Jurnal Teknologi dan Sains Modern
E-ISSN: 3062-7966
DOI: https://doi.org/10.69930/jtsm.v3i4.840
Research Article

ScienceTech. Group

Vol. 3 (4), 2025
Hal. 198-213

Deteksi Serangan Cross-Site Scripting (XSS) menggunakan Pendekatan Hybrid TF-IDF dan Bert Embeddings

Ahmad Anas *, Erna Dwi Astuti, Nahar Mardiyantoro

Program Studi Teknik Informatika, Fakultas Teknik dan Ilmu Komputer, Universitas Sains Al Qur'an, Wonosobo, Indonesia
*Email (Penulis Korespondensi): ctowet@mhs.unsiq.ac.id

Abstrak. *Serangan Cross-Site Scripting (XSS) secara konsisten mendominasi daftar kerentanan aplikasi web; lebih dari 8.000 entri Common Vulnerabilities and Exposures (CVE) aktif terkait CWE-79 tercatat sepanjang 2025. Sistem deteksi berbasis pola terbukti tidak memadai menghadapi payload yang menerapkan teknik obfuscation berlapis. Penelitian ini mengusulkan pendekatan hybrid yang menggabungkan fitur statistik TF-IDF berbasis karakter n-gram (1,3) berdimensi 5.000 dengan representasi semantik BERT embeddings (token [CLS]) berdimensi 768 melalui operasi feature concatenation, menghasilkan vektor fitur berdimensi 5.768 sebagai masukan classifier Logistic Regression. Eksperimen dilakukan pada dataset XSS_dataset.csv berisi 13.686 sampel berlabel (7.373 XSS, 6.313 Benign) dengan pembagian stratified 80:20. Evaluasi pada testing set (n=2.738) menghasilkan accuracy 99,78%, precision 99,80%, recall 99,80%, F1-score 99,80%, dan ROC-AUC 1,0000. Pendekatan hybrid secara konsisten melampaui konfigurasi TF-IDF tunggal (F1=99,66%) maupun BERT tunggal (F1=99,73%). Analisis confusion matrix mengidentifikasi hanya enam kesalahan klasifikasi, tiga false negative di antaranya berasal dari payload yang disembunyikan dalam base64-sebuah temuan yang mengonfirmasi batas kemampuan pipeline preprocessing berbasis URL decoding dan HTML entity decoding. Model diimplementasikan sebagai prototipe XSSentinel melalui Flask REST API dan antarmuka React. Penelitian ini membuktikan bahwa penggabungan representasi statistik dan semantik menghasilkan sistem deteksi XSS yang akurat dan efisien tanpa bergantung pada arsitektur deep learning yang berat. Penelitian ini relevan dengan SDG 9 (infrastruktur digital yang andal), SDG 16 (perlindungan data dan keamanan siber), dan SDG 8 (produktivitas ekonomi digital).*

Kata kunci: *Deteksi XSS; Machine Learning; Hybrid Features; TF-IDF; BERT Embeddings*

Abstract. *Cross-Site Scripting (XSS) attacks consistently rank among the most prevalent web application vulnerabilities, with over 8,000 active Common Vulnerabilities and Exposures (CVE) entries associated with CWE-79 recorded throughout 2025. Conventional pattern-based detection systems have proven inadequate against payloads employing multi-layered obfuscation techniques. This study proposes a hybrid approach combining character-level TF-IDF features (n-gram range 1–3, 5,000 dimensions) with BERT semantic embeddings (CLS token, 768 dimensions) through feature concatenation, producing a 5,768-dimensional feature vector for a Logistic Regression classifier. Experiments were conducted on the XSS_dataset.csv dataset containing 13,686 labeled samples (7,373 XSS, 6,313 Benign) with an 80:20 stratified split. Evaluation on the testing set (n=2,738) achieved accuracy of 99.78%, precision of 99.80%, recall of 99.80%, F1-score of 99.80%, and ROC-AUC of 1.0000. The hybrid approach consistently outperformed single-feature configurations of TF-IDF alone (F1=99.66%) and BERT alone (F1=99.73%). Confusion matrix analysis identified only six misclassifications; three false negatives originated from base64-encoded payloads, confirming the processing boundary of the URL decoding and HTML entity decoding pipeline. The model was implemented as the XSSentinel prototype through a Flask REST API backend and React frontend. This study demonstrates that combining statistical and semantic text*

Received: 29 Juni 2026 / Accepted: 11 Juli 2026 / Available online: 17 Juli 2026

representations produces an accurate and computationally efficient XSS detection system without relying on heavyweight deep learning architectures, contributing to SDG 9 (resilient digital infrastructure), SDG 16 (cybersecurity and data protection), and SDG 8 (digital economic productivity).

Keywords: *XSS Detection; Machine Learning; Hybrid Features; TF-IDF; BERT Embeddings*

1. Pendahuluan

Kerentanan aplikasi web tumbuh pada laju yang belum pernah tercatat sebelumnya. Sepanjang tahun 2025, total publikasi *Common Vulnerabilities and Exposures* (CVE) mendekati 48.185 entri rekor tertinggi sepanjang sejarah pencatatan kerentanan dengan *Cross-Site Scripting* (XSS) bertahan sebagai kategori kerentanan terbanyak dalam ekosistem web (Gamblin, 2026). OWASP mengklasifikasikan XSS ke dalam kategori A05:2025 *Injection*, mencatat lebih dari 8.000 CVE aktif yang berkaitan langsung dengan CWE-79 (*Improper Neutralization of Input During Web Page Generation*) (OWASP Foundation, 2025b). Setiap CVE merepresentasikan celah konkret yang dapat dieksploitasi untuk mencuri sesi pengguna, mendistribusikan *malware*, atau mengeksfiltrasi data sensitif dalam skala masif.

Dampak finansial dari eksploitasi XSS bersifat langsung dan terukur. Insiden British Airways tahun 2018 yang bersumber dari celah XSS pada pustaka JavaScript pihak ketiga mengekspos 380.000 transaksi pemesanan (Bright Security, 2025). (IBM Security, 2023) melaporkan bahwa rata-rata kerugian per insiden pelanggaran data yang difasilitasi oleh serangan injeksi mencapai 4,45 juta dolar AS. Secara teknis, serangan XSS memanfaatkan kegagalan aplikasi web dalam memvalidasi masukan sebelum merender konten HTML: penyerang menyuntikkan skrip JavaScript ke dalam halaman yang kemudian dieksekusi di *browser* korban dalam konteks origin yang dipercaya, secara efektif melewati mekanisme *Same-Origin Policy* (Stuttard & Pinto, 2011). (OWASP Foundation, 2025a) mengklasifikasikan XSS ke dalam tiga varian utama: *Stored* XSS yang disimpan secara permanen di server, *Reflected* XSS yang tersematkan dalam URL, dan *DOM-based* XSS yang terjadi sepenuhnya di sisi klien tanpa melibatkan respons dari server. Di Indonesia, ancaman ini semakin relevan seiring pesatnya transformasi digital yang mendorong pertumbuhan aplikasi web di sektor perbankan, pemerintahan, dan perdagangan elektronik. Rendahnya adopsi mekanisme pertahanan yang memadai, khususnya Web Application Firewall (WAF) yang dikonfigurasi dengan benar pada infrastruktur digital lokal memperbesar risiko eksposur terhadap serangan XSS dan potensi kerugian finansial maupun kebocoran data berskala besar.

