**PENGEMBANGAN WEB (TEORI)**

**LAPORAN EKSPERIMEN MENGENAI PROBLEM DIRECTORY TRAVERSAL DENGAN PENYELESAIAN MENGGUNAKAN SOLUSI WHITELISTING**

*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](#_Toc175625547)

[A. IDENTIFIKASI PROBLEM 1](#_Toc175625548)

[B. DESKRIPSI PROBLEM 1](#_Toc175625549)

[C. METODOLOGI EKSPERIMEN 1](#_Toc175625550)

[D. PELAKSANAAN EKSPERIMEN 3](#_Toc175625551)

[E. ANALISIS HASIL EKSPERIMEN 5](#_Toc175625552)

# **IDENTIFIKASI PROBLEM**

Serangan Directory Traversal adalah salah satu jenis serangan keamanan yang memungkinkan penyerang untuk mengakses file yang seharusnya tidak dapat diakses oleh pengguna biasa. Serangan ini terjadi ketika input dari pengguna tidak divalidasi dengan baik, sehingga memungkinkan akses ke direktori dan file di luar cakupan yang diizinkan oleh aplikasi. Tujuan dari eksperimen ini adalah untuk mencegah akses ilegal tersebut dengan menerapkan teknik Whitelisting pada aplikasi.

# **DESKRIPSI PROBLEM**

Directory Traversal, atau dikenal juga sebagai Path Traversal, terjadi ketika aplikasi web memungkinkan pengguna untuk menavigasi melalui direktori file sistem menggunakan input seperti ../. Ini memungkinkan penyerang untuk mengakses file di luar direktori yang diizinkan, seperti file konfigurasi sistem atau data pengguna lainnya yang sensitif.

Sebagai contoh, dalam aplikasi yang dirancang untuk mengakses file yang disimpan dalam folder uploads, penyerang dapat mencoba mengakses file di luar folder ini dengan memasukkan jalur yang tidak diharapkan, seperti http://localhost:3001/api/file?filename=../coba.txt. Jika tidak ada mekanisme keamanan yang memadai, file di luar folder uploads bisa diakses, yang dapat menyebabkan kebocoran data atau akses yang tidak sah ke informasi sensitif.

# **METODOLOGI EKSPERIMEN**

**Variabel Terikat:** Keamanan Aplikasi**,** yaitu tingkat keamanan aplikasi dalam mencegah serangan Directory Traversal.

**Variabel Bebas:** Mekanisme Keamanan (Whitelisting), yaitu menggunakan atau tidak menggunakan Whitelisting untuk menentukan file mana yang dapat diakses oleh pengguna.

**Metode Pengukuran:**

* **Keberhasilan Akses File:** Mengukur apakah file yang diminta dapat diakses sesuai dengan aturan Whitelisting, serta apakah aplikasi mampu mencegah akses ke file yang tidak diizinkan.
* **Respon Aplikasi:** Mengamati dan mencatat respon aplikasi saat mencoba mengakses file yang diizinkan dan file yang tidak diizinkan.

**Skenario Pengujian:**

* **Pengujian pada File yang Diizinkan:** Mengakses file yang termasuk dalam daftar whitelist (misalnya, test.txt dan report.txt).
* **Pengujian pada File yang Tidak Diizinkan:** Mencoba mengakses file di luar daftar whitelist menggunakan teknik Directory Traversal (misalnya, ../coba.txt).

**Alat dan Teknologi:**

* **Node.js dan Express.js:** Untuk mengembangkan dan menjalankan server backend yang melayani permintaan file.
* **React.js dan Axios:** Untuk membuat antarmuka pengguna yang mengirimkan permintaan file ke server.
* **Modul fs dan path di Node.js:** Untuk mengakses dan mengelola file pada server, serta membangun jalur file yang aman.
* **Browser (misalnya, Chrome, Firefox):** Untuk mengakses aplikasi web dan mengirimkan permintaan file.

