# KLASIFIKASI SERANGAN CROSS-SITE SCRIPTING (XSS) MENGGUNAKAN MODEL DEEP LEARNING BERBASIS TRANSFORMER

**Naufal Eurasia Negara, Arda Surya Editya**
Informatika, Universitas Nahdlatul Ulama Sidoarjo
Jalan Lingkar Timur KM 5,5 Rangkah Kidul Sidoarjo, Indonesia
*naufaleurasia@gmail.com*

**ABSTRAK**

*Cross-Site Scripting (XSS)* merupakan salah satu jenis serangan siber yang sering terjadi pada aplikasi web dan dapat menyebabkan pencurian data pengguna, manipulasi konten, serta pengambilalihan sesi pengguna. Permasalahan yang muncul adalah metode deteksi tradisional berbasis aturan *(rule-based)* memiliki keterbatasan dalam mengenali variasi *payload* serangan yang terus berkembang dan semakin kompleks. Oleh karena itu, penelitian ini bertujuan untuk mengembangkan model klasifikasi serangan *Cross-Site Scripting* menggunakan pendekatan deep learning berbasis arsitektur *Transformer*. Metode yang digunakan memanfaatkan model *BERT (Bidirectional Encoder Representations from Transformers)* untuk menangkap hubungan kontekstual antar-token dalam *payload*. Dataset yang digunakan terdiri dari *payload benign* dan *malicious* yang melalui tahapan *preprocessing*, tokenisasi, serta pembagian data menjadi data latih dan data uji. Hasil eksperimen menunjukkan bahwa model *Transformer* berbasis *BERT* mampu mengklasifikasikan *payload XSS* secara efektif dengan performa yang lebih baik dibandingkan model *deep learning* konvensional, sehingga berpotensi meningkatkan kemampuan sistem keamanan aplikasi web dalam mendeteksi serangan *XSS* secara otomatis dan adaptif.

***Kata kunci :*** *Cross-Site Scripting, Deep Learning, Transformer, BERT*

## 1. PENDAHULUAN

Perkembangan teknologi internet yang sangat pesat telah menjadikan aplikasi web sebagai salah satu platform utama dalam penyediaan layanan informasi, komunikasi, serta transaksi digital. Website saat ini digunakan secara luas dalam berbagai sektor seperti pendidikan, bisnis, pemerintahan, maupun layanan publik. Kemudahan akses serta fleksibilitas yang ditawarkan oleh aplikasi web menjadikannya sebagai infrastruktur penting dalam sistem informasi modern. Namun demikian, peningkatan penggunaan aplikasi web juga diikuti oleh meningkatnya ancaman keamanan siber yang berpotensi mengganggu integritas, ketersediaan, dan kerahasiaan data pengguna [1].

Salah satu jenis serangan yang paling sering ditemukan pada aplikasi web adalah *Cross-Site Scripting (XSS).* Serangan ini terjadi ketika penyerang menyisipkan skrip berbahaya ke dalam halaman web yang kemudian dijalankan oleh browser pengguna tanpa disadari. Ketika skrip tersebut dieksekusi, penyerang dapat melakukan berbagai aktivitas berbahaya seperti mencuri *cookie* pengguna, memanipulasi konten halaman web, hingga mengambil alih sesi autentikasi pengguna [2].

Kerentanan *XSS* menjadi perhatian serius karena serangan ini relatif mudah dilakukan dan dapat memberikan dampak signifikan terhadap keamanan sistem maupun privasi pengguna [3].

Berbagai pendekatan telah dikembangkan untuk mendeteksi serangan *XSS*, salah satunya adalah metode deteksi berbasis aturan *(rule-based detection).* Pendekatan ini bekerja dengan cara mencocokkan pola tertentu yang telah didefinisikan sebelumnya untuk mengidentifikasi *payload* berbahaya. Meskipun metode ini cukup efektif dalam mendeteksi pola serangan yang telah dikenal, pendekatan berbasis aturan memiliki keterbatasan dalam menghadapi variasi *payload* baru yang terus berkembang [4].

Teknik modifikasi menggunakan *encoding, obfuscation*, maupun manipulasi struktur skrip sering digunakan oleh penyerang untuk menghindari sistem deteksi berbasis pola sehingga metode ini menjadi kurang adaptif terhadap dinamika serangan siber modern [5].

Permasalahan tersebut menunjukkan bahwa diperlukan pendekatan deteksi yang lebih adaptif dan mampu memahami pola serangan yang kompleks.

Seiring dengan perkembangan teknologi kecerdasan buatan, pendekatan berbasis *deep learning* mulai banyak digunakan dalam bidang keamanan siber untuk mengatasi keterbatasan metode tradisional [6].

Salah satu arsitektur yang saat ini banyak digunakan adalah *Transformer* yang memiliki kemampuan untuk memahami hubungan antar-token secara kontekstual melalui mekanisme *self-attention* [7].

Model berbasis *Transformer* seperti *BERT (Bidirectional Encoder Representations from Transformers)* telah menunjukkan kinerja yang sangat baik dalam berbagai tugas pemrosesan bahasa alami, termasuk klasifikasi teks, analisis sentimen, serta deteksi anomali pada data berbasis teks [8].

Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk mengembangkan model klasifikasi serangan *Cross-Site Scripting (XSS)* menggunakan pendekatan deep learning berbasis *Transformer* [9].

Rencana penyelesaian masalah dalam penelitian ini dilakukan dengan memanfaatkan model *BERT* untuk memahami pola *payload XSS* secara kontekstual

melalui tahapan pengumpulan dataset, preprocessing data, tokenisasi, pelatihan model, serta evaluasi performa menggunakan metrik klasifikasi [10].

