**PENGEMBANGAN WEB (TEORI)**

**LAPORAN EKSPERIMEN MENGENAI PROBLEM DIRECTORY TRAVERSAL DENGAN PENYELESAIAN MENGGUNAKAN SOLUSI FUNGSI API**

*Laporan ini disusun untuk memenuhi tugas 1 mata kuliah Pengembangan Web (Teori)*



Disusun oleh kelompok B4:

Asri Husnul Rosadi 221524035

Faris Abulkhoir 221524040

Mahardika Pratama 221524044

Muhamad Fahri Yuwan 221524047

Najib Alimudin Fajri 221524053

Septyana Agustina 221524058

Sarah 221524059

Dosen Pengampu:

Joe Lian Min, M.Eng.

**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA**

**PROGRAM STUDI D4 TEKNIK INFORMATIKA**

**POLITEKNIK NEGERI BANDUNG**

**2024**

# **DAFTAR ISI**

[DAFTAR ISI i](#_Toc175623184)

[A. IDENTIFIKASI PROBLEM 1](#_Toc175623185)

[B. DESKRIPSI PROBLEM 1](#_Toc175623186)

[C. METODOLOGI EKSPERIMEN 1](#_Toc175623187)

[D. PELAKSANAAN EKSPERIMEN 3](#_Toc175623188)

[E. ANALISIS HASIL EKSPERIMEN 5](#_Toc175623189)

# **IDENTIFIKASI PROBLEM**

Directory Traversal adalah jenis serangan yang memungkinkan penyerang untuk mengakses file di luar direktori yang diizinkan dalam aplikasi web. Ini dapat menyebabkan kebocoran data sensitif, akses tidak sah, dan kerentanan keamanan lainnya.

# **DESKRIPSI PROBLEM**

Directory Traversal terjadi ketika parameter input, seperti nama file dalam URL atau query string, dimanipulasi untuk mengakses file di luar direktori yang diizinkan oleh aplikasi. Serangan ini biasanya memanfaatkan karakter seperti ../ untuk berpindah ke direktori yang lebih tinggi dan mengakses file yang tidak seharusnya dapat diakses. Untuk melindungi aplikasi dari serangan ini, penting untuk membatasi akses file hanya dalam direktori yang diizinkan.

Contoh masalah yang bisa muncul:

* Mengakses file konfigurasi sensitif seperti config.json yang terletak di luar folder yang dimaksud.
* Mengakses file yang tidak seharusnya terlihat oleh pengguna, seperti file log atau data pribadi.

# **METODOLOGI EKSPERIMEN**

* **Variabel Terikat**: Keamanan API terhadap serangan Directory Traversal (diukur dengan apakah file di luar direktori yang diizinkan dapat diakses atau tidak).
* **Variabel Bebas**: Implementasi sanitasi path dan verifikasi direktori.
* **Metode Pengukuran**: Menguji apakah API memungkinkan akses file di dalam direktori yang diizinkan dan mencegah akses file di luar direktori tersebut.
* **Skenario Pengujian:**
* **Pengujian Akses File yang Diizinkan**: Mengakses file yang berada dalam direktori uploads yang diizinkan untuk memastikan API berfungsi sebagaimana mestinya.
* **Pengujian Akses File yang Tidak Diizinkan**: Mencoba mengakses file di luar direktori uploads untuk menguji apakah API berhasil mencegah serangan Directory Traversal.
* **Alat dan Teknologi:**
* **Express.js**: Framework Node.js yang digunakan untuk membangun API.
* **Node.js**: Platform runtime JavaScript untuk menjalankan aplikasi Express.js.
* **Path Module (Node.js)**: Untuk sanitasi dan manipulasi jalur file.
* **File System (fs) Module (Node.js)**: Untuk memverifikasi dan mengirim file.
* **React.js**: Untuk membangun antarmuka pengguna di sisi klien.
* **Axios**: Untuk mengirim permintaan HTTP dari klien ke API.
* **Browser Web**: Untuk menguji API melalui URL dan mengamati hasilnya.
* **Langkah-langkah:**

1. **Pengaturan Kode API**:
   * Mengimplementasikan API dengan sanitasi path dan verifikasi direktori menggunakan modul path dan fs di Express.js.
   * Menentukan direktori uploads sebagai satu-satunya direktori yang diizinkan untuk akses file.
2. **Pengujian Akses File yang Diizinkan**:
   * Mengakses file test.txt yang berada di dalam direktori uploads melalui URL seperti http://localhost:5000/api/file?filename=test.txt.
   * Mengamati apakah konten file tersebut berhasil ditampilkan.
3. **Pengujian Akses File yang Tidak Diizinkan**:
   * Mencoba mengakses file coba.txt yang berada di luar direktori uploads melalui URL seperti http://localhost:5000/api/file?filename=../coba.txt.
   * Mengamati apakah API mencegah akses dan mengembalikan pesan error yang sesuai.
4. **Analisis Hasil**:
   * Membandingkan hasil pengujian untuk menentukan apakah sanitasi path dan verifikasi direktori berhasil mencegah akses yang tidak diizinkan.
   * Evaluasi efektivitas pendekatan ini dalam mencegah serangan Directory Traversal.