**Langkah-langkah:**

1. **Pengembangan Aplikasi Node.js dan Express:**
   * Buat server backend yang melayani permintaan file dari direktori tertentu (uploads).
   * Implementasikan endpoint API untuk mengelola permintaan file dari klien.
2. **Implementasi Whitelisting:**
   * Definisikan daftar file yang diizinkan (whitelist) dalam kode backend.
   * Modifikasi endpoint API untuk memeriksa apakah file yang diminta termasuk dalam whitelist sebelum mengizinkan akses.
3. **Pengujian Akses File Melalui URL:**
   * Kirim permintaan untuk mengakses file yang diizinkan menggunakan URL (misalnya, http://localhost:3001/api/file?filename=test.txt).
   * Coba akses file di luar daftar whitelist menggunakan teknik Directory Traversal (misalnya, http://localhost:3001/api/file?filename=../coba.txt).
   * Catat respon aplikasi untuk setiap permintaan.
4. **Analisis Hasil:**
   * Evaluasi apakah file yang diizinkan berhasil diakses dan apakah akses ke file yang tidak diizinkan berhasil dicegah.
   * Analisis efektivitas Whitelisting dalam mencegah serangan Directory Traversal berdasarkan hasil pengujian.

# **PELAKSANAAN EKSPERIMEN**

**Tahap 1: Setup Lingkungan**

1. **Siapkan Lingkungan Aplikasi:**
   * Aplikasi Node.js dikembangkan menggunakan Express sebagai framework. Aplikasi ini diatur untuk berjalan pada localhost dengan port 3001.
   * Direktori uploads digunakan sebagai lokasi penyimpanan file yang dapat diakses oleh pengguna melalui aplikasi.
   * Implementasi Whitelisting dilakukan dengan membuat daftar file yang diizinkan (test.txt dan report.txt), yang dapat diakses melalui endpoint API.
2. **Pastikan Kode Backend dan Klien Berfungsi:**
   * Kode backend telah diatur untuk melayani permintaan file dari klien melalui URL yang mengandung nama file sebagai parameter.
   * Aplikasi klien menggunakan React.js dan Axios untuk mengirim permintaan HTTP ke server dan menampilkan konten file yang diterima dari server.
3. **Struktur File:**

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

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

**Tahap 2: Pengujian Awal**

1. **Pengujian Akses File yang Diizinkan:**
   * Kirim permintaan HTTP GET untuk mengakses test.txt melalui URL: http://localhost:3001/api/file?filename=test.txt.
   * Kirim permintaan HTTP GET untuk mengakses report.txt melalui URL: http://localhost:3001/api/file?filename=report.txt.
   * Observasi dan catat bahwa file yang diizinkan berhasil diakses, dan kontennya ditampilkan pada halaman klien.
2. **Pengujian Akses File yang Tidak Diizinkan:**
   * Kirim permintaan HTTP GET untuk mencoba mengakses file di luar daftar whitelist menggunakan Directory Traversal: http://localhost:3001/api/file?filename=../coba.txt.
   * Observasi dan catat bahwa permintaan ini ditolak oleh server, dan respon yang diberikan adalah pesan kesalahan: "Access to this file is not allowed".

**Tahap 3: Implementasi dan Pengujian Whitelisting**

1. **Implementasi Whitelisting:**
   * Dalam aplikasi backend, implementasikan logika Whitelisting yang memeriksa apakah file yang diminta termasuk dalam daftar file yang diizinkan sebelum mengizinkan akses ke file tersebut.
   * Whitelisting dilakukan dengan mencocokkan nama file yang diminta dengan daftar file yang diizinkan (test.txt dan report.txt).
2. **Pengujian Akses File Setelah Implementasi Whitelisting:**
   * Ulangi pengujian akses file yang diizinkan (test.txt dan report.txt) dan catat bahwa file ini masih dapat diakses dengan sukses.
   * Ulangi pengujian akses file yang tidak diizinkan (../coba.txt) dan catat bahwa akses ini tetap ditolak dengan pesan kesalahan yang sesuai.