Dengan pendekatan tersebut, model diharapkan mampu mengidentifikasi *payload* berbahaya secara lebih akurat dibandingkan metode deteksi konvensional sehingga dapat meningkatkan kemampuan sistem keamanan aplikasi web dalam mendeteksi serangan *XSS* secara otomatis dan adaptif terhadap variasi serangan yang terus berkembang [11].

## 2. TINJAUAN PUSTAKA

### 2.1. Penelitian Terkait

Beberapa penelitian sebelumnya telah mengkaji penggunaan metode *machine learning* dan *deep learning* dalam mendeteksi serangan siber pada aplikasi web [12].

Penelitian oleh Alfatah (2024) menunjukkan bahwa model *transformer* mampu memahami hubungan antar kata secara kontekstual sehingga menghasilkan tingkat akurasi yang tinggi dalam tugas klasifikasi teks [1].

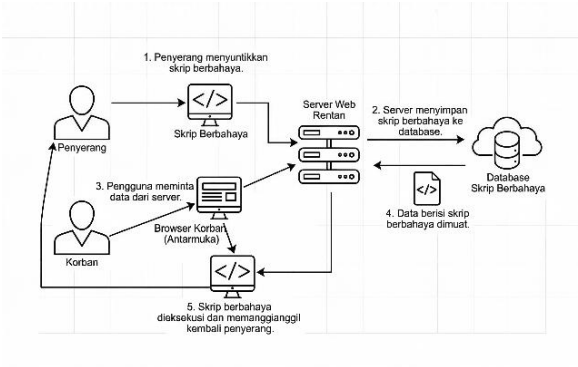
Penelitian lain yang dilakukan oleh Priatna, Sembiring, dan Setiawan (2024) mengembangkan sistem deteksi serangan aplikasi web menggunakan model *transformer* yang dikombinasikan dengan teknik *Natural Language Processing* (NLP). Hasil penelitian menunjukkan adanya peningkatan performa deteksi dibandingkan metode pembelajaran mesin konvensional [11].

Selain itu, penelitian oleh Bakır dan Bakır (2025) menggunakan pendekatan *hybrid semantic embeddings* untuk meningkatkan kemampuan sistem dalam mendeteksi serangan *Cross-Site Scripting* (XSS). Penelitian tersebut menunjukkan bahwa pemodelan konteks semantik mampu meningkatkan akurasi deteksi serangan dibandingkan metode berbasis pola statis [3].

### 2.2. Cross-Site Scripting (XSS)

*Cross-Site Scripting* (XSS) merupakan salah satu jenis serangan injeksi yang memungkinkan penyerang menyisipkan skrip berbahaya ke dalam halaman web yang kemudian dieksekusi oleh browser pengguna. Serangan ini dapat menyebabkan pencurian data sensitif seperti *cookie*, token autentikasi, maupun informasi kredensial pengguna yang tersimpan pada browser [15].

Serangan XSS umumnya terjadi karena kurangnya validasi input pada aplikasi web sehingga memungkinkan pengguna memasukkan kode berbahaya ke dalam sistem. Ketika kode tersebut diproses oleh aplikasi tanpa proses penyaringan yang tepat, skrip tersebut dapat dijalankan pada sisi klien sehingga membuka peluang bagi penyerang untuk melakukan berbagai tindakan berbahaya seperti manipulasi halaman web atau pengambilalihan sesi pengguna [6].




Gambar 1. Alur Serangan XSS

Menurut OWASP, terdapat tiga jenis utama serangan *Cross-Site Scripting* yaitu *Reflected XSS, Stored XSS,* dan *DOM-Based XSS. Reflected XSS* terjadi ketika skrip berbahaya dikirim melalui parameter URL dan langsung dieksekusi oleh browser [3].

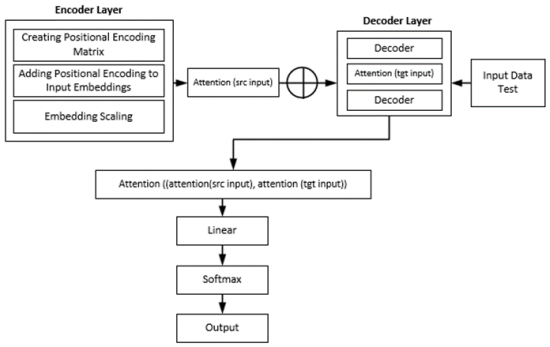
*Stored XSS* terjadi ketika skrip berbahaya disimpan di server, misalnya dalam database, dan dijalankan setiap kali halaman diakses [14].

Sementara itu *DOM-Based XSS* terjadi akibat manipulasi objek *Document Object Model (DOM)* pada sisi klien [13].

### 2.3. Deep Learning dalam Keamanan Web

Deep learning merupakan salah satu cabang dari machine learning yang menggunakan jaringan saraf tiruan berlapis untuk mempelajari pola kompleks dari data dalam jumlah besar. Pendekatan ini banyak digunakan dalam berbagai bidang seperti pengenalan citra, pemrosesan bahasa alami, serta keamanan siber [9].

Dalam konteks keamanan web, deep learning dapat digunakan untuk mendeteksi pola serangan yang sulit dikenali oleh metode tradisional. Model seperti *Convolutional Neural Network* (CNN) dan *Recurrent Neural Network* (RNN) telah banyak digunakan untuk menganalisis pola lalu lintas jaringan maupun *payload* serangan yang bersifat tekstual [10].




