DECODE: JURNAL PENDIDIKAN TEKNOLOGI INFORMASI
P-ISSN: 2775-2984 | E-ISSN: 2775-1813

Vol. 5, No 3, November 2025, Hal: 888-902
Doi: http://dx.doi.org/10.51454/decode.v5i3.1313
https://journal.umkendari.ac.id/index.php/decode

Implementasi Deteksi Intrusi Aplikasi Web Berbasis Supervised Machine Learning: Studi Kasus LMS STT Terpadu Nurul Fikri

Henry Saptono[1], Yuda Fatahillah Achmar[2], Hendro Sasongko Hadi[2], Salman Fathy Shiroth[3], Lambang Ramadhian Putra Aria[1], Muhamad Masayid Alfarizqi[1]

[1]Program Studi Teknik Informatika, Sekolah Tinggi Teknologi Terpadu Nurul Fikri, Indonesia.
[2]Program Studi Sistem Informasi, Sekolah Tinggi Teknologi Terpadu Nurul Fikri, Indonesia.
[3]Program Studi Bisnis Digital, Sekolah Tinggi Teknologi Terpadu Nurul Fikri, Indonesia.

Artikel Info

Kata Kunci:

Deteksi intrusi;
Lms;
Machine learning;
Redis;
Supervised learning.

Keywords:

Intrusion detection;
Lms;
Machine learning;
Redis;
Supervised learning.

Riwayat Artikel:

Submitted: 05 Agustus 2025
Accepted: 19 Oktober 2025
Published: 20 Oktober 2025

Abstrak: Sistem deteksi intrusi merupakan komponen penting dalam menjaga keamanan aplikasi web, terutama pada platform pembelajaran daring seperti *Learning Management System (LMS).* Penelitian ini bertujuan mengimplementasikan sistem deteksi intrusi berbasis *supervised machine learning* untuk mengidentifikasi serangan *SQL Injection* dan *Cross-Site Scripting (XSS)* melalui analisis *payload HTTP* yang diterima oleh *LMS Moodle*. Model yang digunakan adalah algoritma *Random Forest* dengan representasi fitur berbasis (*Term Frequency–Inverse Document Frequency) TF-IDF* pada level karakter. Data pelatihan berasal dari gabungan dataset publik dan log aktivitas *LMS* internal yang telah melalui proses *preprocessing* serta *masking* data sensitif. Arsitektur sistem dirancang menggunakan *plugin middleware* pada LMS untuk menangkap log secara *real-time*, *Redis* sebagai *message broker*, dan *Flask-RQ* sebagai *worker* pemrosesan model, serta *dashboard Grafana-Loki* untuk visualisasi hasil deteksi. Hasil pengujian menunjukkan bahwa model *Random Forest* mencapai akurasi 99,94% dengan nilai presisi, *recall*, dan *AUC* yang sangat tinggi, menunjukkan kemampuan deteksi yang andal terhadap serangan *SQL Injection* dan *XSS*. Sistem ini mampu beroperasi secara *real-time* tanpa mengganggu kinerja *LMS*, sehingga efektif diterapkan sebagai solusi keamanan siber pada lingkungan pendidikan. Implementasi ini berpotensi dikembangkan lebih lanjut untuk mendeteksi jenis serangan web lainnya secara adaptif.

Abstract: *Intrusion attacks. Intrusion detection systems are essential components in maintaining the security of web applications, particularly in online learning platforms such as Learning Management Systems (LMS). This study aims to implement a supervised machine learning–based intrusion detection system to identify SQL Injection and Cross-Site Scripting (XSS) attacks through the analysis of HTTP payloads received by the Moodle LMS. The model employed is the Random Forest algorithm with feature representation based on character-level Term Frequency–Inverse Document Frequency (TF-IDF). The training data were obtained from a combination of public datasets and internal LMS activity logs that had undergone preprocessing and sensitive data masking. The system architecture is designed using a middleware plugin on the LMS to capture logs in real time, Redis as a message broker, and Flask-RQ as a model processing worker, with a Grafana-Loki dashboard for detection result visualization. Experimental results show that the Random Forest model achieved an accuracy of 99.94%, with very high precision, recall, and AUC values, demonstrating reliable detection capability for SQL Injection and*

XSS attacks. The system operates in real time without affecting LMS performance, making it an effective cybersecurity solution for educational environments. This implementation has the potential to be further developed to detect other types of web attacks adaptively.