Mekanisme pertahanan konvensional khususnya *Web Application Firewall* (WAF) berbasis pola statis terbukti tidak memadai menghadapi lanskap ancaman yang terus berevolusi. Penyerang modern tidak lagi mengandalkan *payload* sederhana, mereka memanfaatkan teknik *obfuscation* berlapis seperti *hexadecimal encoding*, *Unicode encoding*, *HTML entity encoding* dengan lebih dari 70 representasi valid per karakter, serta *polymorphic payload* yang menghasilkan representasi leksikal berbeda pada setiap eksekusi (OWASP Foundation, 2025b; Stuuxx, 2024). Penelitian (Ethiack, 2025) menunjukkan bahwa tingkat keberhasilan *bypass* meningkat dari 17,6% untuk *payload* sederhana hingga 70,6% untuk teknik *HTTP Parameter Pollution*, bahkan mencapai 100% pada beberapa konfigurasi WAF saat diuji dengan *fuzzer* berbasis kecerdasan buatan. Studi WAFFLED (Akhavani et al., 2025) lebih jauh mendemonstrasikan bahwa seluruh *framework* dan parser web yang diuji rentan

terhadap teknik *bypass* berbasis perbedaan interpretasi parsing antara WAF dan aplikasi target.

Sejumlah penelitian *machine learning* telah berupaya menutup kelemahan sistem berbasis pola. (Ayoubi et al., 2026) mengusulkan model *hybrid* CNN dan SVM yang mencapai akurasi tinggi, namun membutuhkan sumber daya komputasi besar dan tidak memanfaatkan representasi Transformer. (Wan, Xian, Wang, & Lu, 2024) mengombinasikan BERT *embeddings* dengan BiLSTM untuk mendeteksi *payload* XSS yang tersamarkan, ketergantungan pada arsitektur LSTM meningkatkan biaya *inference* tanpa mengeksploitasi karakteristik statistik distribusi token. (Bakır, 2025) mengajukan pendekatan UniEmbed berbasis *embedding* mirip BERT yang mencapai *accuracy* 96-98% pada *dataset* gabungan XSS dan SQL *Injection*, tetapi fokus yang terbagi membatasi kedalaman representasi untuk XSS secara spesifik. (Miczek et al., 2025) memanfaatkan *Large Language Model* (LLM) sebagai augmentasi data dan mencapai accuracy 99,5%, tetapi ketergantungan pada LLM menjadikan pendekatan ini sangat berat secara komputasi.

(Vignesh et al., 2025) mendemonstrasikan bahwa kombinasi *hybrid* TF-IDF dan BERT *embeddings* secara konsisten melampaui penggunaan salah satu metode secara independen pada tugas klasifikasi teks multibahasa. Landasan konseptualnya kuat: TF-IDF unggul dalam menangkap pola statistik-leksikal berulang yang bersifat diskriminatif terhadap kelas tertentu, sementara BERT menyediakan pemahaman kontekstual bidireksional yang *robust* terhadap variasi representasi permukaan (Devlin et al., 2019). Dua perspektif ini bersifat komplementer dan komplementaritas inilah yang belum divalidasi secara empiris pada domain *payload* XSS yang pendek, padat, dan kerap terobfuskasi.

Penelitian ini mengisi kesenjangan tersebut. Kontribusi utamanya adalah validasi empiris pendekatan *hybrid* TF-IDF berbasis karakter *n-gram* dan BERT *embeddings* untuk deteksi XSS secara spesifik, disertai analisis langsung terhadap bobot model dan sampel kesalahan klasifikasi aktual. Novelty penelitian ini terletak pada tiga aspek yang secara bersama belum dikombinasikan sebelumnya dalam literatur deteksi XSS: (1) validasi empiris pendekatan hybrid TF-IDF char_wb n-gram dan BERT embeddings secara spesifik pada domain payload XSS bukan domain teks umum, (2) perbandingan tiga algoritma klasifikasi pada representasi hybrid yang identik dengan prosedur cross-validation terkontrol, dan (3) penelusuran sumber kesalahan klasifikasi aktual ke komponen representasi yang bertanggung jawab, menghasilkan panduan pengembangan yang actionable. Secara lebih rinci, penelitian ini: (1) mengimplementasikan *pipeline* ekstraksi fitur *hybrid* yang menggabungkan TF-IDF char_wb *n-gram* (1,3) berdimensi 5.000 dengan BERT *embeddings* berdimensi 768 melalui *feature concatenation,* (2) membandingkan tiga algoritma klasifikasi *Logistic Regression*, SVM kernel RBF, dan *Random Forest* pada representasi *hybrid* yang identik, (3) mengevaluasi performa model menggunakan metrik accuracy, precision, recall, F1-score, dan ROC-AUC, serta (4) mengimplementasikan model terbaik sebagai prototipe aplikasi web XSSentinel berbasis Flask REST API dan antarmuka React.

Relevansi penelitian ini melampaui dimensi teknis dan bersinggungan langsung dengan Tujuan Pembangunan Berkelanjutan (TPB/Sustainable Development Goals/SDGs). Sistem deteksi XSS yang efektif dan efisien secara komputasi berkontribusi pada SDG 9.1 pembangunan infrastruktur digital yang andal dan aman sebagai fondasi transformasi digital inklusif. Perlindungan terhadap data pribadi pengguna dan keamanan akses informasi publik dari ancaman injeksi skrip mendukung SDG 16.10, yang menekankan

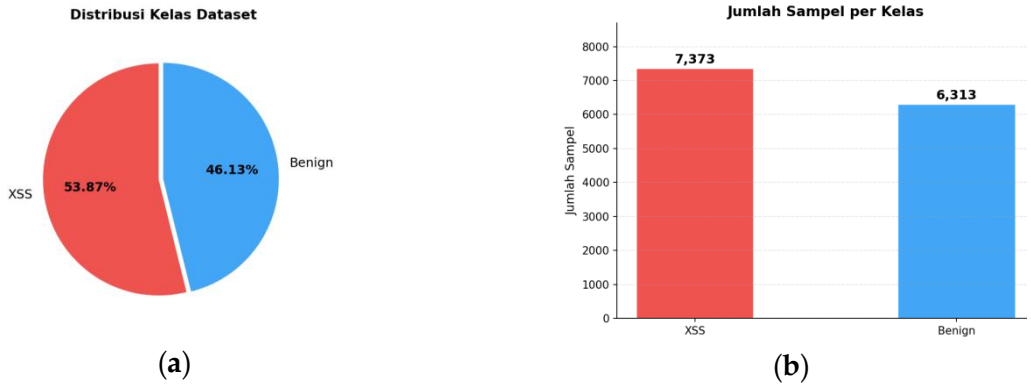
perlindungan kebebasan dasar dan privasi digital. Lebih jauh, kepercayaan digital yang terbangun melalui keamanan aplikasi web yang lebih kuat secara langsung mendukung SDG 8.2 melalui peningkatan produktivitas dan pertumbuhan ekonomi berbasis teknologi mengingat bahwa 4,45 juta dolar AS kerugian rata-rata per insiden pelanggaran data (IBM Security, 2023) merepresentasikan hambatan nyata terhadap pertumbuhan ekonomi digital. Dengan menyediakan lapisan deteksi yang dapat diimplementasikan tanpa infrastruktur GPU yang mahal, penelitian ini mendukung inklusivitas keamanan digital bagi organisasi berskala menengah ke bawah di Indonesia dan negara berkembang lainnya.

