakses halaman index di situs web ini, Anda harus melakukan login. Jadi, pada awal membuka, Anda akan diarahkan ke URL login. Percobaan pertama melibatkan pembuatan token acak yang dibuat di jwt.io.

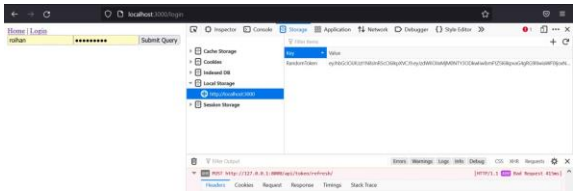


Gambar 13. Halaman Index Langsung Mengarahkan ke Login



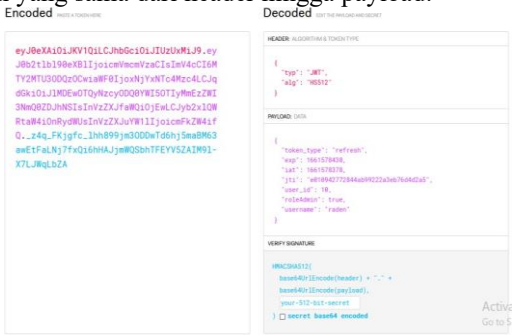
Gambar 14. Membuat Random Token dari Website JWT.IO

Karena random token tidak memiliki data yang tepat dari server, percobaan pengaksesan dilarang. Walau ada token di localstorage, sistem masih menolak akses user ke halaman index. Meskipun token tersimpan, Anda masih perlu melakukan login, seperti yang ditunjukkan pada Gambar 15.

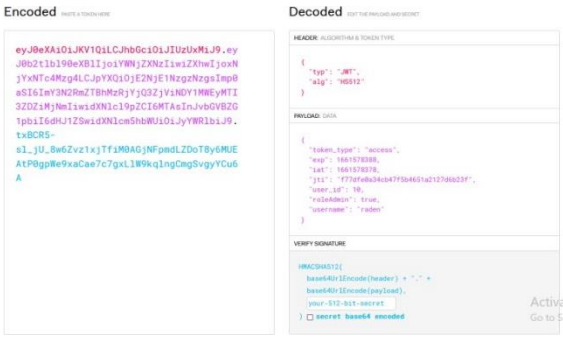


Gambar 15. Random Token Tersimpan Pada Local Storage

Eksperimen akses menggunakan token dengan data yang serupa dengan format yang diberikan server, dengan nama kunci bearer "authTokens" dan menggunakan isi token yang sama dari header hingga payload.

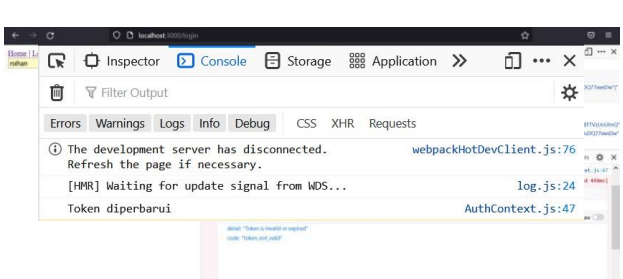


Gambar 16. Refresh Token Yang Dibuat Manual



Gambar 17. Access Token Yang Dibuat Manual

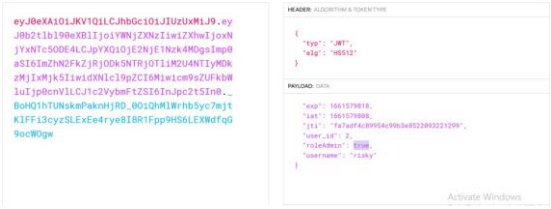
Karena token yang dibuat manual tidak dibuat oleh servernya sendiri dan tidak memiliki kunci tanda tangan yang ada pada server, token yang dibuat manual tidak valid meskipun memiliki isi yang sama. Oleh karena itu, meskipun data token sama dengan yang diatur oleh server, akses login tetap tidak valid. Meskipun token digunakan secara manual, web masih menolak, seperti yang ditunjukkan pada Gambar 18.