Corresponding Author:
Henry Saptono
Email: henry@nurulfikri.ac.id

PENDAHULUAN

Dalam lanskap digital kontemporer, aplikasi web telah menjadi infrastruktur kritis yang mendukung operasi sektor pendidikan, pemerintahan, dan bisnis (Sugara & Sriyasa, 2024). Keandalan *platform* ini menentukan kontinuitas layanan dan menjadi garda utama untuk melindungi data sensitif dari ancaman siber yang semakin canggih (Fau zi, Hermawan, Adhy, & Maesaroh, 2024). *Learning Management System (LMS)* seperti Moodle memainkan peran penting dalam ekosistem pendidikan tinggi sebagai sarana distribusi konten, evaluasi, dan kolaborasi; namun meningkatnya ketergantungan juga membuka celah terhadap serangan intrusi (Veluvali & Surisetti, 2021).

Data empiris menunjukkan bahwa *SQL Injection*, *Cross-Site Scripting (XSS)*, dan *Distributed Denial of Service (DDoS)* adalah vektor serangan dominan yang berpotensi melumpuhkan layanan, mencuri data akademik, dan menimbulkan kerugian finansial serta reputasi (Prasetyo, Haeruddin, & Ariesryo, 2024; Risky & Yuhandri, 2024). Pendekatan konvensional berbasis *signature* (mis. *WAF* dan *IDS signature-based*) efektif untuk serangan yang sudah dikenal dan cenderung menghasilkan sedikit *false positive*, tetapi kesulitan mendeteksi varian baru atau serangan *zero-day* tanpa pembaruan *signature* yang rutin—sebuah beban operasional yang signifikan bagi institusi pendidikan dengan sumber daya TI terbatas (Hubballi & Suryanarayanan, 2014; Sangadji, Muhammad, & Gunawan, 2023; Setiawan, Munandar, & Astuti, 2022).

Pendekatan *supervised* dan *semi-supervised machine learning* menawarkan alternatif adaptif yang mampu mengenali pola serangan termasuk varian yang belum terdokumentasi; kajian literatur terkini juga menekankan peran *mahcine learning (ML)* dalam pemantauan lalu lintas jaringan dan deteksi serangan secara efektif. Studi-studi menunjukkan akurasi sangat tinggi pada tugas deteksi intrusi dan memberikan wawasan metodologis untuk pemodelan trafik dan ancaman. (Das et al., 2024; Wang et al., 2023; Genuario et al., 2024). Algoritma seperti *Random Forest*, *SVM*, dan *Decision Tree* terbukti efektif menganalisis fitur *HTTP request* dan *payload* untuk mendeteksi *SQLi* dan *XSS*; penelitian komparatif juga menunjukkan keunggulan *Random Forest* pada deteksi *SQL Injection* dibandingkan beberapa metode lainnya. (Rattrout, Al-Hawawreh, & Sitnikova, 2023; Rahayu, Yulyanti & Ghalib, 2023; Aulianoor & Koprawi, 2024). Teknik *ensemble* dan *boosting* (mis. *AdaBoost*, *GBDT*) turut dilaporkan efektif dalam menangani data berdimensi tinggi dan varian serangan baru, sesuai temuan di literatur ulasan *ML* untuk keamanan siber. (Genuario et al., 2024).

Namun, implementasi *ML* pada *LMS* menghadapi beberapa tantangan penting: ketidakseimbangan kelas dalam *dataset*, kebutuhan efisiensi komputasi untuk inferensi *real-time*, serta kompleksitas integrasi tanpa mengganggu layanan utama *LMS*. Masalah operasional pada *platform* pembelajaran elektronik (termasuk aspek implementasi dan pemeliharaan model) juga telah didokumentasikan khusus untuk konteks *e-learning*. (Shiddiq, Karna, & Irawati, 2025; Tony Tan et al., 2023; Nuruddin, Sadikin, & Saputra, 2023; Amirah & Sanmorino, 2023).

