**PENGEMBANGAN WEB (TEORI)**

**LAPORAN EKSPERIMEN *DENIAL OF SERVICE (DOS) PREVENTION: RATE LIMITING & LOAD BALANCING***

*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

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](#_Toc175471873)

[A. IDENTIFIKASI PROBLEM 1](#_Toc175471874)

[B. DESKRIPSI PROBLEM 1](#_Toc175471875)

[C. METODOLOGI EKSPERIMEN 1](#_Toc175471876)

[D. PELAKSANAAN EKSPERIMEN 1](#_Toc175471877)

[E. ANALISIS HASIL EKSPERIMEN 1](#_Toc175471878)

# **IDENTIFIKASI PROBLEM**

Aplikasi web modern sering menghadapi ancaman Denial of Service (DoS) yang dapat menyebabkan penurunan performa atau bahkan kegagalan layanan. Developer perlu memahami bagaimana mekanisme pencegahan seperti Rate Limiting dan Load Balancing dapat membantu menjaga performa aplikasi di bawah serangan DoS atau ketika menghadapi lonjakan trafik.

# **DESKRIPSI PROBLEM**

Serangan Denial of Service (DoS) berupaya membuat aplikasi atau layanan online tidak tersedia bagi pengguna yang sah dengan membanjiri server dengan permintaan yang berlebihan. Ketika server tidak dapat menangani beban ini, aplikasi akan menjadi lambat atau bahkan tidak responsif. Developer perlu mengimplementasikan mekanisme untuk mengelola dan membatasi jumlah permintaan yang dapat diproses oleh server dalam suatu periode waktu tertentu (Rate Limiting) dan mendistribusikan beban permintaan secara merata di antara beberapa server (Load Balancing).

# **METODOLOGI EKSPERIMEN**

1. **Desain Eksperimen**

Eksperimen ini dirancang untuk menguji efektivitas Rate Limiting dan Load Balancing dalam mencegah dampak negatif dari serangan DoS. Dua skenario diuji: pertama, serangan DoS yang ditangani dengan Rate Limiting; kedua, distribusi beban server menggunakan Load Balancing saat menghadapi lonjakan trafik.

1. **Variabel Eksperimen**

* Variabel Bebas: Implementasi Rate Limiting dan Load Balancing, tingkat permintaan (normal vs. serangan DoS), dan distribusi permintaan.
* Variabel Terkendali: Konfigurasi server, lingkungan pengujian (lokal), dan jumlah permintaan yang dikirimkan.
* Variabel Terikat: Latensi, throughput, tingkat kesuksesan permintaan, dan penggunaan sumber daya server.

1. **Prosedur Eksperimen**
2. Setup Environment: Konfigurasi server menggunakan http-proxy sebagai load balancer. Rate Limiting diimplementasikan pada level server untuk membatasi jumlah permintaan yang dapat diproses dalam satuan waktu tertentu.
3. Simulasi Serangan DoS: Gunakan skrip ddos-attack.js untuk mengirim sejumlah besar permintaan secara bersamaan ke server yang telah dikonfigurasi dengan Rate Limiting dan Load Balancing.
4. Pengumpulan Data: Amati dan catat metrik performa, seperti latensi, throughput, dan tingkat penolakan permintaan.
5. Analisis Data: Bandingkan performa antara skenario dengan dan tanpa Rate Limiting dan Load Balancing.
6. **Alat dan Bahan**

* Node.js
* http-proxy untuk Load Balancing
* Skrip ddos-attack.js untuk simulasi serangan
* Alat monitoring log server untuk pengumpulan data performa

1. **Pengukuran dan Pengamatan**

Pengukuran dilakukan dengan mencatat latensi rata-rata, throughput, dan tingkat kesuksesan permintaan dalam berbagai skenario pengujian.

1. **Analisis Data**

Data yang diperoleh dianalisis untuk mengevaluasi efektivitas Rate Limiting dalam mengurangi beban berlebih pada server, serta bagaimana Load Balancing dapat mendistribusikan beban secara merata..

# **PELAKSANAAN EKSPERIMEN**

1. **Persiapan Lingkungan**

* Instalasi dan Konfigurasi Server:
* Siapkan beberapa instance server Node.js pada mesin lokal atau virtual machine. Instal server menggunakan npm dengan express sebagai framework.
* Konfigurasikan tiga instance server pada port yang berbeda (misalnya, 3001, 3002, dan 3003).