Gambar 2. Arsitektur Transformer

Pendekatan *deep learning* memiliki keunggulan dalam mempelajari representasi data secara otomatis tanpa memerlukan proses ekstraksi fitur secara manual. Dengan kemampuan tersebut, sistem

keamanan berbasis deep learning dapat mendeteksi variasi serangan baru yang sebelumnya belum pernah ditemukan pada sistem berbasis aturan [12].

### 2.4. Metode Transformer

*Transformer* merupakan salah satu arsitektur deep learning yang dirancang untuk memproses data berurutan seperti teks dengan cara yang lebih efisien dibandingkan model sekuensial sebelumnya seperti *RNN* dan *LSTM*. Arsitektur ini pertama kali diperkenalkan dalam penelitian yang berjudul *Attention is All You Need* dan menjadi dasar dari berbagai model bahasa modern [13].

Karakteristik utama dari *transformer* adalah penggunaan mekanisme *self-attention* yang memungkinkan model untuk memperhatikan hubungan antar token dalam suatu urutan data secara bersamaan. Dengan mekanisme ini, model dapat memahami konteks global dari sebuah kalimat atau payload tanpa harus memprosesnya secara berurutan seperti pada model *RNN* [7].

Model *Transformer* menggabungkan mekanisme *multi-head attention*. Metode ini secara bersamaan menghitung beberapa keluaran *attention* yang independen dan menggabungkannya untuk menghasilkan hasil akhir. Persamaan untuk menentukan *multi-head attention* disajikan pada Persamaan 1.

$$MultiHead(Q,K,V) = concat(head_1, \dots\dots, head_h)W^o \quad (1)$$

di mana *Q*, *K*, dan *V* masing-masing mewakili matriks kueri, kunci, dan nilai. Variabel *h* menunjukkan jumlah *attention head*, sedangkan $head_i$ adalah $i_{th}$ *attention head*. Selain itu, $W^o$ adalah matriks proyeksi keluaran dan *concat* adalah operasi penggabungan. *Attention* produk titik yang dinormalisasi adalah metode yang digunakan untuk menghitung skor *attention* antara kueri *Q* dan matriks kunci *K*. Persamaan yang digunakan untuk menghitung skor *attention* tersebut ditunjukkan pada Persamaan 2.

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2)$$

dimana $QK^T$ menghitung *dot product* antara vektor kueri *Q* dan vektor kunci $K^T$ yang ditransposisikan, $d_k$ adalah dimensi *K*, dan *softmax* adalah fungsi *softmax*. *Feed Forward Network* berbasis posisi (FFN) terdiri dari jaringan saraf dua lapis yang sederhana yang menggunakan *layer* yang terhubung penuh diikuti oleh ReLU (*Rectified Linear Unit*), yang diterapkan secara independen ke setiap posisi dalam urutan tersebut. Persamaan untuk menghitung keluaran FFN berdasarkan posisi dapat digambarkan pada Persamaan 3.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \quad (3)$$

dimana *FFN* terdiri dari dua *layer* padat yang terhubung penuh, sedangkan *x* adalah vektor masukan, sedangkan $W_1$, $b_1$, $W_2$, dan $b_2$ mewakili bobot dan bias dari dua *fully connected layer*, dan *max* merujuk pada fungsi aktivasi ReLU.

Keunggulan lain dari transformer adalah kemampuannya dalam melakukan pemrosesan paralel sehingga mempercepat proses pelatihan model serta meningkatkan kemampuan model dalam menangkap hubungan jarak jauh antar token. Hal ini menjadikan transformer sangat efektif digunakan dalam berbagai tugas pemrosesan bahasa alami dan analisis keamanan siber [2].

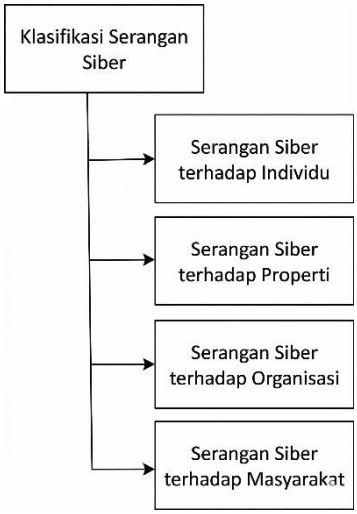
### 2.5. Klasifikasi Serangan Siber

Klasifikasi serangan siber merupakan proses mengidentifikasi dan mengelompokkan aktivitas jaringan atau *payload* berdasarkan karakteristik tertentu apakah termasuk aktivitas normal atau aktivitas berbahaya [12].

Proses ini merupakan bagian penting dalam sistem keamanan seperti *Intrusion Detection System* (IDS) dan *Web Application Firewall* (WAF) [14].

Dalam implementasinya, data yang dikumpulkan dari log jaringan, permintaan HTTP, atau *payload* aplikasi web perlu diubah menjadi representasi numerik agar dapat diproses oleh algoritma pembelajaran mesin. Tahapan ini meliputi proses *feature extraction*, *tokenization*, serta proses pelabelan data untuk membedakan antara *payload benign* (normal) dan *payload malicious* (berbahaya) [10].

Dengan menggunakan metode klasifikasi berbasis *machine learning* atau *deep learning*, sistem keamanan dapat mempelajari pola serangan secara otomatis sehingga mampu mendeteksi variasi serangan baru yang sebelumnya tidak dikenal oleh sistem berbasis aturan tradisional [11].




Gambar 3. Skema Klasifikasi Serangan Siber

## 3. METODE PENELITIAN

### 3.1. Diagram Rancangan Penelitian

Metodologi penelitian ini dirancang untuk menjelaskan tahapan yang dilakukan dalam proses pengembangan sistem klasifikasi serangan *Cross-Site Scripting (XSS)* menggunakan model *deep learning* berbasis *Transformer* [7].