2. Metode

2.1. Dataset

Dataset yang digunakan adalah *XSS_dataset.csv*, sebuah dataset publik yang tersedia di platform Kaggle dan GitHub dan telah digunakan secara luas sebagai *benchmark* dalam penelitian deteksi XSS berbasis *machine learning*. *Dataset* terdiri dari 13.686 sampel *payload* teks berlabel biner: label 0 untuk *payload Benign* dan label 1 untuk *payload* XSS. Gambar 1 menunjukkan distribusi kelas pada *dataset* ini: 7.373 sampel (53,87%) berlabel XSS dan 6.313 sampel (46,13%) berlabel Benign. Rasio mayoritas-minoritas sebesar 1,17:1 ini dikategorikan sebagai *slight imbalance* dan tidak memerlukan teknik *oversampling* seperti SMOTE (Gogoi et al., 2021), sehingga *accuracy* tetap dapat digunakan sebagai indikator performa yang representatif, meskipun *F1-score* dan ROC-AUC tetap dipertahankan sebagai metrik pelengkap.

Variasi tekstual dalam dataset representatif terhadap lanskap ancaman nyata: *payload* XSS mencakup konstruksi sederhana seperti <script>alert(1)</script> hingga *payload* tersembunyi yang menerapkan *hexadecimal encoding*, *HTML entity encoding* berlapis, *event handler injection*, dan *polymorphic obfuscation*. *Dataset* dibagi menggunakan *stratified* random *splitting* dengan rasio 80:20 dan random *state* tetap (*seed*=42) untuk memastikan reprodusibilitas, mempertahankan rasio kelas pada Gambar 1 di kedua subset: *training set* (10.948 sampel) dan *testing set* (2.738 sampel), masing-masing dengan rasio XSS:*Benign* sebesar 1,17:1.



(**a**) (**b**)

Gambar 1. Distribusi kelas dataset XSS_dataset.csv (total 13.686 sampel). (a) Diagram lingkaran proporsi kelas: XSS 53,87% (7.373 sampel) dan Benign 46,13% (6.313 sampel), menunjukkan slight imbalance dengan rasio 1,17:1 yang tidak memerlukan teknik oversampling; (b) Diagram batang jumlah sampel absolut per kelas yang mengonfirmasi dominasi marginal kelas XSS.

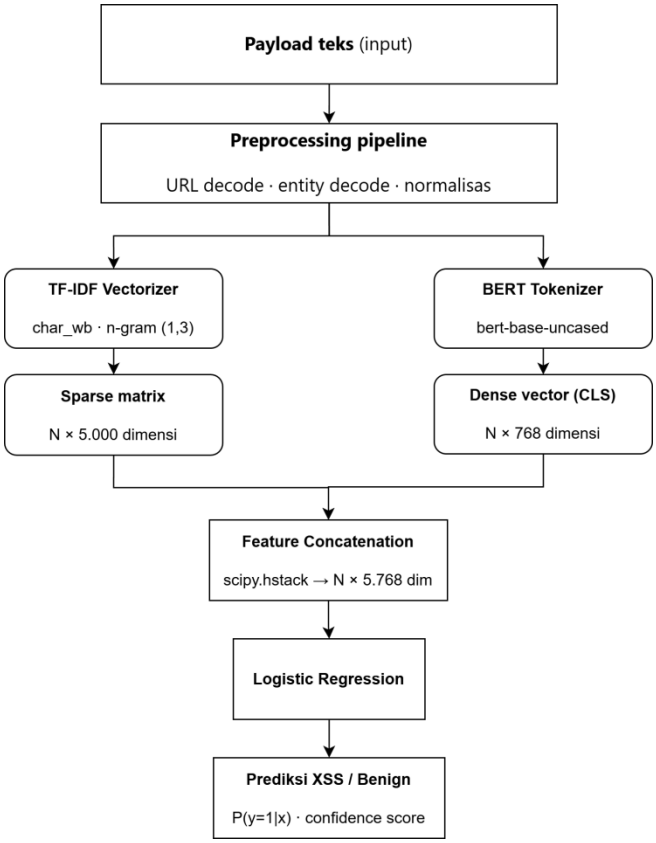
2.2. Preprocessing Payload

Seluruh *payload* diproses melalui tiga transformasi berurutan sebelum ekstraksi fitur dilakukan. Pertama, *URL decoding* menggunakan *urllib.parse.unquote()* mengkonversi karakter *percent-encoded* ke representasi aslinya, sehingga %3Cscript%3E diubah menjadi <script>. Kedua, *HTML entity decoding* menggunakan *html.unescape()* menangani representasi seperti <script>, menyatukan representasi yang berbeda secara leksikal namun identik secara semantik. Ketiga, normalisasi *whitespace* menggunakan operasi *split-join* menghilangkan spasi berlebihan yang dimanfaatkan sebagai teknik *obfuscation* tambahan. Fungsi *preprocessing* ini diterapkan secara identik saat *training* dan *inference*, mengikuti prinsip *fit-on-train*, *transform-on-all* untuk mencegah *data leakage* (Géron, 2019).

Sebagai implikasi dari desain pipeline ini, base64 encoding secara eksplisit berada di luar cakupan preprocessing yang diimplementasikan. Untuk pengembangan lanjutan, penambahan tahap base64 decoding sebagai transformasi keempat dalam urutan preprocessing direkomendasikan, sehingga payload yang menyembunyikan konstruksi JavaScript berbahaya dalam string base64 dapat dikonversi ke representasi aslinya sebelum ekstraksi fitur dilakukan.

2.3. Ekstraksi Fitur Hybrid

Representasi fitur dibangun dari dua sumber yang bersifat komplementer, kemudian digabungkan melalui *feature concatenation*. Ilustrasi alur *pipeline* ini ditunjukkan pada Gambar 2.



Gambar 2. Pipeline ekstraksi fitur hybrid. Payload diproses melalui preprocessing, diekstraksi secara paralel menjadi representasi TF-IDF (5.000 dimensi) dan BERT embeddings (768 dimensi), digabungkan melalui feature concatenation menjadi vektor 5.768 dimensi sebagai masukan Logistic Regression.