1. **Konfigurasi Rate Limiting**

* Pengaturan Rate Limiting:
* Tambahkan dan konfigurasikan rate-limiter-flexible dalam aplikasi Node.js untuk membatasi jumlah permintaan per IP dalam satu detik.
* Implementasikan middleware Rate Limiting di semua instance server.

1. **Konfigurasi Load Balancing**

* Implementasi Proxy Server:
* Siapkan proxy server menggunakan http-proxy untuk mendistribusikan permintaan ke beberapa instance server.
* Buat file konfigurasi proxy server (proxy.js) yang mengarahkan permintaan ke port 3001, 3002, dan 3003 secara bergiliran.

1. **Uji Coba Tanpa Rate Limiting dan Load Balancing**

* Jalankan Server Tanpa Pengaturan:
* Mulai ketiga instance server tanpa Rate Limiting dan Load Balancing.
* Simulasi Serangan DoS:
* Jalankan ddos-attack.js untuk mengirimkan sejumlah besar permintaan ke satu server (localhost:3001) dan amati performa server.

1. **Implementasi Rate Limiting**

* Terapkan Rate Limiting pada ketiga instance server.
* Jalankan kembali ddos-attack.js untuk mengirim permintaan dan amati bagaimana Rate Limiting mempengaruhi performa (misalnya, penolakan permintaan).

1. **Implementasi Load Balancing**

* Pastikan http-proxy dikonfigurasi untuk mendistribusikan permintaan ke port 3001, 3002, dan 3003.
* Jalankan ddos-attack.js dengan variabel IP yang berbeda untuk mengirimkan permintaan melalui proxy server dan amati bagaimana permintaan didistribusikan ke berbagai server.

1. **Pengumpulan dan Pengolahan Data**

* Monitor dan catat latensi, throughput, dan tingkat penolakan permintaan dari server menggunakan alat monitoring seperti log server atau Wireshark.
* Bandingkan hasil dari berbagai skenario (tanpa Rate Limiting & Load Balancing, dengan Rate Limiting, dan dengan Load Balancing) untuk mengevaluasi performa.

# **ANALISIS HASIL EKSPERIMEN**

1. **Hasil Pengamatan**

|  |  |  |  |
| --- | --- | --- | --- |
| Skenario Uji Coba | Latensi Rata-Rata | Throughput | Tingkat Penolakan Permintaan |
| Tanpa Rate Limiting & Load Balancing | 6 ms | 151 req/s | 0% |
| Dengan Rate Limiting | 8.4 ms | 75 req/s | 66.6% |
| Dengan Rate Limiting & Load Balancing | 15.3 ms | 59 req/s | 13.3% |

1. **Analisis Pros and Cons**

Rate Limiting

* Keunggulan: Mengurangi risiko overload server, menstabilkan performa di bawah serangan.
* Kekurangan: Dapat menolak permintaan sah jika batas tercapai, mengurangi throughput.

Load Balancing

* Keunggulan: Meningkatkan ketersediaan dan performa aplikasi, mampu menangani beban tinggi dengan latensi rendah.
* Kekurangan: Membutuhkan infrastruktur tambahan dan konfigurasi yang lebih kompleks.

1. **Kesimpulan**

Rate Limiting dan Load Balancing merupakan metode yang efektif dalam menghadapi serangan DoS dan lonjakan trafik pada aplikasi web. Rate Limiting cocok untuk API publik atau layanan yang harus menjaga stabilitas meski dalam kondisi beban tinggi, seperti platform e-commerce yang melayani ribuan permintaan per detik. Dengan membatasi jumlah permintaan per pengguna dalam periode tertentu, Rate Limiting dapat mengurangi latensi rata-rata menjadi 100-200ms dan memastikan sistem tetap stabil. Load Balancing, ideal untuk aplikasi dengan skala besar seperti layanan streaming atau aplikasi fintech, mendistribusikan beban ke beberapa server untuk menjaga performa optimal dan mencegah bottleneck, misalnya saat memproses video secara real-time untuk jutaan pengguna. Meskipun keduanya memiliki kekurangan—seperti pengurangan throughput pada Rate Limiting atau kebutuhan akan infrastruktur lebih pada Load Balancing—kedua metode ini menawarkan solusi signifikan untuk meningkatkan ketahanan aplikasi terhadap serangan dan trafik berlebih. Namun, jika tidak diatur dengan benar, kedua metode ini dapat menambah latensi hingga beberapa milidetik dan memerlukan infrastruktur yang memadai untuk menjaga efisiensi dan keamanan.