Untuk menjawab tantangan tersebut, penelitian ini merancang arsitektur *Intrusion Detection System (IDS)* terintegrasi untuk *LMS* Moodle di STT Terpadu Nurul Fikri. Arsitektur yang diusulkan memiliki empat komponen utama: (1) *plugin middleware* Moodle untuk menangkap *HTTP request* secara *real-time*; (2) *message broker publish–subscribe* berbasis Redis; (3) worker berbasis Flask dan Redis Queue (RQ) yang menjalankan model Random Forest yang dioptimasi menggunakan *SMOTE-ENN* untuk mengatasi ketidakseimbangan; dan (4) *dashboard* berbasis Grafana untuk visualisasi anomali secara

interaktif dan monitoring *real-time*—praktik visualisasi serupa telah diterapkan dalam sistem *monitoring server* modern. (Wulandari et al., 2018; Febriyani et al., 2019; Husna & Rosyani, 2023; Rasyidi & Pratama, 2024). Rancangan ini bertujuan menjaga latensi rendah, responsivitas terhadap ancaman *zero-day*, dan kemudahan integrasi ke *LMS* yang ada.

Berdasarkan uraian masalah dan tujuan arsitektur di atas, penelitian ini mengarahkan fokusnya pada pertanyaan penelitian berikut: (1) algoritma *supervised machine learning* mana—*Random Forest*, *Decision Tree*, atau *SVM*—yang paling efektif mendeteksi intrusi pada *payload LMS* (kelas: *normal*, *sqli*, *xss*) bila dievaluasi berdasarkan metrik akurasi, *precision*, *recall*, dan *F1-score*; (2) bagaimana strategi pra-proses dan sampling (mis. *TF-IDF* berbasis karakter, *SMOTE-ENN*) dapat mengatasi ketidakseimbangan kelas tanpa menurunkan kemampuan deteksi varian serangan; dan (3) bagaimana desain arsitektur *real-time* yang mengintegrasikan model *ML* ke dalam Moodle (menggunakan Redis, Flask-RQ, dan Grafana) dapat meminimalkan latensi dan beban komputasi sambil mempertahankan keandalan deteksi.

Penelitian ini diharapkan memberikan kontribusi praktis berupa bukti eksperimen atas performa algoritma *ML* untuk deteksi intrusi pada lingkungan *LMS* serta rancangan arsitektur integrasi *real-time* yang dapat diadopsi oleh institusi pendidikan dengan sumber daya TI terbatas.

METODE

Pada Penelitian ini menggunakan pendekatan kuantitatif terapan untuk merancang, mengimplementasi, dan mengevaluasi sistem Deteksi Intrusi (*IDS*) berbasis *supervised machine learning* yang dapat diintegrasikan secara real-time ke *LMS* Moodle. Tujuan eksperimen adalah mengukur dan membandingkan performa model klasifikasi tiga kelas *payload HTTP*—*normal*, *sqli*, dan *xss*—serta mengevaluasi strategi pra-proses dan sampling untuk mengatasi ketidakseimbangan data, dengan fokus pada penerapan yang dapat diterapkan pada arsitektur *LMS* Moodle (Redis + Flask-RQ + Grafana). Data berasal dari gabungan *dataset* publik (HuggingFace: web-attacks/all.csv) dan koleksi *log payload* Moodle internal (serangkaian file CSV). Setelah pembersihan dan harmonisasi kolom, ukuran *dataset* akhir adalah 291.678 baris *payload* dengan distribusi label: *normal* = 136.193, *sqli* = 79.821, *xss* = 75.664. Semua file dibaca menggunakan *Pandas*, kolom distandarkan menjadi *Payload* dan *text_label*, dan nilai kosong pada *Payload* diisi *string* kosong ('') untuk mencegah *error* vektorisasi.

Pra-proses meliputi label *encoding* manual (['*normal*','*xss*','*sqli*'] → [0,1,2]) menggunakan *LabelEncoder*, serta ekstraksi fitur teks dengan *TF-IDF* berbasis karakter menggunakan konfigurasi: *TfidfVectorizer*(*analyzer='char', ngram_range=(2,5), max_features=50000, lowercase=False, min_df=2, sublinear_tf=True*). (Catatan praktik terbaik: *TfidfVectorizer* sebaiknya di-*fit* hanya pada *training set* untuk menghindari kebocoran informasi dari *test set*; laporan ini merekomendasikan praktik tersebut dan eksperimen utama mengikuti rekomendasi ini.) Pembagian data dilakukan secara *stratified train/test split* dengan *test_size=0.2* dan *random_state=42* untuk menjaga proporsi kelas. Ukuran *set* setelah *split* adalah: *training* (80%) = 233.342 sampel (*normal* 108.954; *xss* 63.857; *sqli* 60.531) dan *test* (20%) = 58.336 sampel (*normal* 27.239; *xss* 15.964; *sqli* 15.133). Untuk eksperimen *tuning* tambahan digunakan *Stratified K-Fold Cross-Validation* (contoh: *k=5*) pada *training set*.