2.3.1. Fitur TF-IDF

Ekstraksi fitur TF-IDF menggunakan TfidfVectorizer dari Scikit-learn (Pedregosa et al., 2011) dengan konfigurasi *analyzer='char_wb'* dan rentang *n-gram* (1,3). Mode *char_wb* menghasilkan n-gram karakter dalam batas kata sehingga pola sintaksis *payload* seperti fragmen <sc, scr, ipt, ale, ert dapat ditangkap secara konsisten pada berbagai granularitas. Konfigurasi ini dipilih karena payload XSS umumnya sangat pendek dan mengandung karakter-karakter khusus yang tidak lazim dalam bahasa alami, tokenisasi berbasis kata menghasilkan vokabulari yang tidak stabil dan rentan terhadap variasi permukaan akibat *obfuscation (*Tamamura et al., 2023*)*. TfidfVectorizer di-*fit* hanya pada *training data*, menghasilkan matriks *sparse* berdimensi N×5.000.

2.3.2. BERT Embeddings

Representasi semantik diekstrak menggunakan model *pre-trained bert-base-uncased* dari *library* Hugging Face Transformers (Wolf et al., 2020). Model ini didasarkan pada arsitektur Transformer bidireksional yang diusulkan oleh (Devlin et al., 2019), memanfaatkan mekanisme *self-attention* untuk menangkap relasi kontekstual antar-token secara menyeluruh. Atribut *uncased* relevan karena *payload* XSS sering menggunakan variasi kapitalisasi sebagai teknik *obfuscation*. Setiap *payload* yang telah di-*preproses* dikonversi menjadi urutan token menggunakan BertTokenizer dengan panjang maksimum 128 token; representasi semantik diekstrak dari *embedding* token [*CLS*] pada lapisan terakhir dalam mode *torch.no_grad()* untuk efisiensi komputasi, menghasilkan vektor dense berdimensi 768 per *payload*. BERT digunakan sebagai *feature extractor* statis tanpa *fine-tuning* untuk menjaga efisiensi komputasi.

2.3.3. Feature Concatenation

Matriks BERT yang bersifat *dense* dikonversi ke format *sparse* menggunakan *csr_matrix* dari *scipy.sparse*, kemudian digabungkan secara horizontal dengan matriks TF-IDF menggunakan *scipy.sparse.hstack.* Representasi *hybrid* yang dihasilkan berdimensi N×5.768 (5.000 dari TF-IDF dan 768 dari BERT) menjadi masukan final bagi algoritma klasifikasi.

2.4. Pemilihan Algoritma dan Konfigurasi Model

Pemilihan algoritma dilakukan melalui eksperimen perbandingan empiris terhadap tiga kandidat: *Logistic Regression, Support Vector Machine* (SVM) kernel RBF, dan *Random Forest*. Ketiga algoritma dievaluasi menggunakan *Stratified 5-Fold Cross-Validation* pada *training set* dengan representasi fitur *hybrid* yang identik (Géron, 2019). *Logistic Regression* dikonfigurasi dengan *hyperparameter* C=1,0 (*inverse regularization strength L2*), *solver* L-BFGS, dan *max_iter=1.000.* Pemilihan ini didasarkan pada sifat konveksitas fungsi loss yang menjamin konvergensi ke solusi optimal global (Bishop, 2006), serta properti *white-box interpretability* yang memungkinkan inspeksi kontribusi tiap fitur melalui vektor bobot model.

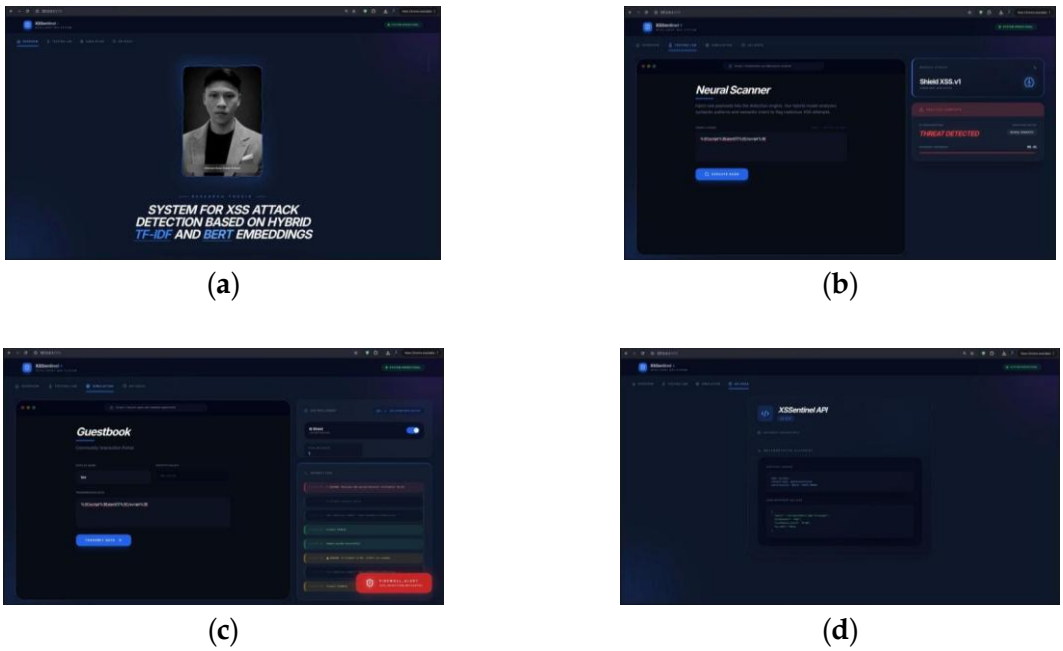
2.5. Evaluasi

Evaluasi dilakukan pada *testing set* yang sepenuhnya terpisah dari proses pelatihan. Metrik yang digunakan meliputi *accuracy*, *precision*, *recall*, *F1-score*, dan ROC-AUC. Model

dinyatakan akurat apabila seluruh metrik ≥98,00% dan ROC-AUC ≥0,99, dinyatakan adaptif apabila mampu mendeteksi *payload* XSS dari minimal lima kategori teknik *obfuscation URL encoding*, *HTML entity encoding*, *event handler injection*, *null byte injection*, dan *JavaScript URI injection* dengan *precision* dan *recall* ≥95,00% per kategori (OWASP Foundation, 2025a). Analisis *confusion matrix* dilakukan untuk mengidentifikasi distribusi prediksi secara detail, termasuk penelusuran sumber kesalahan ke komponen representasi yang bertanggung jawab.