Proses penelitian dilakukan secara sistematis mulai dari tahap studi literatur hingga analisis hasil eksperimen. Setiap tahapan penelitian disusun secara terstruktur untuk memastikan bahwa model yang

dikembangkan mampu mengklasifikasikan *payload XSS* secara efektif dan akurat [11].

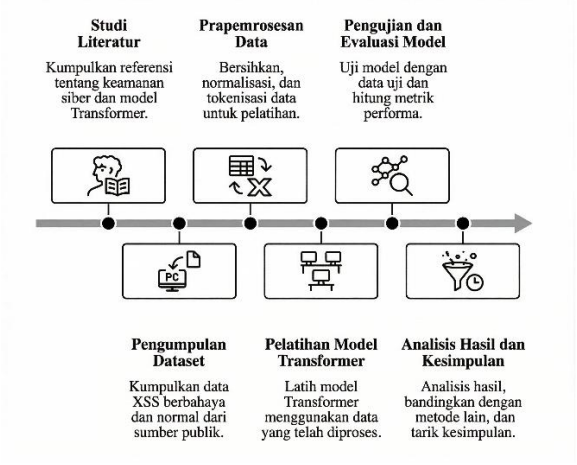
Tahap pertama dalam penelitian ini adalah studi literatur, yang bertujuan untuk memperoleh pemahaman teoritis mengenai topik yang diteliti [1].

Pada tahap ini dilakukan penelaahan terhadap berbagai sumber ilmiah seperti jurnal, buku, dan publikasi penelitian yang berkaitan dengan keamanan siber, serangan *Cross-Site Scripting*, serta penerapan metode deep learning khususnya arsitektur *Transformer* dalam tugas klasifikasi teks [13].

Studi literatur ini memberikan landasan konseptual mengenai karakteristik serangan *XSS*, metode deteksi yang telah digunakan sebelumnya, serta perkembangan teknologi pembelajaran mesin yang dapat dimanfaatkan untuk meningkatkan kemampuan sistem keamanan web [10].

Tahap berikutnya adalah pengumpulan dataset yang akan digunakan dalam proses penelitian. Dataset yang digunakan terdiri dari dua kategori utama, yaitu *payload* berbahaya *(malicious)* yang mengandung skrip *XSS* dan *payload* normal *(benign)* yang tidak mengandung serangan [5].

Data tersebut diperoleh dari berbagai sumber dataset keamanan web yang tersedia secara publik. Dataset yang telah dikumpulkan kemudian disusun dalam format yang terstruktur sehingga dapat digunakan sebagai data latih (*training data*) dan data uji (testing data) dalam proses pengembangan model klasifikasi [12].




Gambar 4. Diagram Rancangan Penelitian

Setelah dataset diperoleh, dilakukan tahap prapemrosesan data (*data preprocessing*) sebelum data digunakan dalam proses pelatihan model [6].

Tahap ini bertujuan untuk membersihkan dan menyiapkan data agar sesuai dengan kebutuhan model pembelajaran mesin [9].

Proses prapemrosesan meliputi pembersihan data *(data cleaning)* untuk menghilangkan karakter yang tidak relevan atau duplikasi data, normalisasi teks *payload* agar memiliki format yang seragam, serta proses tokenisasi untuk memecah *payload* menjadi unit-unit token yang dapat diproses oleh model *Transformer* [7].

Selain itu, dataset juga dibagi menjadi data latih dan data uji untuk mendukung proses pelatihan dan evaluasi model.

Tahap selanjutnya adalah pelatihan model *Transformer*. Pada tahap ini model dilatih menggunakan dataset latih yang telah diproses sebelumnya [8].

Model *Transformer* memanfaatkan mekanisme *self-attention* untuk memahami hubungan antar-token dalam *payload* secara kontekstual [7].

Melalui proses pelatihan ini, model diharapkan mampu mempelajari pola-pola yang membedakan payload normal dengan payload yang mengandung serangan *XSS* [3].

Proses pelatihan dilakukan secara iteratif sehingga parameter model dapat disesuaikan secara optimal untuk meningkatkan kemampuan klasifikasi [9].

Setelah proses pelatihan selesai, tahap berikutnya adalah pengujian dan evaluasi model. Pengujian dilakukan menggunakan data uji yang tidak digunakan selama proses pelatihan [8].

Tujuan dari tahap ini adalah untuk mengetahui kemampuan model dalam mengklasifikasikan *payload* baru yang belum pernah dilihat sebelumnya [11].

Evaluasi performa model dilakukan menggunakan beberapa metrik evaluasi yang umum digunakan dalam penelitian klasifikasi, yaitu *accuracy, precision, recall,* dan *F1-score* [10]. Penggunaan metrik evaluasi tersebut bertujuan untuk memberikan gambaran yang lebih komprehensif mengenai kinerja model dalam mendeteksi *payload* yang mengandung serangan *XSS* [7].

Tahap terakhir dalam penelitian ini adalah analisis hasil dan penarikan kesimpulan. Pada tahap ini dilakukan interpretasi terhadap hasil pengujian yang diperoleh dari metrik evaluasi yang digunakan. Hasil analisis kemudian digunakan untuk menilai efektivitas model *Transformer* dalam mengklasifikasikan serangan *Cross-Site Scripting* [13].

Berdasarkan hasil tersebut, peneliti menarik kesimpulan mengenai kinerja model yang dikembangkan serta potensi penerapannya dalam meningkatkan sistem keamanan aplikasi web terhadap serangan XSS [14].

