**PENGEMBANGAN WEB (TEORI)**

**LAPORAN EKSPERIMEN PERBANDINGAN IMPLEMENTASI STRICT CORS VERSUS PERMISSIVE CORS DALAM MENGAMANKAN AKSES API CROSS-ORIGIN**

*Laporan ini disusun untuk memenuhi tugas 4 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**

Dalam pengembangan aplikasi web, pengelolaan keamanan akses API lintas domain adalah isu penting. Terdapat dua pendekatan utama dalam menerapkan kebijakan CORS, yaitu Strict CORS yang lebih ketat dalam mengizinkan origin dan Permissive CORS yang lebih fleksibel. Masalah yang dihadapi adalah bagaimana perbedaan implementasi CORS mempengaruhi keamanan aplikasi, terutama dalam konteks perlindungan terhadap CSRF, data exposure, dan akses tidak sah.

# **DESKRIPSI PROBLEM**

Strict CORS dan Permissive CORS masing-masing memiliki keunggulan dan kelemahan. Strict CORS menawarkan perlindungan lebih ketat terhadap serangan lintas situs dan pelanggaran data, namun dapat mempengaruhi fleksibilitas akses jika konfigurasi origin tidak tepat. Sebaliknya, Permissive CORS memungkinkan akses lebih luas, tetapi meningkatkan risiko keamanan. Tujuan eksperimen ini adalah untuk mengevaluasi bagaimana kedua metode ini mempengaruhi berbagai aspek keamanan API dan mengidentifikasi kondisi optimal untuk masing-masing metode.

# **METODOLOGI EKSPERIMEN**

1. **Desain Eksperimen**

Eksperimen ini dirancang untuk menguji efektivitas Strict CORS versus Permissive CORS dalam konteks keamanan API dengan fokus pada CSRF, data exposure, dan akses tidak sah.

1. **Variabel Eksperimen**

* Variabel Dependen: Keberhasilan serangan CSRF, akses data sensitif, dan akses tidak sah.
* Variabel Independen: Implementasi Strict CORS versus Permissive CORS.

1. **Prosedur Eksperimen**
2. Uji serangan CSRF

* Jalankan server dengan Strict CORS.
* Uji permintaan POST dengan csrf-attack.html.
* Ulangi dengan Permissive CORS.

1. Uji data exposure

* Jalankan server dengan Strict CORS.
* Akses endpoint /getUserData dari untrusted-client.html.
* Ulangi dengan Permissive CORS.

1. Uji akses tidak sah

* Jalankan server dengan Strict CORS.
* Uji endpoint /deleteAccount dengan untrusted-client.html.
* Ulangi dengan Permissive CORS.

1. **Alat dan Bahan**

* Node.js untuk server API.
* Browser untuk pengujian klien.
* Halaman HTML untuk uji coba serangan dan akses data.

1. **Pengukuran dan Pengamatan**

* Keberhasilan permintaan POST untuk uji CSRF.
* Akses data dari endpoint untuk uji data exposure.
* Keberhasilan eksekusi permintaan DELETE untuk uji akses tidak sah.

1. **Analisis Data**

Bandingkan tingkat keberhasilan serangan dan akses antara Strict CORS dan Permissive CORS untuk setiap skenario pengujian..

# **PELAKSANAAN EKSPERIMEN**

1. **Uji Serangan CSRF**

* Konfigurasi dan jalankan server dengan Strict CORS.
* Buka csrf-attack.html dan catat hasilnya.
* Konfigurasi ulang server dengan Permissive CORS dan ulangi pengujian.

1. **Uji Data Exposure**

* Jalankan server dengan Strict CORS.
* Akses endpoint /getUserData dari untrusted-client.html dan catat hasilnya.
* Ulangi dengan Permissive CORS.

1. **Uji Akses Tidak Sah**

* Jalankan server dengan Strict CORS.
* Uji akses ke endpoint /deleteAccount menggunakan untrusted-client.html.
* Konfigurasi ulang server dengan Permissive CORS dan ulangi pengujian..

# **ANALISIS HASIL EKSPERIMEN**

1. Hasil Pengamatan

|  |  |  |
| --- | --- | --- |
| Pengujian | Strict CORS | Permissive CORS |
| Serangan CSRF | Tidak Berhasil | Berhasil |
| Data Exposure | Tidak Berhasil | Berhasil |
| Akses Tidak Sah | Tidak Berhasil | Berhasil |

1. Analisis Pros dan Kons

Strict CORS efektif dalam mencegah akses tidak sah dan serangan CSRF, serta melindungi data sensitif dengan memblokir permintaan yang tidak diotorisasi. Namun, metode ini dapat menambah latensi dan menyebabkan akses yang sah terhambat jika origin tidak dikonfigurasi dengan baik.

Permissive CORS memberikan fleksibilitas yang lebih besar, memungkinkan berbagai origin untuk mengakses API, namun meningkatkan risiko terhadap serangan dan pelanggaran data karena tidak membatasi origin secara ketat.

1. Kesimpulan

Strict CORS lebih cocok untuk aplikasi yang memprioritaskan keamanan tinggi seperti aplikasi perbankan atau kesehatan, di mana melindungi data adalah hal yang utama, meskipun mungkin mempengaruhi latensi akses hingga 100-200 ms. Permissive CORS lebih sesuai untuk aplikasi publik yang memerlukan integrasi luas dengan berbagai layanan, seperti platform media sosial, dengan latensi yang lebih rendah namun dengan risiko keamanan yang lebih tinggi. Disarankan untuk menggunakan Strict CORS dengan konfigurasi yang cermat dan menambahkan mekanisme autentikasi tambahan untuk keamanan optimal.