2.6. Implementasi Prototipe

Model yang telah dilatih diimplementasikan dalam prototipe XSSentinel menggunakan arsitektur *decoupled client-server*. *Backend* berbasis Flask (Grinberg, 2018) mengekspos lima endpoint REST: GET /, GET /health, POST /predict, GET /metrics, dan GET /info. *Endpoint* /predict menerima *payload* teks dalam format JSON, menjalankan *pipeline preprocessing* dan ekstraksi fitur *hybrid* sebagaimana digambarkan pada Gambar 2, kemudian mengembalikan label prediksi beserta *confidence score* P(y=1|x) dari fungsi *sigmoid Logistic Regression*. *Frontend* dibangun menggunakan React (Meta Open Source, 2024) dan Vite sebagai *build tool*, menghadirkan empat halaman antarmuka yang ditunjukkan pada Gambar 3: (a) *Overview* berisi ringkasan penelitian, (b) *Testing Lab* tempat pengguna menginput *payload* dan menerima hasil deteksi beserta *confidence score* secara *real-time*, (c) *Simulation* yang mendemonstrasikan skenario perlindungan formulir web melalui *toggle AI Shield*, dan (d) *API Docs* yang menyajikan dokumentasi teknis *endpoint*. Seluruh artefak pelatihan disimpan dalam format .pkl menggunakan *joblib* untuk memastikan konsistensi *pipeline* antara training dan *inference*.



(**a**) (**b**)

(**c**) (**d**)

Gambar 3. Antarmuka prototipe XSSentinel. (a) Halaman *Overview*; (b) Halaman *Testing Lab* menampilkan hasil deteksi *THREAT DETECTED* pada *payload* %3Cscript%3Ealert(1)%3C/script%3E; (c) Halaman *Simulation* dengan notifikasi *FIREWALL_ALERT* saat *AI Shield* aktif memitigasi *payload* serupa; (d) Halaman *API Docs* berisi dokumentasi teknis *endpoint*.

3. Hasil dan Pembahasan

3.1. Perbandingan Algoritma Klasifikasi

Tabel 1 menyajikan hasil *Stratified 5-Fold Cross-Validation* pada *training set* untuk tiga kandidat algoritma dengan representasi fitur *hybrid* yang identik. *Logistic Regression* mengungguli SVM kernel RBF dan *Random Forest* secara konsisten pada seluruh metrik evaluasi.

Tabel 1. Perbandingan Empiris Algoritma Klasifikasi

Algoritma	Accuracy	Precision	Recall	F1-Score
Logistic Regression	99,71%	99,74%	99,73%	99,73%
SVM (kernel RBF)	99,12%	99,18%	99,09%	99,13%
Random Forest	98,83%	98,91%	98,76%	98,83%

Keunggulan *Logistic Regression* atas SVM kernel RBF (+0,60 poin *F1-score*) dapat dijelaskan oleh karakteristik representasi *hybrid 5.768-dimensional* yang sudah cukup *linearly separable*, sehingga transformasi *non-linear* kernel RBF tidak memberikan peningkatan signifikan namun menambah biaya komputasi (Vapnik, 1998). *Random Forest* kurang optimal untuk fitur *sparse* berdimensi sangat tinggi karena setiap pohon hanya mempertimbangkan *subset* fitur secara acak, sehingga fitur TF-IDF yang diskriminatif berpotensi terpinggirkan (Géron, 2019). Selain keunggulan performa, *Logistic Regression* dipilih karena fungsi *loss* yang konveks menjamin konvergensi ke solusi optimal global, serta properti *white-box interpretability* yang penting untuk sistem keamanan (Bishop, 2006).

3.2. Performa Model Hybrid TF-IDF + BERT

Evaluasi pada *testing set* (n=2.738) menghasilkan performa yang melampaui seluruh *threshold* kinerja yang ditetapkan. Tabel 2 menyajikan hasil lengkap model *hybrid*.

Tabel 2. Hasil Evaluasi Model Hybrid TF-IDF + BERT Embeddings

Metrik Evaluasi	Nilai	Interpretasi
Accuracy	99,78%	Prediksi benar dari seluruh sampel testing
Precision	99,80%	Dari prediksi XSS, 99,80% benar-benar XSS (FP sangat rendah)
Recall	99,80%	Model mendeteksi 99,80% dari seluruh payload XSS aktual (FN sangat rendah)
F1-Score	99,80%	Keseimbangan optimal precision dan recall antar kelas
ROC-AUC	1,0000	Kemampuan diskriminasi sempurna pada distribusi data yang digunakan

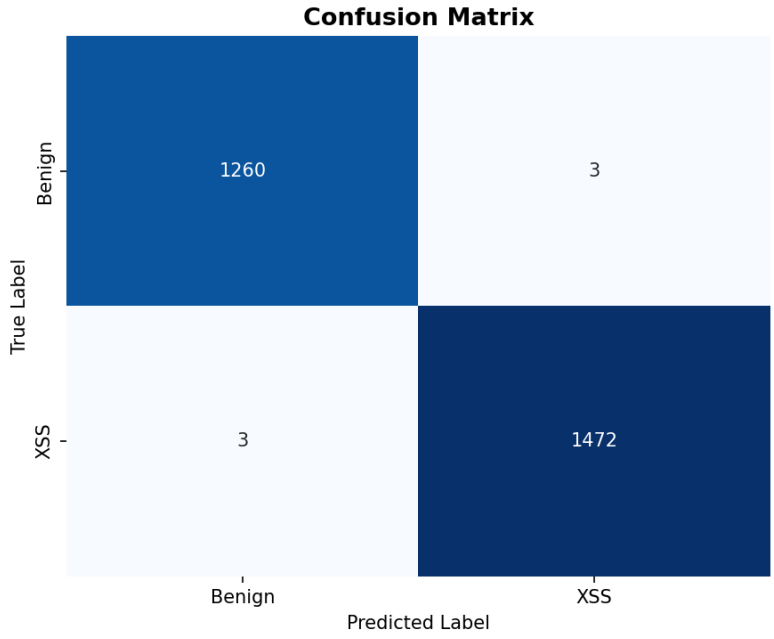
Nilai *F1-score* yang identik dengan *precision* dan *recall* menunjukkan bahwa model tidak bias terhadap salah satu kelas penting dalam konteks keamanan web di mana kedua jenis kesalahan memiliki konsekuensi serius: *false negative* membiarkan *payload* berbahaya *lolos*, sementara *false positive* mengganggu pengalaman pengguna yang sah.

3.3. Analisis Confusion Matrix

Dari 2.738 sampel *testing*, model menghasilkan hanya enam kesalahan klasifikasi: tiga *False Positive* (*payload Benign* yang salah diprediksi sebagai XSS) dan tiga *False Negative* (*payload* XSS yang lolos dari deteksi). Tabel 3 menampilkan distribusi prediksi secara lengkap, visualisasi heatmapnya ditunjukkan pada Gambar 4.

Tabel 3. Confusion Matrix Model Hybrid

	Prediksi: Benign	**Prediksi: XSS**
Aktual: Benign	TN = 1.260	FP = 3
Aktual: XSS	FN = 3	TP = 1.472



Gambar 4. Confusion matrix model hybrid pada testing set (n=2.738): 1.260 True Negative, 3 False Positive, 3 False Negative, 1.472 True Positive.

