**PENGEMBANGAN WEB (TEORI)**

**LAPORAN EKSPERIMEN MENGENAI PROBLEM MENCEGAH SERANGAN CROSS-SITE SCRIPTING (XSS)**

**DENGAN PENYELESAIAN MENGGUNAKAN SOLUSI VALIDASI INPUT PADA WEBSITE NATIVE PHP**

*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](#_Toc175484086)

[**A.** **IDENTIFIKASI PROBLEM** 1](#_Toc175484087)

[**B.** **DESKRIPSI PROBLEM** 1](#_Toc175484088)

[**C.** **METODOLOGI EKSPERIMEN** 1](#_Toc175484089)

[**D.** **PELAKSANAAN EKSPERIMEN** 1](#_Toc175484090)

[**E.** **ANALISIS HASIL EKSPERIMEN** 2](#_Toc175484091)

Link chatGPT : https://chatgpt.com/share/7da9d47d-9a37-4889-8ff8-1193e1ab0131

# **IDENTIFIKASI PROBLEM**

Pada eksperimen ini, kita berfokus pada masalah keamanan web, khususnya terhadap serangan **Cross-Site Scripting (XSS)**. Masalah XSS muncul ketika pengguna dapat memasukkan kode berbahaya (biasanya JavaScript) ke dalam aplikasi web, yang kemudian dieksekusi oleh browser pengguna lain. Serangan ini berpotensi membahayakan privasi, keamanan data, dan bahkan integritas sistem.

# **DESKRIPSI PROBLEM**

Serangan **XSS** memungkinkan penyerang untuk menyisipkan kode berbahaya di halaman web yang dapat dijalankan oleh browser korban. Ini bisa dilakukan melalui kolom input seperti formulir komentar, formulir kontak, atau bidang input lainnya. Contoh sederhana dari serangan XSS adalah memasukkan skrip JavaScript yang berbahaya seperti:

|  |
| --- |
| <script>alert('XSS');</script> |

Jika aplikasi web tidak memvalidasi dan membersihkan input pengguna dengan benar, maka skrip ini dapat dieksekusi di sisi pengguna, menyebabkan potensi pengambilalihan sesi, pencurian cookie, atau bahkan serangan lainnya.

# **METODOLOGI EKSPERIMEN**

Pada eksperimen ini, metodologi yang digunakan untuk mencegah XSS adalah dengan menerapkan **validasi input** pada kolom input yang ada di dalam aplikasi web.

**Langkah-langkah Eksperimen:**

1. **Identifikasi titik lemah aplikasi** di mana input pengguna diterima dan ditampilkan kembali tanpa ada validasi atau sanitasi yang cukup.
2. **Modifikasi kode aplikasi** dengan menerapkan validasi input pada data yang dimasukkan pengguna. Ini termasuk memeriksa apakah input mengandung tag HTML atau skrip yang berbahaya.
3. **Pengujian input berbahaya** untuk memverifikasi apakah validasi yang diterapkan efektif dalam menolak input yang berbahaya.
4. **Analisis hasil** untuk menentukan apakah metode validasi input efektif mencegah serangan XSS.

# **PELAKSANAAN EKSPERIMEN**

Eksperimen dilakukan dengan cara memasukkan kode JavaScript berbahaya ke dalam form komentar di halaman web yang sebelumnya rentan terhadap XSS. Berikut adalah langkah-langkah detailnya:

* **Langkah 1:** Aplikasi web dengan form komentar sederhana disiapkan. Tanpa validasi input, input <script>alert('XSS');</script> diterima dan ditampilkan di halaman web tanpa penyaringan, yang memungkinkan kode tersebut dieksekusi oleh browser.
* **Langkah 2:** Implementasi validasi input dilakukan dengan memeriksa apakah input mengandung tag HTML atau skrip yang berbahaya menggunakan regex preg\_match().
* **Langkah 3:** Setelah validasi input diterapkan, aplikasi diuji kembali dengan input yang sama <script>alert('XSS');</script>. Jika input mengandung karakter atau tag berbahaya, aplikasi menolak input tersebut dan menampilkan pesan kesalahan.
* **Langkah 4:** Input yang valid, seperti teks tanpa elemen HTML atau skrip berbahaya, diterima dan ditampilkan di halaman web tanpa dieksekusi sebagai kode.

# **ANALISIS HASIL EKSPERIMEN**

Dari hasil eksperimen, dapat disimpulkan bahwa penerapan validasi input efektif dalam mencegah serangan XSS. Ketika input berbahaya seperti <script>alert('XSS');</script> dimasukkan, aplikasi dapat mendeteksi dan menolak input tersebut. Sebagai hasilnya, tidak ada skrip yang dieksekusi di browser, sehingga aplikasi tidak rentan terhadap serangan XSS.

