

OTENTIKASI MULTI-FAKTOR

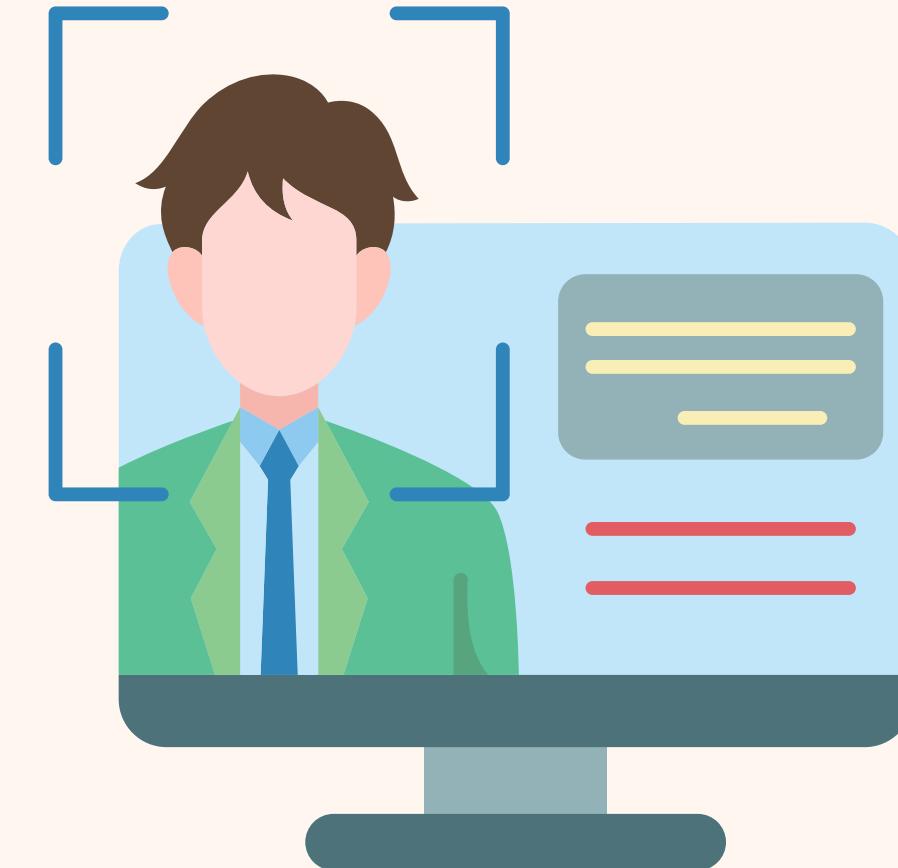
By . Ludang prasetyo .N
Nim 225510017



Apa itu ...?

Otentikasi Multi-Faktor (MFA)

Ini adalah suatu metode verifikasi keamanan yang menggunakan lebih dari satu cara untuk memverifikasi identitas seseorang Untuk memberikan keamanan ekstra selain menggunakan kata sandi yang rentan untuk di bobol



Cara kerjanya

Otentikasi Multi-Faktor (MFA) dengan meminta pengguna untuk melewati beberapa tahap verifikasi identitas sebelum diberikan akses ke suatu sistem atau layanan.

Tahapan verifikasi

① Pengguna Memasukkan Kredensial Pertama

Pengguna pertama-tama memasukkan informasi yang biasa digunakan untuk login, seperti username dan kata sandi.



2 Verifikasi Faktor Kedua

Ini biasanya setelah memasukan sandi biasanya user di suruh memasukan code “OTP” Yang akan di kirimkan melalui Nomor hp & Email

Biometrik, seperti sidik jari atau pengenalan wajah. Ini adalah contoh sesuatu yang kamu miliki secara biologis.