Dari nilai tersebut diperoleh *Specificity* (*True Negative Rate*) = 99,76%, *False Positive Rate* = 0,24%, dan *False Negative Rate* = 0,20%. Nilai FN yang hanya tiga sampel mengindikasikan sistem hampir tidak pernah membiarkan *payload* XSS lolos tanpa terdeteksi, prioritas utama dalam sistem deteksi ancaman keamanan.

3.4. Analisis Fitur dan Sumber Kesalahan Klasifikasi

Inspeksi langsung terhadap vektor bobot (*coef_*) model yang telah dilatih mengungkapkan bahwa fitur-fitur paling diskriminatif terhadap kelas XSS bukan semata-

mata *trigram* tag klasik seperti <sc, scr, dan ipt, melainkan fragmen yang berkaitan dengan pola eksekusi fungsi JavaScript: fragmen skema URL (/., p:/), referensi *document*. (cum, nt.), serta pola pemanggilan fungsi berargumen numerik seperti (1) dan ert dari "alert". Temuan ini menunjukkan bahwa model lebih bergantung pada sinyal yang konsisten lintas berbagai teknik *obfuscation* dibandingkan kehadiran tag <script> secara eksplisit. Rata-rata magnitudo bobot pada blok BERT tercatat sekitar dua belas kali lebih besar dibandingkan blok TF-IDF, mengindikasikan representasi semantik sebagai penyumbang utama daya pisah kelas.

Ketiga *False Negative* adalah *payload* yang menyembunyikan <script>alert("XSS")</script> dalam *encoding base64* yang dibungkus dalam atribut *SRC* elemen *EMBED* skema *encoding* yang tidak ditangani oleh *preprocessing* dan tidak pernah terlihat oleh representasi TF-IDF maupun BERT. Pada kasus ini, kedua komponen representasi secara bersamaan mengarah ke prediksi *Benign*, karena string *base64* terbaca sebagai rangkaian karakter acak tanpa semantik berbahaya yang terdeteksi. Ini bukan kegagalan satu metode yang tidak terkoreksi metode lain, melainkan titik buta yang dibagi bersama dan letak penyebabnya ada di tahap *preprocessing*, bukan pada pemilihan model. Ketiga *False Positive*, sebaliknya, adalah fragmen penutup sintaksis JavaScript yang umum dan tidak berbahaya-}); dan }}();-yang dibaca oleh *bert-base-uncased* (dilatih pada korpus bahasa alami umum) sebagai konstruksi yang mencurigakan, dengan magnitudo kontribusi BERT yang jauh lebih besar dari TF-IDF sehingga koreksi tidak dapat dilakukan.

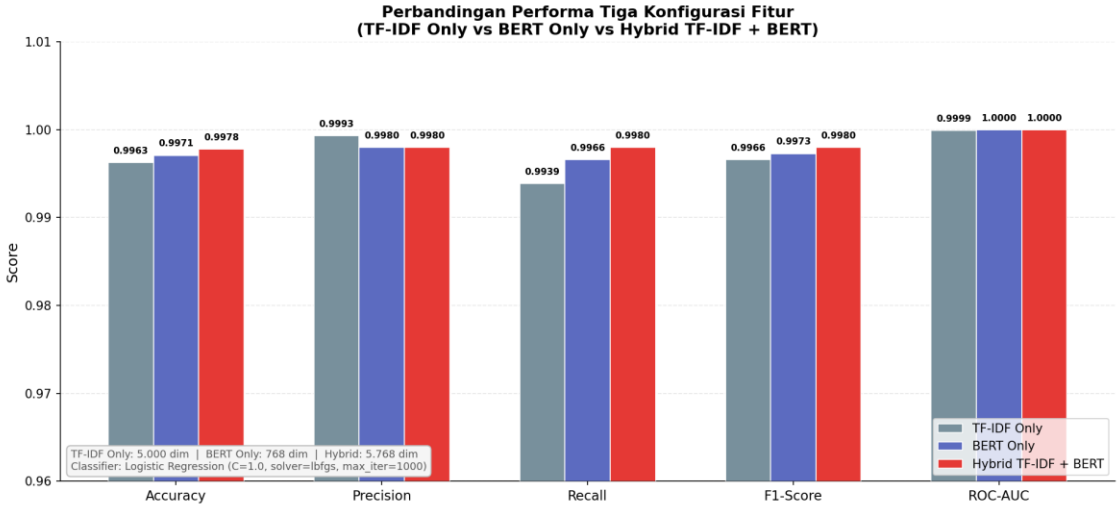
Temuan false negative berbasis base64 ini menghadirkan dua jalur pengembangan dengan karakteristik yang berbeda. Jalur pertama adalah penambahan base64 decoding sebagai tahap preprocessing keempat, yang secara deterministik mengonversi semua string base64 ke representasi aslinya sebelum ekstraksi fitur. Keunggulannya adalah jaminan cakupan yang komprehensif tanpa mengubah arsitektur model, keterbatasannya adalah risiko memproses konten base64 yang sah seperti gambar inline dalam URI data secara tidak perlu, yang berpotensi memperkenalkan noise ke dalam representasi fitur. Jalur kedua adalah augmentasi dataset pelatihan dengan payload XSS berbasis base64 yang telah didekode, membiarkan pipeline preprocessing tetap sederhana namun melatih model agar lebih sensitif terhadap pola semantik payload tersembunyi. Keunggulannya adalah tidak mengubah alur inference, keterbatasannya adalah membutuhkan kurasi dan labeling dataset tambahan yang cermat agar tidak memperkenalkan noise label. Untuk sistem yang memprioritaskan false negative rate yang rendah sebagaimana konteks deteksi ancaman keamanan penambahan base64 decoding ke preprocessing direkomendasikan sebagai prioritas pertama karena memberikan jaminan cakupan yang lebih deterministik tanpa bergantung pada distribusi data pelatihan. Kedua jalur dapat dikombinasikan untuk hasil yang lebih robust.

3.5. Perbandingan Konfigurasi Fitur

Untuk membuktikan kontribusi nyata pendekatan *hybrid*, tiga konfigurasi dievaluasi pada kondisi yang identik dengan algoritma dan *hyperparameter* yang sama. Tabel 4 dan Tabel 5 menyajikan perbandingan metrik dan confusion matrix, Gambar 5 dan Gambar 6 memvisualisasikan hasilnya.