Gambar 18. Penolakan Token Yang Dibuat Manual

4.8. Percobaan Merubah Isi Data Pada Token

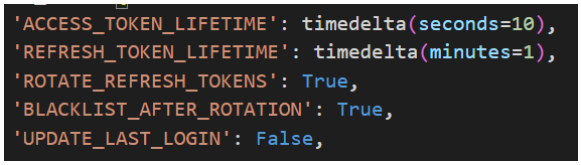
Percobaan mengubah data payload token. Ini dilakukan untuk menunjukkan apakah token akan tetap sah jika data diubah secara manual, yang dapat menyebabkan user menyalahgunakan akses mereka. Ada 1 (satu) perubahan pada data token, yaitu dari roleAdmin : False menjadi roleAdmin : True.



Gambar 19. Payload Token Asli

4.9. Pengujian *Token Access Lifetime*

Pengujian dilakukan dengan mengatur Access_Token_Lifetime menjadi 10 detik dan Refresh_Token_Lifetime menjadi 1 menit pengaturan ini dilakukan pada settings.py yang dapat dilihat pada Gambar 20.

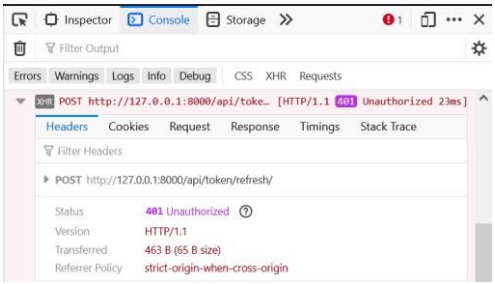


```
'ACCESS_TOKEN_LIFETIME': timedelta(seconds=10),
'REFRESH_TOKEN_LIFETIME': timedelta(minutes=1),
'ROTATE_REFRESH_TOKENS': True,
'BLACKLIST_AFTER_ROTATION': True,
'UPDATE_LAST_LOGIN': False,
```

Gambar 20. Pengaturan Access Lifetime Pada

Refresh_Token_Lifetime digunakan untuk memperbarui Access_Token, sedangkan Access_Token_Lifetime menunjukkan berapa lama Access_Token Valid untuk digunakan. Selama Refresh Token masih valid atau belum expired, klien dapat terus memperbarui Access Token. Percobaan ini melibatkan login ke halaman web client dan menunggu sepuluh detik. Setelah itu, pengguna mencoba merefresh halaman web client. Hasilnya adalah bahwa web tetap memungkinkan pengguna mengakses halaman Home, tetapi ada perubahan pada token yang disimpan sebelumnya di lokasi. Ini disebabkan oleh fakta bahwa Access Token telah habis masa berlakunya. Akibatnya, sistem web client melakukan permintaan ulang dengan mengirimkan Refresh Token yang masih berfungsi. Log diterapkan untuk membuktikan. Ini dapat dilihat di console browser saat ini.

Dalam langkah kedua, pengguna menunggu satu menit untuk melihat hasil setelah masa berlaku Refresh Token habis. Karena batas waktu penggunaan Refresh Token hanya 1 menit, jika user tidak melakukan pembaruan token dalam waktu tersebut, website otomatis mengeluarkan user karena token tidak lagi valid. Karena itu, jika user tidak melakukan pembaruan token dalam waktu tersebut, server otomatis mem-blacklist Refresh Token yang sudah expired. Gambar 21 menunjukkan bahwa respons server adalah 401, yang berarti Unauthorized.



Gambar 21. Respon Server Saat Refresh Token Sudah Expired

5. Kesimpulan dan Saran

Hasil diskusi menunjukkan bahwa Django Rest Framework berhasil merancang sistem REST API, dan metode pengembangan aplikasi cepat digunakan. Metode pengamanan JSON Web Token mengirimkan token ke server dan kemudian disimpan di localstorage pada sisi klien. Algoritma HMAC-SHA 512 atau HS512 dapat digunakan untuk mendekode token yang dikirim. Token yang diterima untuk pengujian program memiliki kemampuan untuk melindungi bagian validasi akun sebagai hak akses login. Karena refreshing token memiliki kemampuan untuk memberikan akses selama masa hidup token dan memiliki kemampuan untuk memasukkan token yang sudah expired ke dalam blacklist, penggunaan refreshing token lebih efektif daripada akses token biasa.

