**PENGEMBANGAN WEB (TEORI)**

**LAPORAN EKSPERIMEN MENGENAI PROBLEM SERANGAN XSS DENGAN PENYELESAIAN MENGGUNAKAN SOLUSI SANITASI INPUT DENGAN LIBRARY DOMPURIFY**

*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**

[DAFTAR ISI i](#_Toc175499445)

[A. IDENTIFIKASI PROBLEM 1](#_Toc175499446)

[B. DESKRIPSI PROBLEM 1](#_Toc175499447)

[C. METODOLOGI EKSPERIMEN 1](#_Toc175499448)

[D. PELAKSANAAN EKSPERIMEN 2](#_Toc175499449)

[E. ANALISIS HASIL EKSPERIMEN 3](#_Toc175499450)

[F. KESIMPULAN 4](#_Toc175499451)

# **IDENTIFIKASI PROBLEM**

Serangan XSS adalah jenis kerentanan keamanan web yang memungkinkan penyerang untuk menyuntikkan skrip berbahaya ke dalam halaman web yang kemudian dieksekusi di sisi klien. Masalah ini muncul ketika aplikasi web gagal memvalidasi dan membersihkan input pengguna sebelum menampilkannya kembali ke browser. Salah satu metode serangan XSS yang umum adalah dengan menyisipkan skrip berbahaya melalui parameter URL atau formulir input.

# **DESKRIPSI PROBLEM**

Dalam eksperimen ini, terdapat dua halaman web:

* vulnerable.html: Halaman web yang tidak memiliki mekanisme sanitasi untuk input pengguna, sehingga memungkinkan skrip berbahaya dieksekusi.
* secure.html: Halaman web yang menggunakan library DOMPurify untuk membersihkan input pengguna sebelum menampilkannya, mencegah eksekusi skrip berbahaya.

Masalah ini ditunjukkan dengan membandingkan bagaimana kedua halaman menangani input yang mengandung kode JavaScript berbahaya, khususnya <script>alert('XSS!')</script>, yang ditambahkan melalui parameter URL.

# **METODOLOGI EKSPERIMEN**

Langkah-langkah Eksperimen:

1. Persiapan Halaman Web:

* Buat dua halaman HTML: vulnerable.html dan secure.html.
* vulnerable.html akan menampilkan input langsung dari parameter URL menggunakan document.write().
* secure.html akan menampilkan input dari parameter URL setelah membersihkannya menggunakan DOMPurify.

1. Implementasi Kode:

* Pada vulnerable.html, implementasikan kode JavaScript yang langsung menulis input pengguna ke dalam halaman tanpa sanitasi.
* Pada secure.html, implementasikan DOMPurify untuk membersihkan input sebelum menampilkannya.

1. Pengujian Kerentanan XSS:

* Uji kedua halaman dengan memasukkan input yang mengandung skrip berbahaya melalui parameter URL.
* Observasi apakah skrip berbahaya dijalankan pada halaman yang rentan dan dicegah pada halaman yang menggunakan sanitasi.

# **PELAKSANAAN EKSPERIMEN**

1. Implementasi dan Pengujian vulnerable.html:
2. Kode Implementasi:

|  |
| --- |
|  |

1. Pengujian

* Masukkan <script>alert('XSS!')</script> sebagai nilai parameter URL (misalnya ?name=<script>alert('XSS!')</script>).
* Hasil: Pop-up alert "XSS!" muncul, menunjukkan bahwa skrip berbahaya berhasil dijalankan.

1. Implementasi dan Pengujian secure.html:
2. Kode Implementasi:

|  |
| --- |
|  |

1. Pengujian

* Masukkan <script>alert('XSS!')</script> sebagai nilai parameter URL (misalnya ?name=<script>alert('XSS!')</script>).
* Hasil: Tidak ada pop-up alert yang muncul, dan skrip berbahaya ditampilkan sebagai teks biasa, menunjukkan bahwa DOMPurify berhasil mencegah eksekusi XSS.

# **ANALISIS HASIL EKSPERIMEN**

1. Hasil pada vulnerable.html:

Input berbahaya <script>alert('XSS!')</script> dieksekusi, menunjukkan bahwa halaman ini rentan terhadap serangan XSS karena tidak ada mekanisme sanitasi input.

1. Hasil pada secure.html:

Input yang sama ditampilkan sebagai teks biasa tanpa eksekusi, menunjukkan bahwa DOMPurify efektif dalam membersihkan input pengguna dan mencegah serangan XSS.

1. Kelebihan dan Kekurangan:

Kelebihan menggunakan DOMPurify:

* Melindungi aplikasi web dari serangan XSS dengan efektif.
* Mudah diintegrasikan ke dalam proyek web.
* Terbukti dan diuji oleh komunitas pengembang serta standar industri.

Kekurangan:

* Perlu mengupdate versi DOMPurify untuk memastikan perlindungan terbaru.
* Penggunaan yang tidak tepat atau konfigurasi yang buruk dapat mengurangi efektivitasnya.

# **KESIMPULAN**

Eksperimen ini menunjukkan bahwa tanpa sanitasi input, aplikasi web rentan terhadap serangan XSS. Dengan menggunakan DOMPurify, input yang berpotensi berbahaya dapat dibersihkan sebelum ditampilkan, mencegah eksekusi skrip berbahaya dan meningkatkan keamanan aplikasi web.