# **PELAKSANAAN EKSPERIMEN**

**Tahap 1: Setup Lingkungan**

1. **Menyiapkan Lingkungan Pengembangan**:
   * Pasang Node.js dan NPM di sistem untuk menjalankan server Express.js.
   * Buat direktori proyek yang mencakup struktur folder seperti uploads untuk menyimpan file yang akan diuji.
   * Instal dependensi yang diperlukan, termasuk express, body-parser, cors, path, dan fs.
2. **Pengaturan Kode API**:
   * Implementasikan API dengan sanitasi path dan verifikasi direktori di index.js.
   * Pastikan API hanya mengizinkan akses file yang berada dalam direktori uploads.
3. **Menyiapkan Aplikasi Klien**:
   * Buat aplikasi React.js menggunakan create-react-app.
   * Implementasikan komponen yang memungkinkan pengguna memasukkan nama file dan mengirim permintaan ke API untuk mengakses file tersebut.
   * Gunakan axios untuk berkomunikasi dengan API dari aplikasi klien.
4. **Struktur File**

‘directory-traversal-experiment-API/’ (root folder)

* server/ (backend folder)
* client/ (frontend folder)
* index.js (server entry point)
* uploads/ (folder untuk menyimpan file yang bisa diakses)

**Tahap 2: Pengujian Akses File**

1. **File yang Diizinkan**:
   * **Langkah Pengujian**:
     + Simpan file test.txt dalam direktori uploads.
     + Akses file ini melalui URL http://localhost:3001/api/file?filename=test.txt di browser.
   * **Hasil yang Diharapkan**:
     + Konten file test.txt ditampilkan di halaman klien, menunjukkan bahwa API berhasil mengakses file dalam direktori yang diizinkan.
2. **File yang Tidak Diizinkan**:
   * **Langkah Pengujian**:
     + Letakkan file coba.txt di luar direktori uploads.
     + Coba akses file ini melalui URL http://localhost:3001/api/file?filename=../coba.txt di browser.
   * **Hasil yang Diharapkan**:
     + API menolak permintaan dan menampilkan pesan "Invalid file path", menunjukkan bahwa API berhasil mencegah serangan Directory Traversal.

**Tahap 3: Pengumpulan Data dan Observasi**

1. **Analisis Hasil Pengujian**:
   * **Pengujian Akses File yang Diizinkan**:
     + Observasi apakah konten file test.txt berhasil ditampilkan.
     + Jika API mengembalikan konten file dengan benar, ini menunjukkan bahwa file yang berada dalam direktori yang diizinkan dapat diakses dengan aman.
   * **Pengujian Akses File yang Tidak Diizinkan**:
     + Observasi apakah permintaan untuk ../coba.txt menghasilkan pesan error "Invalid file path".
     + Jika API mengembalikan pesan error, ini menunjukkan bahwa API berhasil mencegah akses ke file di luar direktori uploads.
2. **Evaluasi Efektivitas API**:
   * Kumpulkan semua hasil pengujian untuk menilai apakah API efektif dalam mencegah akses file yang tidak diizinkan melalui sanitasi dan verifikasi path.
   * Analisis apakah ada potensi kerentanan yang tersisa atau apakah pendekatan ini cukup untuk mencegah Directory Traversal.

# **ANALISIS HASIL EKSPERIMEN**

**Hasil Pengujian**:

1. **File yang Diizinkan**:
   * **Hasil**: Konten file test.txt ditampilkan dengan benar.
   * **Analisis**: Sistem berhasil mengakses file yang berada di dalam direktori uploads sesuai dengan yang diharapkan.
2. **File yang Tidak Diizinkan**:
   * **Hasil**: Pesan "Invalid file path" ditampilkan.
   * **Analisis**: Sistem berhasil mencegah akses ke file di luar direktori uploads, menunjukkan bahwa sanitasi path dan verifikasi direktori berfungsi dengan baik.

**Pros**:

* **Keamanan yang Ditingkatkan**: Menggunakan sanitasi dan verifikasi path mencegah akses ke file di luar direktori yang diizinkan, melindungi data sensitif.
* **Kesederhanaan Implementasi**: Kode yang disediakan relatif sederhana dan mudah diintegrasikan ke dalam aplikasi yang ada.

**Cons**:

* **Keterbatasan Sanitasi Path**: Meskipun sanitasi path dapat mengatasi sebagian besar kasus Directory Traversal, penanganan yang lebih canggih mungkin diperlukan untuk kasus yang lebih kompleks.
* **Pengujian yang Terbatas**: Pengujian hanya dilakukan dengan dua skenario; pengujian lebih mendalam dengan berbagai jenis path dan kondisi bisa memberikan gambaran yang lebih lengkap.

**Kesimpulan:**

Hasil eksperimen menunjukkan bahwa pendekatan yang digunakan untuk mencegah Directory Traversal dalam API ini berhasil dengan baik. Sistem mampu mengizinkan akses hanya ke file yang berada dalam direktori uploads dan mencegah akses ke file di luar direktori tersebut melalui sanitasi dan verifikasi path.

Keberhasilan dalam pencegahan ini menunjukkan bahwa metode sanitasi path yang diterapkan cukup efektif untuk skenario yang diuji. Namun, keterbatasan sanitasi path dan pengujian yang terbatas menandakan bahwa untuk aplikasi yang lebih kompleks, diperlukan langkah tambahan dan pengujian lebih mendalam untuk memastikan keamanan menyeluruh.