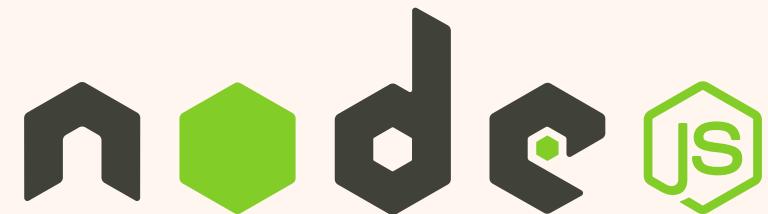
3 Akses Diberikan

Saat pengguna / user sudah memberikan verifikasi (OTP ,Biometrik, seperti sidik jari atau pengenalan wajah.) Maka akan langsung di berikan akses untuk masuk

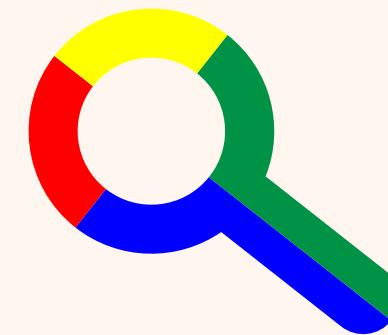


Tehnik Peorograman

Dalam pemrograman otentifikasi Otentikasi Multi-Faktor (MFA) Biasanya menggunakan Bahasa pemrograman Node.JS dan Google Authenticator sebagai aplikasi otentifikasi berbasis waktu (Time-Based OTP)



Node.JS



Google Authenticator



Penerapan dalam Programan

**Langkah- langkah dasar Implementasi Otentikasi Multi-Faktor(MFA)
untuk memverifikasi code OTP Untuk Dari email untuk login/Masuk
kedalam web,instansi pemerintah,aplikasi,Dll**

**Kalian juga membutuhkan librari yang harus terpasang untuk menyediakan cara
untuk mengirim email dari server.**

API !

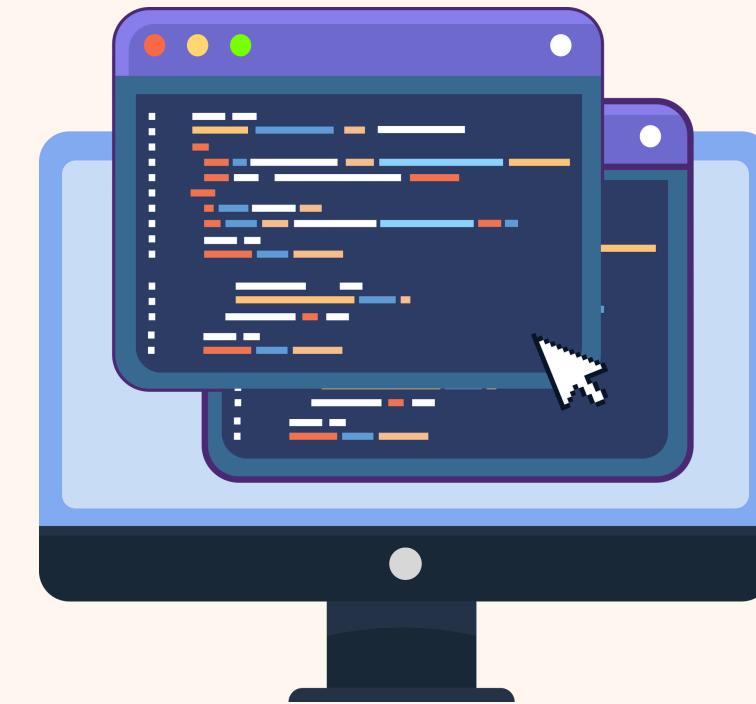
API dibuat secara langsung menggunakan Express.js, yang merupakan framework untuk Node.js

Librari !