### 3.2. Pengumpulan Dataset

Tahap awal dalam penelitian ini adalah melakukan pengumpulan dataset yang akan digunakan dalam proses pelatihan dan pengujian model klasifikasi serangan *Cross-Site Scripting (XSS)* [8].

Dataset yang digunakan terdiri dari dua kategori utama, yaitu *payload* berbahaya *(malicious)* yang mengandung skrip XSS dan *payload* normal *(benign)* yang tidak mengandung unsur serangan [5].

Dataset ini digunakan sebagai dasar bagi model untuk mempelajari pola-pola yang membedakan antara input normal dan *payload* yang mengandung skrip berbahaya.

Data yang digunakan dalam penelitian ini diperoleh dari beberapa sumber dataset publik yang berkaitan dengan keamanan web, di antaranya *OWASP XSS Payload Dataset* dan *Kaggle XSS Payload Collection*. Dataset tersebut berisi berbagai variasi contoh payload serangan XSS yang umum digunakan oleh penyerang serta contoh input normal yang sering ditemukan pada aplikasi web [15].

Penggunaan dataset dari berbagai sumber bertujuan untuk memberikan variasi pola payload sehingga model dapat mempelajari karakteristik serangan XSS secara lebih komprehensif [7].

Secara keseluruhan, jumlah dataset yang digunakan dalam penelitian ini adalah 3000 data, yang terdiri dari 1500 *payload* XSS *(malicious)* dan 1500 *payload* normal *(benign)*. Data tersebut kemudian diberi label untuk memudahkan proses klasifikasi pada tahap pelatihan model [10].

*Payload* yang mengandung serangan XSS diberi label 1, sedangkan *payload* normal yang tidak mengandung serangan diberi label 0. Contoh struktur dataset yang digunakan dalam penelitian ini dapat dilihat pada Tabel 1.

Tabel 1. Pengumpulan Dataset

| No | Payload | Label |
|---|---|---|
| 1 | <script>alert('xss')</script> | 1 |
| 2 | Hello World! | 0 |
| 3 | <img src=x onerror=alert(1)> | 1 |

Dataset yang telah dikumpulkan kemudian digunakan pada tahap selanjutnya yaitu prapemrosesan data *(data preprocessing)* sebelum diproses lebih lanjut dalam pelatihan model *Transformer* [13].

### 3.3. Preprocessing dan Pelatihan Model

Setelah proses pengumpulan dataset selesai dilakukan, tahap berikutnya adalah prapemrosesan data (*data preprocessing*) sebelum dataset digunakan dalam proses pelatihan model. Tahap prapemrosesan bertujuan untuk membersihkan serta menyiapkan data agar dapat diproses secara optimal oleh model deep learning. Kualitas data pada tahap ini sangat berpengaruh terhadap performa model dalam proses pembelajaran [2].

Proses prapemrosesan yang dilakukan dalam penelitian ini meliputi beberapa tahapan utama, yaitu pembersihan data (*data cleaning*), normalisasi teks, tokenisasi, serta encoding data menggunakan tokenizer dari model *BERT (Bidirectional Encoder Representations from Transformers)* [13].

Pada tahap data cleaning, dilakukan proses penghapusan karakter yang tidak diperlukan serta penanganan format payload yang tidak konsisten. Selanjutnya dilakukan normalisasi teks untuk menyeragamkan struktur input sehingga lebih mudah diproses oleh model [9].

Tahap berikutnya adalah tokenisasi *(tokenization)*, yaitu proses memecah *payload* menjadi unit-unit token yang lebih kecil [7].

Tokenisasi dilakukan menggunakan *tokenizer BERT* yang telah dilatih sebelumnya. Token-token yang dihasilkan kemudian dikonversi menjadi representasi numerik berupa input IDs agar dapat diproses oleh model *Transformer*. Selain itu, dibuat juga *attention mask* yang berfungsi untuk menunjukkan token mana yang merupakan bagian dari input yang valid dan mana yang merupakan *padding* [7].

Pada penelitian ini panjang maksimum token ditetapkan sebanyak 64 token untuk setiap *payload* [8].

Contoh hasil proses prapemrosesan dataset dapat dilihat pada Tabel 2, yang menunjukkan representasi numerik dari payload setelah melalui proses tokenisasi dan encoding menggunakan *tokenizer BERT*.

Tabel 2. Data Hasil Preprocessing

| No | Payload | Label | Tokenized Input IDs (64) | Attention Mask |
|---|---|---|---|---|
| 1. | <script>alert('xss')</script> | 1 | [101, 188, 2556, 15456, ... , 0, 0, 0] | [1, 1, 1, 1, ..., 0, 0, 0] |
| 2. | Hello World! | 0 | [101, 7592, 2088, 999, ..., 0, 0, 0] | [1, 1, 1, ..., 0, 0, 0] |
| 3. | <img src=x onerror=alert(1)> | 1 | [101, 188, 1969, 8324, ... , 0, 0, 0] | [1, 1, 1, ..., 0, 0, 0] |

Setelah proses prapemrosesan selesai, dataset kemudian dibagi menjadi tiga bagian utama untuk mendukung proses pelatihan dan evaluasi model. Dataset dibagi menjadi data latih *(training data)* sebesar 70%, data uji *(testing data)* sebesar 20%, dan data validasi *(validation data)* sebesar 10% [10].

Data latih digunakan untuk melatih model agar dapat mempelajari pola *payload* serangan, data validasi digunakan untuk memantau proses pelatihan dan mencegah terjadinya *overfitting*, sedangkan data uji digunakan untuk mengevaluasi performa model pada data yang belum pernah dilihat sebelumnya [11].

