PENGEMBANGAN WEB (TEORI)

LAPORAN EKSPERIMEN MENGENAI PROBLEM SERANGAN XSS DENGAN PENYELESAIAN MENGGUNAKAN SOLUSI ESCAPED OUTPUT 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
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

A. IDENTIFIKASI PROBLEM

Seiring dengan perkembangan teknologi web, keamanan aplikasi web menjadi salah satu aspek yang sangat penting untuk diperhatikan. Salah satu ancaman keamanan yang sering terjadi adalah **Cross-Site Scripting (XSS)**. XSS adalah serangan di mana penyerang memasukkan skrip berbahaya ke dalam halaman web yang kemudian dieksekusi di browser pengguna lain. Hal ini dapat mengakibatkan pencurian data, manipulasi konten, dan berbagai ancaman lainnya. Oleh karena itu, penting bagi para pengembang untuk memahami cara mengidentifikasi dan mencegah kerentanan XSS dalam aplikasi web.

B. DESKRIPSI PROBLEM

Cross-Site Scripting (XSS) terjadi ketika input pengguna yang tidak tepercaya dimasukkan ke dalam halaman web tanpa melalui proses validasi atau sanitasi yang memadai. Pada kasus ini, penyerang dapat menyisipkan kode JavaScript berbahaya yang akan dieksekusi oleh browser setiap kali halaman tersebut diakses oleh pengguna lain.

Contoh sederhana dari serangan XSS adalah ketika seorang penyerang memasukkan kode JavaScript ke dalam formulir yang kemudian ditampilkan langsung di halaman web tanpa disaring. Saat pengguna lain mengunjungi halaman tersebut, kode berbahaya ini akan dijalankan dan dapat melakukan berbagai aksi, seperti mencuri cookie atau mengarahkan pengguna ke situs phishing.

C. METODOLOGI EKSPERIMEN

Untuk mempelajari dan mengatasi masalah XSS, kami melakukan eksperimen pada aplikasi web sederhana yang dikembangkan menggunakan **Node.js** dengan framework **Express**. Eksperimen ini dilakukan dalam beberapa tahap:

1. **Membangun Aplikasi Rentan XSS:** Membuat aplikasi web sederhana yang memungkinkan pengguna untuk mengirimkan dan menampilkan pesan. Aplikasi ini dirancang tanpa proteksi terhadap XSS untuk menunjukkan bagaimana serangan dapat terjadi.
2. **Mengidentifikasi Kerentanan:** Menguji aplikasi dengan memasukkan kode JavaScript berbahaya untuk melihat apakah kode tersebut dapat dieksekusi di browser.

3. **Mengimplementasikan Solusi:** Memperbaiki kerentanan dengan melakukan sanitasi dan escaping pada input pengguna, menggunakan metode yang disediakan oleh template engine dan modul keamanan di Node.js.

4. **Menguji Keamanan Aplikasi yang Telah Diperbaiki:** Melakukan pengujian ulang untuk memastikan bahwa input berbahaya tidak lagi dapat dieksekusi.

D. PELAKSANAAN EKSPERIMEN

1. Membangun Aplikasi Rentan XSS

Aplikasi yang dibuat menggunakan Node.js dan Express dilengkapi dengan template engine EJS untuk merender tampilan. Aplikasi ini memiliki formulir sederhana di mana pengguna dapat memasukkan pesan yang kemudian ditampilkan kembali di halaman web.

Berikut adalah contoh kode aplikasi rentan XSS:

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();

app.use(bodyParser.urlencoded({ extended: true }));
app.set('view engine', 'ejs');

app.get('/', (req, res) => {
    res.render('index', { message: '' });
});

app.post('/submit', (req, res) => {
    const message = req.body.message;
    res.render('index', { message: message });
});

app.listen(3000, () => {
    console.log('Server berjalan di http://localhost:3000');
});
```

Pada aplikasi ini, pesan yang dimasukkan pengguna langsung ditampilkan di halaman web tanpa proses escaping. Hal ini membuat aplikasi rentan terhadap serangan XSS.

2. Mengidentifikasi Kerentatanan

Untuk menguji kerentanan, kami mencoba memasukkan kode JavaScript berikut ke dalam formulir:

```
<script>alert('XSS Attack!');</script>
```

Setelah formulir dikirim, kode JavaScript tersebut dieksekusi oleh browser, yang membuktikan bahwa aplikasi ini rentan terhadap XSS.

3. Mengimplementasikan Solusi

Untuk memperbaiki kerentanan, kami melakukan perubahan pada cara rendering pesan. Pada EJS, escaping otomatis dilakukan saat menggunakan '<%= %>'. Namun, jika ingin memaksa aplikasi rentan (untuk eksperimen), kita bisa menggantinya dengan '<%- %>'. Setelah itu, kami menambahkan sanitasi menggunakan express-validator sebagai langkah preventif.

Contoh kode dengan sanitasi:

```
const { body, validationResult } = require('express-validator');

app.post('/submit',
  body('message').escape(),
  (req, res) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
        return res.status(400).json({ errors: errors.array() });
    }
    const message = req.body.message;
    res.render('index', { message: message });
});
```

Dengan perubahan ini, karakter khusus dalam input pengguna di-escape, sehingga kode berbahaya tidak akan dieksekusi.

4. Menguji Keamanan Aplikasi yang Telah Diperbaiki

Kami kembali mencoba memasukkan kode JavaScript berbahaya yang sama setelah melakukan perbaikan. Kali ini, kode tersebut ditampilkan sebagai teks biasa tanpa dieksekusi, yang menunjukkan bahwa aplikasi telah aman dari serangan XSS.

E. ANALISIS HASIL EKSPERIMEN

Dari eksperimen ini, dapat disimpulkan bahwa XSS adalah kerentanan serius yang dapat dieksploitasi oleh penyerang jika input pengguna tidak disanitasi dengan benar. Penggunaan template engine yang mendukung escaping otomatis, seperti EJS, adalah langkah awal yang baik untuk mencegah XSS. Namun, langkah ini harus dilengkapi dengan validasi dan sanitasi input menggunakan modul keamanan seperti 'express-validator'.

Pengembang harus selalu waspada terhadap potensi XSS dalam aplikasi web mereka, terutama ketika menangani input dari pengguna. Dengan mengimplementasikan teknik sanitasi dan validasi yang tepat, kerentanan seperti XSS dapat dicegah, sehingga aplikasi menjadi lebih aman dan andal.