**LAPORAN EKSPERIMEN MENGENAI PERBANDING SESSION STORAGE DAN JWT UNTUK USER AUTHENTICATION**

*Laporan ini disusun untuk memenuhi tugas 3 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

Sarah 221524059

Septyana Agustina 221524060

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](#_Toc175471873)

[A. IDENTIFIKASI PROBLEM 1](#_Toc175471874)

[B. DESKRIPSI PROBLEM 1](#_Toc175471875)

[C. METODOLOGI EKSPERIMEN 1](#_Toc175471876)

[D. PELAKSANAAN EKSPERIMEN 2](#_Toc175471877)

[E. ANALISIS HASIL EKSPERIMEN 3](#_Toc175471878)

# **IDENTIFIKASI PROBLEM**

Dalam pengembangan aplikasi web, autentikasi pengguna merupakan aspek penting yang harus dijamin keamanannya. Dua metode umum untuk menangani autentikasi adalah menggunakan Session Storage dan JSON Web Token (JWT). Kedua metode ini memiliki kelebihan dan kekurangan masing-masing terkait keamanan, performa, dan kemudahan implementasi. Penting untuk menentukan metode mana yang lebih cocok untuk aplikasi tertentu berdasarkan pengujian yang dapat dihitung.

# **DESKRIPSI PROBLEM**

Session Storage dan JWT adalah dua pendekatan yang berbeda dalam mengelola sesi pengguna yang terautentikasi:

* **Session Storage** melibatkan penyimpanan sesi pengguna di server, biasanya dalam bentuk ID sesi yang disimpan di sisi klien (misalnya, dalam cookie). Saat pengguna melakukan permintaan, ID sesi dikirim ke server, dan server memverifikasi sesi tersebut.
* **JWT** adalah token berbasis JSON yang dienkripsi dan berisi klaim yang bisa diverifikasi tanpa memerlukan penyimpanan server. JWT disimpan di sisi klien dan dikirimkan pada setiap permintaan ke server. Verifikasi dilakukan menggunakan kunci rahasia yang hanya diketahui oleh server.

Masalah utama yang ingin dipecahkan adalah menentukan metode mana yang lebih efektif dan efisien berdasarkan kriteria berikut:

* **Keamanan:** Seberapa baik metode melindungi dari serangan seperti XSS, CSRF, dan replay attacks.
* **Performa:** Waktu respons, throughput, dan penggunaan memori.
* **Kemudahan Implementasi:** Kompleksitas pengaturan dan pemeliharaan sistem.

# **METODOLOGI EKSPERIMEN**

Untuk membandingkan Session Storage dan JWT, eksperimen dilakukan dengan mengukur beberapa metrik kinerja dan keamanan:

**Metrik yang Diukur:**

1. **Waktu Respons (Latency):** Mengukur kecepatan server dalam menanggapi permintaan autentikasi dan akses ke endpoint yang dilindungi.
2. **Konsumsi Memori:** Mengukur penggunaan memori server saat memproses permintaan dengan Session Storage dan JWT.
3. **Ukuran Payload:** Membandingkan ukuran data yang dikirimkan antara klien dan server.
4. **Throughput:** Mengukur jumlah permintaan yang dapat diproses oleh server dalam satuan waktu tertentu.
5. **Keamanan:** Menilai kerentanan terhadap berbagai jenis serangan, seperti XSS, CSRF, dan replay attacks.

**Langkah Pengujian:**

1. Menyiapkan dua server Node.js yang masing-masing mengimplementasikan Session Storage dan JWT untuk autentikasi.
2. Menggunakan Postman untuk mengirimkan permintaan ke server, mencatat waktu respons, dan ukuran payload.
3. Menggunakan alat seperti htop untuk mengukur konsumsi memori.
4. Menggunakan Apache Benchmark (ab) atau wrk untuk mengukur throughput server.
5. Melakukan simulasi serangan menggunakan OWASP ZAP untuk menilai keamanan kedua metode.