Tahap selanjutnya adalah pelatihan model *(model training)* menggunakan arsitektur *Transformer* dengan memanfaatkan model *BERT (Bidirectional Encoder Representations from Transformers)* [13].

Model ini memiliki kemampuan untuk memahami hubungan kontekstual antar-token dalam teks melalui mekanisme *self-attention*. Dengan kemampuan tersebut, model dapat mempelajari pola yang kompleks pada payload *XSS* sehingga mampu membedakan antara *payload* berbahaya dan *payload* normal secara lebih akurat. Proses pelatihan dilakukan dengan menyesuaikan parameter model secara iteratif hingga diperoleh performa klasifikasi yang optimal [9].

### 3.4. Pengujian dan Evaluasi Model

Setelah proses pelatihan model selesai dilakukan, tahap berikutnya adalah pengujian model menggunakan data uji *(testing data).* Data uji merupakan bagian dari dataset yang tidak digunakan selama proses pelatihan, sehingga dapat digunakan untuk mengukur kemampuan model dalam melakukan generalisasi terhadap data baru yang belum pernah dipelajari sebelumnya [8].

Proses pengujian ini bertujuan untuk mengetahui sejauh mana model mampu mengklasifikasikan *payload* yang mengandung serangan *Cross-Site Scripting (XSS)* dan membedakannya dari *payload* normal *(benign)*[5].

Evaluasi performa model dilakukan dengan menggunakan beberapa metrik evaluasi yang umum digunakan dalam penelitian klasifikasi, yaitu *Accuracy*, *Precision, Recall,* dan *F1-score* [10].

Metrik-metrik tersebut dihitung berdasarkan hasil klasifikasi yang diperoleh dari model terhadap data uji. Penggunaan beberapa metrik evaluasi bertujuan untuk memberikan gambaran yang lebih komprehensif mengenai kinerja model dalam mendeteksi serangan XSS [7].

*Accuracy* digunakan untuk mengukur tingkat ketepatan model dalam mengklasifikasikan seluruh data secara keseluruhan. *Precision* digunakan untuk mengukur tingkat ketepatan model dalam mengidentifikasi *payload* yang benar-benar mengandung serangan XSS dari seluruh prediksi positif yang dihasilkan oleh model. *Recall* digunakan untuk mengukur kemampuan model dalam mendeteksi seluruh payload serangan yang terdapat dalam dataset. Sementara itu, *F1-score* merupakan nilai harmonisasi antara *precision* dan *recall* yang digunakan untuk memberikan gambaran keseimbangan antara kedua metrik tersebut [10].

Perhitungan metrik evaluasi tersebut dapat dirumuskan pada Rumus 4-6.

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

$$Recall = \frac{TP}{TP + FN} \quad (5)$$

$$F1\ score = \frac{2 \times precision \times recall}{precision + recall} \quad (6)$$

dimana *TP (True Positive)* adalah jumlah payload serangan yang berhasil diklasifikasikan dengan benar sebagai serangan. *TN (True Negative)* adalah jumlah payload normal yang berhasil diklasifikasikan dengan benar sebagai data normal. *FP (False Positive)* adalah payload normal yang salah diklasifikasikan sebagai serangan. *FN (False Negative)* adalah payload serangan yang salah diklasifikasikan sebagai data normal.

Hasil evaluasi yang diperoleh dari metrik-metrik tersebut kemudian dianalisis untuk menilai efektivitas model *Transformer* berbasis *BERT* dalam melakukan klasifikasi serangan *Cross-Site Scripting (XSS).* Analisis ini digunakan untuk mengetahui tingkat keberhasilan model dalam mengidentifikasi payload berbahaya serta menilai potensi penerapan metode *Transformer* dalam meningkatkan sistem keamanan aplikasi web [14].

## 4. HASIL DAN PEMBAHASAN

### 4.1. Deskripsi Dataset

Dataset yang digunakan dalam penelitian ini merupakan kumpulan payload *Cross-Site Scripting (XSS)* yang terdiri dari dua kategori utama, yaitu *payload benign* (normal) dan *payload malicious* (berbahaya). Dataset ini digunakan sebagai data masukan untuk melatih dan menguji performa model *deep learning* dalam melakukan klasifikasi serangan *XSS* pada aplikasi web. Melalui dataset tersebut, model diharapkan mampu mempelajari karakteristik *payload* yang mengandung skrip berbahaya serta membedakannya dari input normal.

Dataset diperoleh dari beberapa sumber publik yang menyediakan kumpulan payload XSS, di antaranya *OWASP XSS Payload Dataset* dan *Kaggle XSS Payload Collection.* Dataset tersebut berisi berbagai variasi *payload* serangan yang umum digunakan oleh penyerang, seperti penyisipan skrip *JavaScript,* manipulasi atribut *HTML*, serta teknik eksploitasi lain yang sering ditemukan dalam serangan *XSS.* Selain itu, dataset juga dilengkapi dengan contoh input normal yang tidak mengandung skrip berbahaya sehingga dapat digunakan sebagai pembanding dalam proses klasifikasi.

Sebelum digunakan dalam proses pelatihan model, dataset terlebih dahulu melalui tahap prapemrosesan data *(data preprocessing).* Tahapan ini meliputi proses pembersihan data *(data cleaning),* normalisasi teks, *tokenisasi payload,* serta proses encoding menggunakan *tokenizer* dari model *BERT.* Proses prapemrosesan dilakukan untuk memastikan bahwa data yang digunakan memiliki format yang konsisten dan dapat diproses secara optimal oleh model pembelajaran mesin.