**Temuan Penting:**

* **Validasi Input:** Validasi input sangat penting untuk mencegah XSS. Memeriksa apakah input mengandung tag HTML atau skrip berbahaya sebelum disimpan atau ditampilkan dapat mencegah eksekusi kode berbahaya.
* **Fungsi Sanitasi:** Penggunaan fungsi sanitasi seperti htmlspecialchars() tetap diperlukan untuk menambah lapisan keamanan saat menampilkan input pengguna, bahkan setelah validasi dilakukan.
* **Kombinasi Teknik Keamanan:** Validasi input harus digabungkan dengan sanitasi output dan penggunaan header keamanan untuk memberikan perlindungan yang lebih komprehensif terhadap serangan XSS.

**Pro Validasi Input:**

1. **Mencegah Data Berbahaya Masuk:**
   * **Pro:** Memblokir data yang mengandung karakter atau pola berbahaya sebelum diproses atau disimpan, mengurangi risiko eksploitasi seperti XSS dan SQL Injection.
2. **Kontrol Terhadap Data Masuk:**
   * **Pro:** Memberikan kontrol lebih besar atas data yang diterima dari pengguna, memastikan data memenuhi format dan kriteria yang diharapkan.
3. **Peningkatan Keamanan:**
   * **Pro:** Mengurangi risiko masalah keamanan yang disebabkan oleh data yang tidak valid atau berbahaya. Ini adalah langkah awal yang penting untuk perlindungan aplikasi.
4. **Umpan Balik Pengguna:**
   * **Pro:** Memberikan umpan balik langsung kepada pengguna jika data yang dimasukkan tidak valid, meningkatkan pengalaman pengguna dengan mencegah kesalahan.
5. **Mencegah Data yang Tidak Konsisten:**
   * **Pro:** Menghindari data tidak konsisten atau cacat yang dapat mengganggu fungsionalitas aplikasi atau menyebabkan bug.

**Kontra Validasi Input:**

1. **Tidak Selalu Komprehensif:**
   * **Kontra:** Validasi input mungkin tidak dapat menangkap semua variasi atau teknik serangan canggih jika aturan validasi tidak lengkap atau tidak diperbarui.
2. **Kesulitan dalam Mengelola Aturan Validasi:**
   * **Kontra:** Menetapkan dan memelihara aturan validasi yang akurat dan konsisten bisa rumit dan memerlukan perhatian berkelanjutan.
3. **Kemungkinan Menghalangi Input Valid:**
   * **Kontra:** Validasi yang terlalu ketat dapat menolak input yang valid tetapi tidak berbahaya, membatasi fleksibilitas pengguna.
4. **Penanganan Berulang:**
   * **Kontra:** Harus diterapkan di setiap titik di mana data dimasukkan dalam aplikasi, yang dapat menyebabkan duplikasi logika dan potensi kesalahan jika tidak diterapkan secara konsisten.
5. **Tidak Mengganti Sanitasi Output:**
   * **Kontra:** Validasi input tidak menggantikan kebutuhan untuk sanitasi output. Data yang tidak sepenuhnya dikendalikan atau disimpan masih bisa berbahaya jika ditampilkan tanpa sanitasi.

**Kesimpulan:**

**Validasi input** adalah teknik penting dalam strategi keamanan web yang membantu mencegah data berbahaya masuk ke dalam sistem dan memberikan kontrol atas data yang diterima. Namun, meskipun validasi input sangat berguna, ia tidak sepenuhnya menggantikan kebutuhan untuk teknik keamanan tambahan seperti sanitasi output dan Content Security Policy (CSP). Validasi input harus diterapkan dengan hati-hati dan sering digabungkan dengan langkah-langkah keamanan lainnya untuk perlindungan yang lebih komprehensif.

Perbedaan dalam penanganan validasi input antara aplikasi native dan menggunakan framework terletak pada pendekatan, fitur bawaan, pembaruan keamanan, dan konsistensi. Framework menyediakan alat dan pustaka yang mempermudah proses, menawarkan solusi yang lebih konsisten dan terintegrasi, sementara aplikasi native memerlukan pengembangan dan pemeliharaan manual yang lebih intensif. Pilihan antara menggunakan framework atau aplikasi native sering kali tergantung pada kebutuhan proyek, keterampilan tim pengembang, dan preferensi arsitektur.