Kalian bisa install librari yang di butuhkan seperti (speakeasy)



IMPLEMENTASI CODE



1 Memasukan Librari yang sudah di pilih

```
1 const nodemailer = require('nodemailer');
```

Ini adalah code js mempunyai fungsi mengirimkan email ke dalam server



2

Membuat Transporter untuk Mengirim Email

```
1 const transporter = nodemailer.createTransport({
2   service: 'gmail', // Kamu bisa menggunakan layanan email lain (Yahoo, Outlook, dll)
3   auth: {
4     user: 'Ludang@admin.id',
5     pass: 'password emaill' // Gunakan app password jika menggunakan Gmail
6   }
7});
```

Transporter adalah objek yang digunakan untuk mengirim email. Kita perlu mengatur informasi autentikasi untuk email pengirim.

- **service:** Menentukan layanan email yang akan digunakan. Contoh di atas menggunakan Gmail.
- **auth:** Bagian ini berisi email dan password akun pengirim. Jika menggunakan Gmail, disarankan untuk menggunakan App Password untuk alasan keamanan.



3 Function Mengirim OTP ke Email Pengguna

```
1 function sendVerificationEmail(toEmail, otpCode) {  
2     const mailOptions = {  
3         from: 'your-email@gmail.com',  
4         to: toEmail,  
5         subject: 'Your OTP Code',  
6         text: `Your OTP code is: ${otpCode}`  
7     };  
8  
9     transporter.sendMail(mailOptions, (error, info) => {  
10        if (error) {  
11            console.log(error);  
12        } else {  
13            console.log('Email sent: ' + info.response);  
14        }  
15    });  
16}
```

- **sendVerificationEmail():** Fungsi ini mengambil dua parameter, `toEmail` (email penerima) dan `otpCode` (kode OTP yang dihasilkan).
- **mailOptions:** Berisi detail email yang akan dikirim, seperti pengirim (`from`), penerima (`to`), subjek (`subject`), dan isi pesan (`text`).
- **transporter.sendMail():** Mengirim email berdasarkan opsi yang ditentukan dan mencatat jika email berhasil dikirim atau ada error.



4 Cara OTI itu di buat

Kode ini menghasilkan OTP (One-Time Password), yaitu kode verifikasi yang unik dan digunakan satu kali saja.

```
1 function generateOTP() {  
2     return Math.floor(100000 + Math.random() * 900000); // Generate 6-digit OTP  
3 }
```

- **generateOTP():** Menghasilkan kode verifikasi 6 digit. Formula ini membuat angka acak antara 100000 dan 999999 sehingga menghasilkan OTP 6 digit.
- **Math.random():** Menghasilkan angka acak, dan **Math.floor()** membulatkannya ke bawah untuk memastikan nilai integer.



5 Verifikasi OTP yang Dimasukkan Pengguna

```
1 app.post('/verify-otp', (req, res) => {
2     const userOTP = req.body.otp; // OTP yang dimasukkan pengguna
3
4     if (parseInt(userOTP) === storedOTP) {
5         res.send('OTP verified. Access granted.');
6     } else {
7         res.status(401).send('Invalid OTP. Access denied.');
8     }
9 });
10
```

- **/verify-otp:** Endpoint API untuk verifikasi OTP.
- **userOTP:** Kode OTP yang dikirimkan oleh pengguna dari form atau request.
- **storedOTP:** OTP yang disimpan sementara oleh server (misalnya di memori atau database) ketika OTP dikirim melalui email.
- **parseInt(userOTP):** Mengonversi kode OTP yang dimasukkan dari bentuk string ke integer untuk dibandingkan dengan OTP yang disimpan.
- **res.send():** Mengirim respons ke pengguna jika OTP valid dan otentikasi berhasil.
- **res.status(401).send():** Mengirim status 401 Unauthorized jika OTP yang dimasukkan tidak cocok.



6 Menyimpan OTP Sementara

```
1 let storedOTP; // Tempat sementara untuk menyimpan OTP (bisa di database)  
2
```

storedOTP: Variabel ini menyimpan kode OTP yang dihasilkan oleh `generateOTP()`. Dalam skenario yang lebih aman, OTP ini seharusnya disimpan di database dengan batas waktu penggunaan.