Pada penelitian ini jumlah dataset yang digunakan dibatasi sebanyak 2000 payload untuk menjaga keseimbangan kelas serta meningkatkan efisiensi proses pelatihan model. Dataset tersebut terdiri dari 1000 *payload benign* (normal) dan 1000 *payload malicious (XSS)* sehingga distribusi data menjadi seimbang antara kedua kelas. Keseimbangan dataset ini penting untuk menghindari bias model terhadap salah satu kelas selama proses pelatihan.

Selanjutnya dataset dibagi menjadi dua bagian utama, yaitu data latih *(training data)* dan data uji *(testing data)* dengan rasio 80% : 20%. Sebanyak 1600 data digunakan sebagai data latih untuk melatih model dalam mempelajari pola *payload*, sedangkan 400 data digunakan sebagai data uji untuk mengevaluasi kemampuan model dalam mengklasifikasikan *payload* yang belum pernah dipelajari sebelumnya. Pembagian dataset ini bertujuan untuk mengukur kemampuan generalisasi model secara lebih objektif.

Dalam penelitian ini, dataset yang telah diproses digunakan untuk melatih tiga model deep learning,

yaitu *Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN),* dan *Transformer* berbasis *BERT*. Penggunaan beberapa model tersebut bertujuan untuk melakukan perbandingan performa dalam proses klasifikasi serangan *XSS*, sehingga dapat diketahui model mana yang memiliki kinerja terbaik dalam mengidentifikasi payload berbahaya pada aplikasi web.

### 4.2. Hasil Pengujian Model LSTM

Pada penelitian ini dilakukan pengujian terhadap tiga model *deep learning* yaitu *Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN),* dan *Transformer* berbasis *BERT* untuk melakukan klasifikasi *payload Cross-Site Scripting (XSS)* dan *payload benign* (normal). Pengujian dilakukan menggunakan data uji yang telah dipisahkan dari data latih untuk mengukur kemampuan generalisasi masing-masing model dalam mengklasifikasikan *payload* yang belum pernah dipelajari sebelumnya.

Evaluasi performa model dilakukan menggunakan beberapa metrik yang umum digunakan dalam penelitian klasifikasi, yaitu *accuracy*, *precision, recall,* dan *F1-score.* Metrik tersebut digunakan untuk mengukur kemampuan model dalam mengidentifikasi payload berbahaya serta membedakannya dari *payload* normal secara akurat.

Model *LSTM* digunakan sebagai model pembanding karena memiliki kemampuan dalam memproses data sekuensial serta memahami hubungan antar elemen dalam suatu urutan teks. Berdasarkan hasil pengujian, model *LSTM* mampu mendeteksi sebagian besar payload *XSS* dengan baik, namun masih mengalami kesalahan klasifikasi pada beberapa *payload benign* yang dianggap sebagai serangan. Hal ini menyebabkan nilai *precision* dan *accuracy* model *LSTM* tidak terlalu tinggi karena adanya *false positive.*

Model *CNN* menunjukkan performa yang lebih baik dibandingkan *LSTM*. Hal ini disebabkan oleh kemampuan *CNN* dalam menangkap pola lokal pada data teks, seperti kemunculan karakter khusus (<, >, ', ") serta struktur tag *HTML* dan skrip *JavaScript* yang sering digunakan dalam *payload XSS*. Dengan kemampuan tersebut, *CNN* dapat mengenali pola serangan dengan lebih baik sehingga menghasilkan nilai evaluasi yang lebih tinggi.

Sementara itu, model *Transformer* berbasis *BERT* menunjukkan performa terbaik dalam penelitian ini. Keunggulan *Transformer* terletak pada mekanisme *self-attention* yang memungkinkan model memahami hubungan antar token secara global dalam satu *payload*. Dengan kemampuan ini, model tidak hanya bergantung pada pola lokal tetapi juga mampu memahami konteks keseluruhan *payload* sehingga dapat mengenali variasi serangan *XSS* yang lebih kompleks.

Perbandingan hasil evaluasi dari ketiga model yang digunakan dalam penelitian ini dapat dilihat pada Tabel 3.

Tabel 3. Perbandingan Performa Model Klasifikasi XSS

| Model | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| LSTM | 0.87 | 0.86 | 0.89 | 0.87 |
| CNN | 0.93 | 0.92 | 0.94 | 0.93 |
| Transformer (BERT) | **0.96** | **0.95** | **0.97** | **0.96** |

Berdasarkan hasil evaluasi tersebut dapat diketahui bahwa model *Transformer* berbasis *BERT* memiliki performa terbaik dibandingkan model *LSTM* dan *CNN*. Model ini mampu mengklasifikasikan *payload XSS* dengan tingkat akurasi yang lebih tinggi serta memiliki keseimbangan nilai *precision* dan *recall* yang lebih baik. Hal ini menunjukkan bahwa pendekatan berbasis *Transformer* lebih efektif dalam memahami pola dan konteks *payload XSS* dibandingkan model deep learning lainnya.

Hasil penelitian ini menunjukkan bahwa penggunaan arsitektur *Transformer* memiliki potensi yang signifikan dalam meningkatkan kemampuan sistem keamanan aplikasi web untuk mendeteksi dan mengklasifikasikan serangan *Cross-Site Scripting (XSS)* secara otomatis dan lebih akurat.

## 5. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian yang telah dilakukan mengenai klasifikasi serangan *Cross-Site Scripting (XSS)* menggunakan pendekatan deep learning berbasis *Transformer*, dapat disimpulkan bahwa metode yang diusulkan mampu mengklasifikasikan payload berbahaya dan payload normal pada aplikasi web secara efektif. Proses penelitian dimulai dari tahap pengumpulan dataset, prapemrosesan data, pelatihan model, hingga evaluasi performa menggunakan metrik *accuracy*, *precision, recall,* dan *F1-score.* Tahapan tersebut memungkinkan model untuk mempelajari karakteristik *payload XSS* serta membedakannya dari payload normal secara lebih akurat.

