**PENGEMBANGAN WEB (TEORI)**

**LAPORAN EKSPERIMEN MENGENAI PROBLEM SERANGAN XSS DENGAN PENYELESAIAN MENGGUNAKAN SOLUSI CSP DI NODE.JS**

*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

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**

Keamanan aplikasi web adalah salah satu aspek terpenting dalam pengembangan perangkat lunak, terutama dalam konteks serangan yang menargetkan celah-celah yang sering tidak disadari oleh pengembang. Salah satu serangan yang sering terjadi adalah Cross-Site Scripting (XSS). XSS memungkinkan penyerang untuk menyuntikkan skrip jahat ke dalam halaman web, yang kemudian dapat dieksekusi oleh pengguna yang mengunjungi halaman tersebut. Serangan ini dapat digunakan untuk mencuri data sensitif, seperti cookie atau informasi login, atau bahkan untuk mengubah tampilan halaman web yang diserang.

Dalam eksperimen ini, kami mengidentifikasi kebutuhan untuk mengamankan aplikasi web dari serangan XSS dengan menggunakan pendekatan yang tidak melibatkan sanitasi input atau escaped output, melainkan dengan menerapkan Content Security Policy (CSP).

# **DESKRIPSI PROBLEM**

Cross-Site Scripting (XSS) adalah jenis serangan injeksi, di mana penyerang dapat mengeksekusi skrip berbahaya di dalam konteks browser pengguna. Serangan ini memanfaatkan celah dalam aplikasi web yang tidak memvalidasi atau meng-escape input dari pengguna dengan benar. Ketika input tersebut ditampilkan kembali kepada pengguna tanpa di-escape, penyerang dapat menyuntikkan skrip HTML atau JavaScript yang berbahaya.

Masalah ini sangat kritis karena pengguna yang terjebak dapat dieksploitasi oleh penyerang untuk melakukan tindakan tanpa sepengetahuan mereka, seperti mengubah konten halaman, mencuri cookie, atau melakukan phishing. Oleh karena itu, kami perlu menerapkan metode yang efektif untuk melindungi aplikasi dari jenis serangan ini.

# **METODOLOGI EKSPERIMEN**

Metodologi eksperimen ini melibatkan dua langkah utama:

1. **Membangun Lingkungan yang Rentan terhadap XSS:**
   * Membuat aplikasi web sederhana menggunakan Node.js dan EJS sebagai template engine.
   * Secara sengaja menonaktifkan fitur escaped output dalam EJS untuk membuat aplikasi rentan terhadap serangan XSS.
2. **Menerapkan Content Security Policy (CSP) sebagai Solusi:**
   * Menggunakan Helmet, sebuah middleware keamanan untuk Express.js, yang memungkinkan pengaturan header HTTP, termasuk Content-Security-Policy.
   * Menerapkan kebijakan CSP yang membatasi sumber daya eksternal yang dapat dimuat oleh halaman dan hanya mengizinkan eksekusi skrip yang memiliki nonce tertentu.

Tujuan dari eksperimen ini adalah untuk menunjukkan bagaimana CSP dapat digunakan untuk mencegah serangan XSS secara efektif.

# **PELAKSANAAN EKSPERIMEN**

1. **Membangun Lingkungan Rentan XSS:**

* Kami memulai dengan membuat sebuah aplikasi web menggunakan Node.js dan Express.js, di mana kami mengatur EJS sebagai template engine.
* Kami kemudian membuat halaman web sederhana yang menampilkan input pengguna secara langsung tanpa escaped output, menggunakan tag <%- %> pada EJS. Hal ini menciptakan situasi di mana input yang disuntikkan oleh pengguna dapat dieksekusi sebagai skrip oleh browser, yang membuat aplikasi menjadi rentan terhadap serangan XSS.
* Kami menguji kerentanan ini dengan memasukkan skrip JavaScript berbahaya ke dalam form input dan melihat bagaimana skrip tersebut dieksekusi oleh browser, yang menandakan keberhasilan serangan XSS.

1. **Menerapkan Solusi CSP:**

* Setelah memastikan bahwa lingkungan kami rentan terhadap XSS, kami kemudian menerapkan CSP menggunakan Helmet.
* Kami menambahkan konfigurasi CSP di server, yang mengatur bahwa hanya skrip dari sumber yang tepercaya dan yang memiliki nonce yang benar yang boleh dieksekusi.
* Kami memperbarui template EJS untuk menyertakan nonce pada skrip yang diizinkan. Setelah itu, kami menjalankan server dan menguji ulang dengan mencoba menyuntikkan skrip berbahaya. Kali ini, CSP mencegah eksekusi skrip yang tidak diizinkan, sehingga mencegah serangan XSS.

# **ANALISIS HASIL EKSPERIMEN**

Hasil dari eksperimen ini menunjukkan bahwa:

1. **Lingkungan Rentan:** Sebelum penerapan CSP, aplikasi web yang kami buat rentan terhadap serangan XSS. Setiap input pengguna yang disuntikkan ke dalam halaman dapat dieksekusi oleh browser, yang menunjukkan bahwa aplikasi tersebut tidak aman.
2. **Penerapan CSP:** Setelah menerapkan CSP, input yang sama tidak lagi dieksekusi oleh browser. Ini karena kebijakan CSP yang kami terapkan hanya mengizinkan eksekusi skrip dari sumber yang tepercaya dan yang memiliki nonce yang sesuai. Sebagai hasilnya, CSP berhasil mencegah serangan XSS dalam aplikasi ini.
3. **Efektivitas CSP:** Penerapan CSP terbukti menjadi solusi yang efektif untuk mencegah XSS, terutama ketika digabungkan dengan nonce pada skrip yang diizinkan. Hal ini memperkuat keamanan aplikasi web tanpa perlu mengubah logika sanitasi input atau mengandalkan escaped output.

Eksperimen ini menunjukkan pentingnya perlindungan terhadap serangan XSS dalam aplikasi web. Meskipun sanitasi input dan escaped output merupakan metode yang umum digunakan, penerapan Content Security Policy (CSP) memberikan lapisan keamanan tambahan yang sangat efektif. Dengan menggunakan Helmet untuk mengatur CSP, kami dapat mencegah eksekusi skrip berbahaya di aplikasi web, sehingga melindungi pengguna dari potensi serangan XSS.

Pengalaman dari eksperimen ini menekankan bahwa keamanan aplikasi web harus menjadi prioritas utama dalam pengembangan perangkat lunak, dan bahwa CSP adalah salah satu alat yang kuat untuk mencapai tujuan tersebut.