7 Mengirim OTP saat Login atau Registrasi

Pada bagian ini, OTP dikirim setelah pengguna mengirimkan permintaan untuk memulai otentikasi MFA, seperti saat login atau registrasi.

```
1 app.post('/send-otp', (req, res) => {
2   const email = req.body.email;
3   const otpCode = generateOTP(); // Generate OTP
4   storedOTP = otpCode; // Simpan OTP untuk verifikasi nanti
5   sendVerificationEmail(email, otpCode); // Kirim email
6   res.send('OTP sent to your email');
7 });


```

- **/send-otp:** Endpoint API yang digunakan untuk mengirim OTP ke email pengguna.
- **req.body.email:** Mengambil email yang dikirim oleh pengguna untuk mengirimkan OTP.
- **generateOTP():** Memanggil fungsi untuk menghasilkan OTP baru.
- **storedOTP:** Menyimpan OTP yang dihasilkan dalam variabel global (atau seharusnya di database dalam aplikasi yang lebih besar).
- **sendVerificationEmail():** Mengirimkan OTP ke email pengguna.



8 Menerima Request POST dan JSON

Untuk menerima data dari form atau aplikasi frontend, kita perlu menambahkan middleware yang memungkinkan aplikasi menerima JSON.

```
1 const express = require('express');
2 const app = express();
3 app.use(express.json());
```

- **express.json():** Middleware untuk mem-parsing JSON dari request body, yang digunakan saat mengirim data (seperti email dan OTP) dari frontend ke backend.



KELEBIHAN

Keamanan yang Ditingkatkan

- MFA menambah lapisan perlindungan ekstra. Bahkan jika kata sandi bocor atau dicuri, penyerang masih memerlukan faktor tambahan (seperti OTP atau biometrik) untuk mengakses akun.

Mengurangi Risiko Pembobolan

- Dengan menerapkan MFA, risiko akses tidak sah ke akun berkurang secara signifikan, yang penting untuk melindungi data sensitif, terutama pada aplikasi perbankan atau layanan kesehatan.

Meningkatkan Kepercayaan Pengguna

- Pengguna cenderung merasa lebih aman menggunakan layanan yang menerapkan MFA, karena mereka tahu ada langkah tambahan untuk melindungi informasi pribadi mereka.



KEKURANGAN

Ketersediaan Teknologi

- Tidak semua pengguna memiliki akses yang konsisten ke perangkat atau aplikasi yang diperlukan untuk otentikasi, seperti smartphone untuk menerima OTP atau aplikasi lainnya.

Risiko Kehilangan Akses

- Jika pengguna kehilangan perangkat yang digunakan untuk otentikasi (misalnya, ponsel untuk menerima OTP), mereka mungkin tidak bisa mengakses akun mereka. Proses pemulihan akses bisa rumit dan memakan waktu.

Biaya Implementasi

- Untuk organisasi, menerapkan MFA bisa memerlukan investasi dalam perangkat keras, perangkat lunak, atau infrastruktur yang mungkin tidak terjangkau untuk semua perusahaan, terutama bisnis kecil.



CONTOH LYANAN YANG MEMAKAI (MFA)

- Google (Gmail)
- Microsoft (Akun Microsoft)
- Facebook
- Twitter
- Amazon Web Services (AWS)
- Bank Mandiri
- Bank BCA
- Tokopedia
- Slack
- Dropbox
- Salesforce
- PayPal



KESIMPULAN



MFA adalah langkah penting untuk meningkatkan keamanan, tetapi harus diimbangi dengan kemudahan penggunaan dan aksesibilitas. Organisasi perlu mempertimbangkan baik keuntungan maupun kerugian saat merancang sistem otentikasi mereka agar efektif dan nyaman bagi pengguna.

Daftar pustaka



<https://www.fraud.com/post/multi-factor-authentication>

<https://stytch.com/blog/how-to-enforce-multi-factor-authentication-with-node-js/>



SILAHKAN BERTANYA



utdi.ac.id