Beberapa rekomendasi yang dapat diberikan pada akhir penelitian laporan skripsi ini adalah sebagai berikut: diharapkan untuk menerapkan keamanan JWT ini pada situs web client Jastra Card; diharapkan untuk penelitian lanjutan dapat menerapkan keamanan tambahan untuk enkripsi pada bagian payload; dan diharapkan untuk penelitian lanjutan dapat menambahkan algoritma seperti RS256, RS384, atau yang lainnya.

Daftar Pustaka

[1] Le, P. (2022). “Development of an eCommerce website for Ngoc‟s MaxiNutri Company‟. Available at: http://www.theseus.fi/handle/10024/750396.

[2] Setiawan, A., Lusanjaya, G. and Kurnia, T. (2019). “Kesadaran Keamanan Privasi Dan Masyarakat 5.0‟, Journal of Accounting and Business Studies, 4(2), pp. 1–12.

[3] Rosyada, A. (2021). “Sistem Keamanan Web Service (RESTful API) pada JSON Web Token untuk Mengukur Aathentication dan Au- Thorization dengan Hashing Algoritma HMAC SHA-512‟.

[4] Rahmatulloh, A., Sulastri, H. and Nugroho, R. (2018). “Keamanan RESTful Web Service Menggunakan JSON Web Token (JWT) HMAC SHA-512‟, Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI), 7(2). doi: 10.22146/jnteti.v7i2.417.

[5] Peyrott, S. (2022) Introduction to JSON Web Tokens. Available at: https://jwt.io/introduction.

[6] Abdurrohim, I. (2019). “Penetration Testing Sistem Keamanan Aplikasi Web Berbasis e-Commerce Pada Perusahaan Hptasik‟, Jurnal Ilmu Komputer, 1(March), pp. 125–131.

[7] Gunawan, R. and Rahmatulloh, A. (2019). “JSON Web Token (JWT) untuk Authentication pada Interoperabilitas Arsitektur berbasis RESTful Web Service‟, Jurnal Edukasi dan Penelitian Informatika (JEPIN), 5(1). doi: 10.26418/jp.v5i1.27232.

[8] Rajagukguk, R. C. (2018). “Penggunaan Kriptografi pada JWT (JSON Web Token) dalam Implementasi Keamanan API‟.

[9] Wijaya, F., Jacobus, A. and Sambul, A. (2021). “Implementation Of Web Services On University Library Information Systems‟, Jurnal Teknik Informatika, 16(4), pp. 421–428. Available at: https://ejournal.unsrat.ac.id/index.php/informatika/article/download/34226/33732.

[10] Sagala, J. R. (2018). “Model Rapid Application Development (RAD) Dalam Pengembangan Sistem Informasi Penjadwalanbelajar Mengajar‟, Jurnal Mantik Penusa, 2(1), pp. 87–90.

[11] Tedyyana, A., Fauzi, M. and Ratnawati, F. (2021). “Revamp Keamanan Web Service Milik PT XYZ Menggunakan REST API‟, Digital Zone: Jurnal Teknologi Informasi dan Komunikasi, 12(1), pp. 1-10. doi: 10.31849/digitalzone.v12i1.6378.

[12] Rosa, A. S. and Shalahuddin, M. (2018). “Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek)”.

[13] Dykstra, D., Altunay, M. and Teheran, J. (2021). “Secure Command Line Solution for Token-based Authentication‟, EPJ Web of Conferences, 251, p. 02036. doi: 10.1051/epjconf/202125102036.

[14] Satria, B., Kusyanti, A. and Yahya, W. (2018). “Implementasi Algoritme Blake2s pada JSON Web Token (JWT) sebagai Algoritme Hashing untuk Mekanisme Autentikasi Layanan REST-API‟, Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK) Universitas Brawijaya, 2(12).