Tabel 4. Perbandingan Performa Tiga Konfigurasi Representasi Fitur

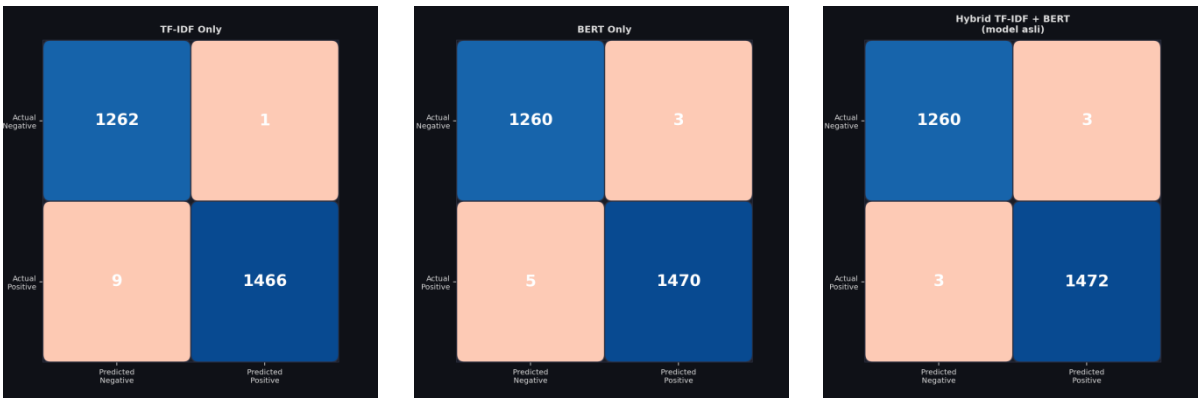
Konfigurasi	Accuracy	Precision	Recall	F1-Score	ROC-AUC
TF-IDF Only (5.000 dim)	99,63%	99,93%	99,39%	99,66%	0,9999
BERT Only (768 dim)	99,71%	99,80%	99,66%	99,73%	1,0000
Hybrid TF-IDF + BERT (5.768 dim)	**99,78%**	**99,80%**	**99,80%**	**99,80%**	**1,0000**
Delta: Hybrid vs TF-IDF Only	+0,15%	-0,13%	+0,41%	+0,14%	+0,0001
Delta: Hybrid vs BERT Only	+0,07%	+0,00%	+0,14%	+0,07%	+0,0000



Gambar 5. Perbandingan performa tiga konfigurasi fitur (TF-IDF Only, BERT Only, Hybrid) pada lima metrik evaluasi.

Tabel 5. Perbandingan Confusion Matrix Tiga Konfigurasi Fitur

Konfigurasi	TN	FP	FN	TP
TF-IDF Only	1.262	1	9	1.466
BERT Only	1.260	3	5	1.470
Hybrid TF-IDF + BERT	1.260	3	**3**	**1.472**



(**a**) (**b**) (**c**)

Gambar 6. Perbandingan confusion matrix tiga konfigurasi fitur. (a) TF-IDF Only; (b) BERT Only; (c) Hybrid TF-IDF + BERT.

Pendekatan *hybrid* mengungguli kedua konfigurasi fitur tunggal pada empat dari lima metrik. Peningkatan *recall* sebesar 0,41 poin persentase dibandingkan TF-IDF saja merupakan kontribusi yang paling substansial secara klinis: dalam skenario deteksi ancaman, setiap *false negative* merepresentasikan *payload* berbahaya yang lolos dari sistem proteksi. Pola pada *confusion matrix* memperlihatkan mekanisme komplementaritas yang konkret: TF-IDF saja menghasilkan hanya satu *false positive* (*precision* tertinggi) tetapi sembilan *false negative* beroperasi secara konservatif namun terbatas dalam mengenali variasi leksikal yang tidak konvensional. BERT saja membalik proporsi ini, menurunkan *false negative* menjadi lima dengan *false positive* naik menjadi tiga. Model *hybrid* mewarisi presisi BERT terhadap trafik *Benign* sekaligus menekan *false negative* ke angka paling rendah (tiga), menghasilkan konfigurasi paling seimbang. Temuan bahwa BERT saja (99,71%) mengungguli TF-IDF saja (99,63%) mengindikasikan representasi semantik sebagai penyumbang dominan, sementara TF-IDF berperan sebagai penyempurna yang efektif pada *payload* bertepi tipis.

3.6. Perbandingan dengan Penelitian Terdahulu

Tabel 6 menempatkan hasil penelitian ini dalam konteks literatur yang ada. Perbandingan didasarkan pada *accuracy* dan *F1-score* sebagai metrik yang paling umum dilaporkan dalam penelitian deteksi XSS.

Tabel 6. Perbandingan Performa dengan Penelitian Terdahulu

Penelitian	Metode	Accuracy	F1-Score	Catatan
(Ayoubi et al., 2026)	CNN + SVM	~99,00%	~98,90%	Dataset berbeda; komputasi berat
(Wan et al., 2024)	BERT + BiLSTM	~98,70%	~98,50%	LSTM berat; tanpa TF-IDF
(Bakır, 2025)	UniEmbed + ML	96-98%	95-97%	Fokus terbagi XSS + SQLi
(Miczek et al.,	LLM	~99,50%	~99,40%	LLM mahal; low-resource

Penelitian	Metode	Accuracy	F1-Score	Catatan
2025)	augmentation + ML			
Penelitian ini (2026)	**Hybrid TF-IDF + BERT + LR**	**99,78%**	**99,80%**	CPU-friendly; prototipe Flask

Model yang dihasilkan penelitian ini melampaui pendekatan BERT + BiLSTM (Wan et al., 2024) dan UniEmbed (Bakır, 2025), serta mendekati performa pendekatan berbasis LLM (Miczek et al., 2025) dengan *accuracy* 99,78% dan *F1-score* 99,80%. Keunggulan kompetitif ini dicapai dengan arsitektur yang jauh lebih ringan: *Logistic Regression* sebagai *classifier*, tanpa *fine-tuning* BERT, dan tanpa ketergantungan pada LLM skala besar. Implikasinya signifikan untuk implementasi praktis: sistem deteksi XSS berbasis *hybrid* ini dapat dijalankan pada server CPU tanpa infrastruktur GPU, menjadikannya layak untuk lingkungan produksi berskala kecil hingga menengah.

Penting untuk dicatat bahwa perbandingan pada Tabel 6 bersifat indikatif, bukan konklusif secara statistik. Setiap penelitian menggunakan dataset yang berbeda dalam hal ukuran sampel, sumber payload, distribusi kelas, dan proporsi teknik obfuscation yang terwakili. Perbedaan-perbedaan ini membatasi komparabilitas langsung metrik accuracy dan F1-score secara absolut, selisih performa yang terukur harus diinterpretasikan sebagai indikasi keunggulan relatif pendekatan, bukan klaim superioritas yang dapat digeneralisasi tanpa validasi lintas dataset yang terkontrol. Perbandingan ini dimaksudkan untuk menempatkan kontribusi penelitian dalam lanskap metodologis yang ada, bukan sebagai pembuktian absolut superioritas pada semua kondisi pengujian.

3.7. Pengujian Adaptivitas terhadap Teknik Obfuscation

Pengujian fungsionalitas prototipe dilakukan pada sepuluh *payload* representatif yang mencakup seluruh lima kategori teknik *obfuscation* yang ditetapkan dalam definisi operasional. Seluruh *payload* diprediksi dengan benar, termasuk *null byte injection* (*confidence* 99,32%), *event handler injection* (*confidence* 99,99%), dan *JavaScript URI injection* (*confidence* 99,86%). *Confidence score* terendah tercatat pada sampel *Benign* berupa *form data* normal (96,74%), yang berada jauh di atas ambang keputusan 50%. Seluruh *threshold* adaptivitas ≥95,00% per kategori terpenuhi, mengonfirmasi kemampuan model mendeteksi berbagai varian serangan XSS di luar pola sederhana.