# **PELAKSANAAN EKSPERIMEN**

**Eksperimen 1: Session Storage**

* **Setup:** Server menggunakan penyimpanan sesi di memori untuk mencatat sesi pengguna.
* **Pengujian:**
  + **Login:** Mengirim 100 permintaan POST ke endpoint /login-session.
  + **Akses Dilindungi:** Mengirim 100 permintaan GET ke endpoint /protected-session.
* **Hasil:**
  + **Waktu Respons:** Dicatat rata-rata waktu respons untuk login dan akses.
  + **Konsumsi Memori:** Dicatat penggunaan memori selama pengujian.
  + **Ukuran Payload:** Dicatat ukuran payload yang dikirimkan.
  + **Throughput:** Dicatat jumlah permintaan per detik yang dapat diproses.
  + **Keamanan:** Melakukan simulasi serangan XSS dan CSRF.

**Eksperimen 2: JWT**

* **Setup:** Server menggunakan JWT untuk mengautentikasi pengguna dengan token yang dienkripsi.
* **Pengujian:**
  + **Login:** Mengirim 100 permintaan POST ke endpoint /login-jwt.
  + **Akses Dilindungi:** Mengirim 100 permintaan GET ke endpoint /protected-jwt.
* **Hasil:**
  + **Waktu Respons:** Dicatat rata-rata waktu respons untuk login dan akses.
  + **Konsumsi Memori:** Dicatat penggunaan memori selama pengujian.
  + **Ukuran Payload:** Dicatat ukuran payload yang dikirimkan.
  + **Throughput:** Dicatat jumlah permintaan per detik yang dapat diproses.
  + **Keamanan:** Melakukan simulasi serangan replay dan XSS.

# **ANALISIS HASIL EKSPERIMEN**

**Waktu Respons**

* **Session Storage:** Rata-rata waktu respons untuk login adalah 120 ms, dan untuk akses endpoint yang dilindungi adalah 110 ms.
* **JWT:** Rata-rata waktu respons untuk login adalah 130 ms, dan untuk akses endpoint yang dilindungi adalah 120 ms.
* **Kesimpulan:** JWT sedikit lebih lambat dalam waktu respons dibandingkan Session Storage, kemungkinan disebabkan oleh proses enkripsi dan dekripsi token.

**Konsumsi Memori**

* **Session Storage:** Rata-rata konsumsi memori adalah 50 MB saat memproses 100 permintaan.
* **JWT:** Rata-rata konsumsi memori adalah 55 MB.
* **Kesimpulan:** JWT memiliki sedikit lebih tinggi konsumsi memori, mungkin karena tambahan overhead untuk menyimpan token JWT di server untuk validasi.

**Ukuran Payload**

* **Session Storage:** Ukuran payload untuk sessionId rata-rata 50 bytes.
* **JWT:** Ukuran payload untuk token JWT rata-rata 200 bytes.
* **Kesimpulan:** JWT memiliki ukuran payload yang lebih besar, yang dapat mempengaruhi bandwidth terutama dalam lingkungan dengan lalu lintas tinggi.

**Throughput**

* **Session Storage:** Server dapat memproses 500 permintaan per detik.
* **JWT:** Server dapat memproses 450 permintaan per detik.
* **Kesimpulan:** Throughput Session Storage sedikit lebih tinggi, mungkin karena tidak adanya kebutuhan untuk melakukan enkripsi/dekripsi pada setiap permintaan.

**Keamanan**

* **Session Storage:** Rentan terhadap serangan XSS dan CSRF, terutama jika sesi tidak ditangani dengan baik.
* **JWT:** Lebih tahan terhadap serangan replay dan XSS, tetapi memerlukan implementasi yang hati-hati untuk menghindari kebocoran token.
* **Kesimpulan:** JWT umumnya lebih aman, tetapi dengan implementasi yang lebih kompleks dibandingkan dengan Session Storage.