Untuk penanganan ketidakseimbangan dieksplorasi teknik *sampling* gabungan *SMOTE–ENN* (*SMOTE* untuk *oversampling* minoritas diikuti *ENN* untuk membersihkan sampel *noisy*). Ketika *SMOTE–ENN* diterapkan, prosedur hanya dilakukan pada *training set*; *test set* tetap dibiarkan asli agar evaluasi mencerminkan kondisi nyata. Laporan menyajikan hasil *baseline* tanpa *oversampling* (karena distribusi relatif mendekati seimbang) dan hasil eksperimen dengan *SMOTE–ENN* saat melakukan optimasi model. Model yang dilatih dan dibandingkan adalah: *Random Forest* (*n_estimators=1500, n_jobs=-1, random_state=42*), *Decision Tree* (*random_state=42*), dan *Support Vector Machine* dengan *kernel* linear (*SVC*(*kernel='linear', probability=True*) atau *LinearSVC* pada skenario skala besar). Model dilatih pada *X_train*, *y_train* dan dievaluasi pada *X_test*, *y_test*; model final disimpan menggunakan *joblib.dump()* untuk keperluan *deployment*.

Metode validasi utama adalah evaluasi pada *held-out test set* (20%) yang distandarisasi melalui *stratified split*. Untuk penyetelan *hyperparameter* digunakan *Stratified K-Fold Cross-Validation* pada *training set* (mis. *k*=5) agar pemilihan *hyperparameter* tidak "melihat" *test set*. Semua eksperimen dan *split* menggunakan *random_state=42* untuk reproduksibilitas. Metrik evaluasi yang dilaporkan meliputi *confusion matrix* (per kelas), *accuracy*, *precision*, *recall*, *F1-score*, dan *AUC (ROC)* dengan pendekatan *one-vs-rest*. Perhitungan metrik diberikan secara eksplisit untuk transparansi: *accuracy* dihitung sebagai $Accuracy = \frac{\text{jumlah prediksi benar}}{N} = \frac{TP+TN}{TP+TN+FP+FN}$; *precision* per kelas $Precision = \frac{TP}{TP+FP}$; *recall* per kelas $Recall = \frac{TP}{TP+FN}$; dan *F1* per kelas $F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$. *AUC ROC* dihitung per kelas dengan skema *one-vs-rest* menggunakan skor probabilitas (*predict_proba*) atau *decision_function* (untuk *SVM* jika *predict_proba* tidak tersedia); untuk ringkasan dilaporkan rata-rata *macro-AUC* dan apabila relevan *micro-AUC*. Selain metrik per-kelas, disajikan juga nilai *macro avg* dan *weighted avg* untuk *precision*/*recall*/*F1* agar pembaca dapat melihat pengaruh distribusi kelas.

Prosedur evaluasi praktis yang diikuti adalah: (1) pra-proses data dan ekstraksi fitur *TF-IDF* (*fit* pada *training*, *transform* pada *test*), (2) *stratified train*/*test split* (80/20), (3) (opsional) terapkan *SMOTE–ENN* pada training set bila menguji penyeimbangan, (4) latih model *RF*/*DT*/*SVM* pada *training set*, (5) prediksi pada *test set* dan hitung *confusion matrix*, *accuracy*, *precision*, *recall*, *F1* per kelas serta *ROC*/*AUC (one-vs-rest)*, (6) visualisasi hasil (*confusion matrix heatmap*, *grouped bar* metrik per-kelas, *ROC curves*, *top TF-IDF features*), dan (7) simpan model dan *pipeline preprocessing* untuk *deployment*. Semua transformasi yang dipelajari (*TF-IDF*, teknik *sampling*) disimpan dalam *pipeline* (mis. *sklearn.pipeline.Pipeline*) bersama model agar tidak terjadi kebocoran data saat evaluasi atau saat *deployment*.