Kesimpulan

Penelitian ini membuktikan bahwa penggabungan fitur statistik TF-IDF berbasis karakter *n-gram* (1,3) dengan representasi semantik BERT *embeddings* (token [*CLS*]) melalui *feature concatenation* menghasilkan sistem deteksi XSS yang akurat, adaptif, dan efisien secara komputasi. Pada *dataset* 13.686 sampel berlabel, model *Logistic Regression* yang dilatih pada representasi *hybrid 5.768-dimensional* mencapai *F1-score* 99,80% dan ROC-AUC 1,0000 pada *testing set* melampaui konfigurasi TF-IDF tunggal (*F1-score*=99,66%) dan BERT tunggal (*F1-score*=99,73%) secara konsisten, dengan hanya enam kesalahan klasifikasi dari 2.738 sampel.

Kontribusi metodologis utama adalah validasi empiris pendekatan *hybrid* TF-IDF dan BERT *embeddings* pada domain *payload* XSS secara spesifik, disertai penelusuran langsung sumber kesalahan ke komponen representasi yang bertanggung jawab. Temuan bahwa dua *false negative* utama bersumber dari *base64 encoding* yang tidak tertangani *preprocessing* bukan dari kelemahan model memberikan panduan *actionable*: penambahan tahap *base64 decoding* pada *pipeline preprocessing* merupakan prioritas pertama untuk pengembangan lanjutan.

Beberapa keterbatasan perlu dicatat. Evaluasi dilakukan pada satu dataset publik yang mungkin tidak merepresentasikan seluruh ragam *payload* XSS di lingkungan produksi nyata. Prototipe XSSentinel bersifat demonstratif dan belum diuji pada skenario deployment dengan volume trafik tinggi, aspek skalabilitas horizontal, model *quantization*, dan optimasi *ONNX runtime* perlu dikaji sebelum implementasi produksi. Penelitian selanjutnya dapat mengeksplorasi penambahan *base64 decoding* sebagai komponen *preprocessing* keempat, *fine-tuning* BERT pada korpus keamanan web seperti laporan CVE dan kode JavaScript, serta penerapan teknik *Explainable AI* (XAI) seperti SHAP untuk memberikan penjelasan prediksi yang dapat diinterpretasikan pada tingkat fitur individual.

Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada Erna Dwi Astuti, M.Kom. dan Nahar Mardiyantoro, M.Kom. selaku pembimbing atas arahan dan masukan yang berharga selama penelitian ini berlangsung, serta kepada Program Studi Teknik Informatika Universitas Sains Al Qur'an atas dukungan akademik yang diberikan.

Daftar Pustaka

Akhavani, S. A., Jabiyev, B., Kallus, B., Topcuoglu, C., Bratus, S., & Kirda, E. (2025). WAFFLED: Exploiting Parsing Discrepancies to Bypass Web Application Firewalls. *2025 IEEE Annual Computer Security Applications Conference (ACSAC)*, 689–702. IEEE. https://doi.org/10.1109/ACSAC67867.2025.00062

Ayoubi, A., Laaouina, L., Jeghal, A., & Tairi, H. (2026). An Improved Detection of Cross-Site Scripting (XSS) Attacks Using a Hybrid Approach Combining Convolutional Neural Networks and Support Vector Machine. *Journal of Cybersecurity and Privacy*, *6*(1), 18. https://doi.org/10.3390/jcp6010018

Bakır, R. (2025). UniEmbed: A Novel Approach to Detect XSS and SQL Injection Attacks Leveraging Multiple Feature Fusion with Machine Learning Techniques. *Arabian Journal for Science and Engineering*, *50*, 15591–15604. https://doi.org/10.1007/s13369-024-09916-4

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bright Security. (2025). *XSS Attack: 3 Real Life Attacks and Code Examples*. Retrieved from https://brightsec.com/blog/xss-attack/

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of NAACL-HLT 2019*, 4171–4186. Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Ethiack. (2025). *Bypassing WAFs to Deliver XSS Payloads Using HTTP Parameter Pollution*. Retrieved from https://cybersecuritynews.com/wafs-protection-bypassed/

Gamblin, J. (2026). *2025 Vulnerability Summary: A New Record of 48,185 CVEs*. Retrieved from https://www.gvip-project.org/blog/2026/2025-cve-report/

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.

Gogoi, B., Ahmed, T., & Saikia, H. K. (2021). Detection of XSS attacks in web applications: A machine learning approach. *International Journal of Innovative Research in Computer Science & Technology*, *9*(1), 1–10. https://doi.org/10.21276/ijircst.2021.9.1.1

Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python* (2nd ed.). O'Reilly Media.

IBM Security. (2023). *Cost of a Data Breach Report 2023*. Retrieved from https://www.ibm.com/reports/data-breach

Meta Open Source. (2024). *React Documentation*. Retrieved from https://react.dev

Miczek, D., Bieszczad, J., & Kwasniewska, A. (2025). Leveraging LLM to strengthen ML-based cross-site scripting detection. *arXiv Preprint arXiv:2504.21045*. Retrieved from https://arxiv.org/abs/2504.21045

OWASP Foundation. (2025a). *Cross-Site Scripting (XSS)*. Retrieved from https://owasp.org/www-community/attacks/xss/

OWASP Foundation. (2025b). *OWASP Top 10:2025 – A05:2025 Injection*. Retrieved from https://owasp.org/Top10/2025/A05_2025-Injection/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Stuttard, D., & Pinto, M. (2011). *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws* (2nd ed.). Wiley.

Stuuxx. (2024). *Exploiting Cross-Site Scripting Vulnerabilities with Polymorphic Payloads*. Retrieved from https://stuuxx.netlify.app/2024/05/23/exploiting-cross-site-scripting-vulnerabilities-with-polymorphic-payloads/

Tamamura, K., Nishikawa, H., & Nakayama, H. (2023). Detection of XSS Attacks with One Class SVM Using TF-IDF. *2023 IEEE International Conference on Computing (ICOCO)*, 45–52. IEEE. https://doi.org/10.1109/ICOCO59262.2023.10397619

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley.

Vignesh, S., Kumar, K., & Vijayan, S. (2025). SKVtrio@LT-EDI-2025: Hybrid TF-IDF and BERT embeddings for multilingual homophobia and transphobia detection. *Proceedings of the Fifth Workshop on Language Technology for Equality, Diversity and Inclusion (LT-EDI 2025)*, 88–95. Association for Computational Linguistics.

Wan, S., Xian, B., Wang, Y., & Lu, J. (2024). Methods for Detecting XSS Attacks Based on BERT and BiLSTM. *Proceedings of the 2024 8th International Conference on Management Engineering, Software Engineering and Service Sciences (ICMSS)*, 112–120. IEEE. https://doi.org/10.1109/ICMSS61211.2024.00008

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., … Gugger, S. (2020). Transformers: State-of-the-art Natural Language Processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. Association for Computational Linguistics.