**Tahap 4: Pengujian Lanjutan dengan Modifikasi Whitelist**

1. **Penambahan File pada Whitelist:**
   * Tambahkan file baru (coba.txt) ke dalam daftar whitelist dan amati bagaimana aplikasi bereaksi terhadap perubahan ini.
   * Kirim permintaan HTTP GET untuk mengakses coba.txt melalui URL: http://localhost:3001/api/file?filename=coba.txt dan catat apakah file tersebut sekarang dapat diakses.
2. **Penghapusan File dari Whitelist:**
   * Hapus salah satu file (report.txt) dari daftar whitelist dan amati bagaimana aplikasi menangani permintaan akses terhadap file yang telah dihapus dari whitelist.
   * Kirim permintaan HTTP GET untuk mengakses report.txt dan catat apakah akses terhadap file tersebut sekarang ditolak.
3. **Evaluasi Dinamisitas Whitelisting:**
   * Analisis apakah perubahan pada daftar whitelist dapat diterapkan secara dinamis (tanpa perlu restart server) dan dampaknya terhadap keamanan serta fleksibilitas aplikasi.

# **ANALISIS HASIL EKSPERIMEN**

**Hasil Eksperimen:**

1. **Akses File yang Diizinkan:**
   * Ketika pengguna mencoba mengakses file yang termasuk dalam daftar whitelist, seperti test.txt dan report.txt, file tersebut berhasil diakses dan ditampilkan pada halaman klien tanpa masalah. Hal ini menunjukkan bahwa Whitelisting bekerja sesuai harapan dengan memungkinkan akses hanya kepada file yang telah ditentukan.
2. **Penolakan Akses ke File yang Tidak Diizinkan:**
   * Percobaan untuk mengakses file yang tidak diizinkan melalui teknik Directory Traversal, seperti ../coba.txt, ditolak oleh server dengan pesan kesalahan "Access to this file is not allowed". Ini membuktikan bahwa Whitelisting berhasil mencegah akses ke file yang berpotensi merusak atau mencuri informasi sensitif.
3. **Pengujian Lanjutan dengan Modifikasi Whitelist:**
   * Setelah coba.txt ditambahkan ke daftar whitelist, file tersebut bisa diakses, yang menunjukkan fleksibilitas Whitelisting dalam mengendalikan akses file. Sebaliknya, setelah report.txt dihapus dari whitelist, akses ke file ini juga ditolak, memperlihatkan bagaimana Whitelisting dapat digunakan untuk secara dinamis mengatur file mana yang diizinkan.

**Pros:**

* **Keamanan:** Dengan Whitelisting, akses ke file dibatasi hanya pada file yang diizinkan, sehingga potensi serangan Directory Traversal berhasil dicegah. Ini memastikan bahwa hanya file yang diidentifikasi aman yang bisa diakses oleh pengguna.
* **Kontrol:** Administrator memiliki kontrol penuh atas file mana yang dapat diakses oleh pengguna, yang membantu dalam mengelola keamanan aplikasi.
* **Simplicity:** Implementasi Whitelisting relatif sederhana dan tidak memerlukan perubahan besar dalam arsitektur aplikasi.

**Cons:**

* **Maintenance:** Daftar file yang diizinkan harus dikelola dan diperbarui secara manual. Jika aplikasi memiliki banyak file, ini bisa menjadi tugas yang memakan waktu.
* **Scalability:** Dalam aplikasi besar, terutama yang dinamis, Whitelisting mungkin tidak cukup fleksibel. Setiap file baru yang perlu diakses harus ditambahkan ke daftar whitelist secara manual.
* **User Experience:** Pengguna mungkin mengalami pembatasan yang ketat jika file yang dibutuhkan tidak ada dalam daftar whitelist, yang memerlukan koordinasi lebih lanjut untuk menambahkannya.

**Kesimpulan:**

Implementasi Whitelisting dalam aplikasi ini terbukti efektif dalam mencegah serangan Directory Traversal dengan membatasi akses file hanya kepada file yang diizinkan. Eksperimen ini menegaskan bahwa Whitelisting adalah solusi dalam meningkatkan keamanan aplikasi dengan memberikan kontrol penuh kepada administrator mengenai file apa saja yang dapat diakses oleh pengguna.