Hasil eksperimen menunjukkan bahwa model *Transformer* berbasis *BERT* memiliki performa terbaik dibandingkan dengan model *LSTM* dan *CNN*. Model *LSTM* menunjukkan performa yang relatif lebih rendah dengan akurasi sebesar 0,87, sedangkan model *CNN* memperoleh akurasi sebesar 0,93. Sementara itu, model *Transformer* menghasilkan performa tertinggi dengan nilai *accuracy* sebesar 0,96, serta nilai *precision, recall,* dan *F1-score* yang lebih baik dibandingkan dua model lainnya. Keunggulan model *Transformer* dipengaruhi oleh mekanisme *self-attention* yang memungkinkan model memahami hubungan antar-token secara kontekstual sehingga mampu mengenali pola serangan *XSS* yang lebih kompleks dan bervariasi.

Meskipun penelitian ini menunjukkan hasil yang cukup baik, masih terdapat beberapa keterbatasan yang perlu diperhatikan. Dataset yang digunakan dalam penelitian ini relatif terbatas sehingga belum sepenuhnya merepresentasikan variasi *payload XSS*

yang sangat beragam di dunia nyata. Oleh karena itu, penelitian selanjutnya disarankan untuk menggunakan dataset yang lebih besar dan lebih bervariasi agar kemampuan generalisasi model dapat ditingkatkan.

Selain itu, pengembangan lebih lanjut dapat dilakukan dengan mengintegrasikan model *Transformer* ke dalam sistem keamanan aplikasi web secara *real-time*, seperti *Web Application Firewall (WAF)* atau *Intrusion Detection System (IDS).* Integrasi tersebut diharapkan dapat meningkatkan kemampuan sistem dalam mendeteksi serangan *XSS* secara otomatis dan memberikan perlindungan yang lebih efektif terhadap aplikasi web.

## DAFTAR PUSTAKA


[1] Alfatah, D. (2024). Penerapan Model Transformer Untuk Deteksi Sentimen Pada Data Twitter Berbahasa Indonesia. Jurnal Komputer, 2(2), 67-70.

[2] Alshomrani, M., Albeshri, A., Alturki, B., Alallah, F. S., & Alsulami, A. A. (2024). Survey of transformer-based malicious software detection systems. Electronics, 13(23), 4677. https://doi.org/10.3390/electronics13234677

[3] Bakır, R., & Bakır, H. (2025). Swift detection of XSS attacks: Enhancing XSS attack detection by leveraging hybrid semantic embeddings and AI techniques. *Arabian Journal for Science and Engineering*, 50(2), 1191–1207.

[4] Duddin, H. A. A., & Fitrani, A. S. (2021). Securing input and output processes on the web to minimize SQL-injection and XSS attacks using IDS and IPS methods. *JOINCS (Journal of Informatics, Network, and Computer Science)*, 4(1).

[5] Hartono, H., Khotimah, K., & Wibowo, A. (2023). Deteksi serangan remote code execution dan cross-site scripting menggunakan machine learning. Jurnal Informatika, 23(2), 229–242.

[6] Kasdana, H., Makmur, Z., Efendi, G., & Achmad, M. M. (2025). Analisis kerentanan aplikasi web terhadap SQL injection dan XSS berdasarkan survei pengguna dengan kategori risiko. *Aisyah Journal of Informatics and Electrical Engineering (AJIEE)*, 7(2), 68–75.

[7] Liu, Y., Wang, L., & Dai, Y. (2025). XSS attack detection with deep learning and AdaBoost. *IEEE Access*, 100, 1–10.

[8] Luu, G. H., Duong, M. K., Pham-Ngo, T. P., Ngo, T. S., Nguyen, D. T., Nguyen, X. H., & Le, K. H. (2024). XSShield: A novel dataset and lightweight hybrid deep learning model for XSS attack detection. *Results in Engineering*, 24, 103363.

[9] Muttaqin, R. Z., & Sudiana, D. (2024). Design of realtime web application firewall based on deep learning to improve web application security. *Jurnal Penelitian Pendidikan IPA*, 10(12), 11121–11129.

[10] Prasetio, D. A., Kusrini, K., & Arief, M. R. (2021). Cross-site scripting attack detection using machine learning with hybrid features. *Jurnal Infotel*, 13(1), 1–6.

[11] Priatna, W., Sembiring, I., Setiawan, A., & Setyawan, I. (2024). Network intrusion detection using transformer models and natural language processing for enhanced web application attack detection. *Jurnal Nasional Pendidikan Teknik Informatika (JANAPATI)*, 13(3), 482–493.

[12] Purwidyantoro, A. R., & Pramukantoro, E. S. (2025). Sistem klasifikasi serangan pada website berbasis WordPress menggunakan machine learning. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 9(3).

[13] Stein, K., Mahyari, A., Francia, G., & El-Sheikh, E. (2024). A transformer-based framework for payload malware detection and classification. In *Proceedings of the 2024 IEEE World AI IoT Congress (AIIoT)* (pp. 105–111). IEEE.

[14] Vellela, S. S. (2025). Detecting SQL injection, cross-site scripting, and cross-site request forgery. *International Journal of Information Security*, 24(1), 1–12.

[15] Wibowo, R. M., & Sulaksono, A. (2021). Web vulnerability through cross-site scripting (XSS) detection with OWASP Security Shepherd. *Indonesian Journal of Information Systems*, 3(2), 149–159.