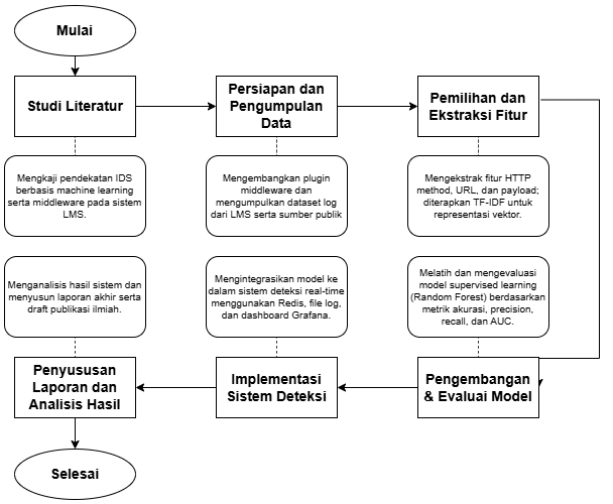
Untuk reproduksibilitas dan pelaporan: semua eksperimen menjalankan *random_state=42* dan konfigurasi lengkap (versi library, parameter *TF-IDF*, *hyperparameter* model, *seed*, ukuran *dataset*) dicatat. Disarankan untuk menjalankan *Stratified K-Fold CV* (k≥5) pada *training set* untuk mengestimasi varians performa sebelum membuat klaim konklusif. Hasil akhir dilaporkan dengan metrik per-kelas, *confusion matrix*, kurva *ROC*/*AUC*, dan interpretabilitas fitur (*top n-grams TF-IDF* via *feature importance* atau *permutation importance* pada *Random Forest*) untuk memberikan gambaran lengkap tentang kemampuan deteksi dan relevansi fitur bagi praktik integrasi *IDS* ke *LMS*.

Jenis Penelitian

Penelitian ini merupakan penelitian kuantitatif terapan yang berfokus pada pengembangan dan pengujian sistem deteksi intrusi pada aplikasi web berbasis Moodle. Penelitian dilakukan secara eksperimental dengan pendekatan klasifikasi terawasi (*supervised classification*) menggunakan data berlabel dari *log HTTP* yang mencakup aktivitas normal dan serangan seperti *SQL Injection* serta *Cross-Site Scripting* (XSS). Pendekatan ini dipilih karena mampu menangkap pola serangan berbasis teks dan parameter *HTTP*, serta memberikan akurasi tinggi dalam proses deteksi intrusi.

Tahapan Penelitian

Penelitian dilakukan melalui enam tahapan utama: Studi Literatur, Persiapan dan Pengumpulan Data, Pemilihan dan Ekstraksi Fitur, Pengembangan dan Evaluasi Model, Implementasi Sistem Deteksi, Penyusunan Laporan sebagaimana digambarkan pada Gambar 1:



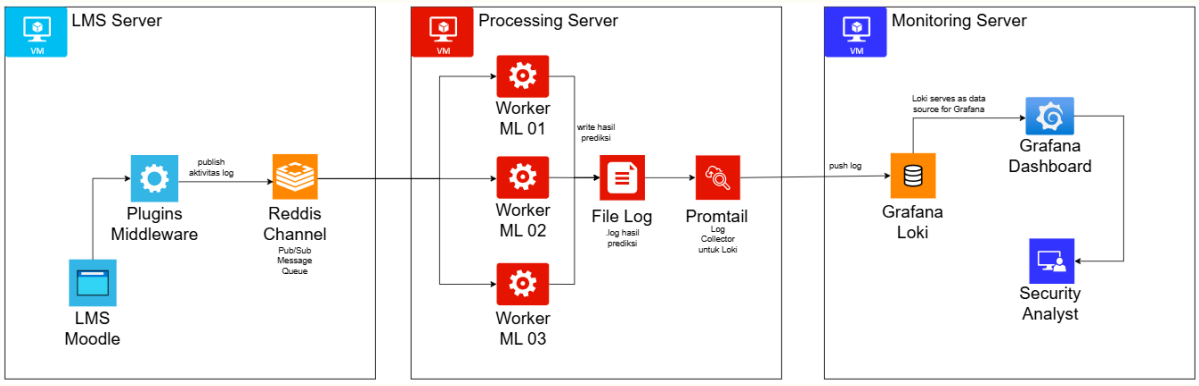
Gambar 1. Tahapan Penelitian

Gambar 1. menunjukkan alur tahapan penelitian yang dilaksanakan selama enam bulan, dimulai dari tahap studi literatur hingga penyusunan laporan. Penelitian dimulai dengan pengkajian literatur untuk memahami pendekatan deteksi intrusi berbasis *machine learning* serta konsep *middleware* pada sistem *LMS* Moodle. Tahap berikutnya adalah pengembangan plugin middleware pada Moodle dan pengumpulan *dataset HTTP log*, baik dari lingkungan *LMS* aktual maupun dataset publik sebagai pembanding. Setelah data terkumpul, dilakukan pemilihan dan ekstraksi fitur penting seperti *HTTP method*, *URL*, dan *payload*. *Payload* kemudian diproses menggunakan teknik *Term Frequency–Inverse Document Frequency (TF-IDF)* untuk menghasilkan representasi numerik yang dapat digunakan dalam pelatihan model. Tahap selanjutnya adalah pengembangan dan evaluasi model klasifikasi menggunakan algoritma *Random Forest*, dengan pengujian performa melalui metrik akurasi, *precision*, *recall*, dan *AUC*.

Model terbaik kemudian diintegrasikan dalam sistem deteksi intrusi *real-time* menggunakan arsitektur Redis untuk pengiriman *log*, serta Grafana sebagai media monitoring. Penelitian diakhiri dengan analisis hasil dan penyusunan laporan akhir serta *draft* publikasi ilmiah sebagai bentuk diseminasi hasil riset.

Rancangan Sistem

Arsitektur sistem deteksi intrusi dirancang dalam tiga *server* utama: *LMS Server*, *Processing Server*, dan *Monitoring Server*, seperti yang ditunjukkan pada Gambar 2:



Gambar 2. Arsitektur Sistem Deteksi Intrusi *LMS* Moodle

Gambar 2 menunjukkan arsitektur sistem deteksi intrusi yang terdiri dari tiga lingkungan utama: *LMS Server*, *Processing Server*, dan *Monitoring Server*. *LMS Server* menjalankan Moodle dengan *plugin middleware* yang mengirim *log* aktivitas pengguna ke *Redis Channel*. *Log* kemudian diproses oleh *worker machine learning* pada *Processing Server*, dan hasil prediksi dicatat dalam file *log*. Selanjutnya, *Monitoring*

Server menggunakan Promtail dan Grafana Loki untuk mengelola serta menampilkan *log* prediksi melalui *dashboard* Grafana bagi analis keamanan.

Alur Pengumpulan Data

Proses pengumpulan data dimulai dengan pemanfaatan *dataset* publik *web-attacks* dari repositori Hugging Face (file all.csv) sebagai data pelatihan awal. *Dataset* ini terdiri atas *request HTTP* yang telah dilabeli berdasarkan jenis serangan dan aktivitas normal, sehingga dapat digunakan untuk membentuk model dasar klasifikasi intrusi menggunakan pendekatan *supervised learning*. Setelah proses pelatihan awal, sistem dikembangkan lebih lanjut dengan melakukan pengumpulan data secara langsung dari lingkungan operasional *Learning Management System (LMS)* Moodle STT Terpadu Nurul Fikri. Untuk mendukung hal ini, dirancang dan diimplementasikan sebuah *plugin middleware* khusus yang berfungsi menangkap *log HTTP* setiap interaksi pengguna. Informasi yang terekam mencakup metode *request*, *URL*, status kode, parameter *query*, dan *payload body*, yang kemudian diteruskan ke *channel* Redis menggunakan skema *publish-subscribe*.

Log aktivitas yang dikumpulkan selanjutnya diproses dan diekstraksi fitur-fiturnya menggunakan pendekatan berbasis teks, di antaranya representasi *Term Frequency-Inverse Document Frequency (TF-IDF)* pada bagian *payload*. Label untuk data *log* ini dihasilkan dengan bantuan *AI-assisted annotation tools* berbasis model klasifikasi awal, disertai validasi manual untuk menjamin kualitas *dataset* hasil tangkapan dari sistem nyata. Pendekatan kombinatif ini memungkinkan model yang dikembangkan tidak hanya memahami pola umum dari data terstruktur yang tersedia secara publik, tetapi juga dapat beradaptasi dengan dinamika aktivitas riil pengguna pada sistem *LMS* institusi.

Analisis Data

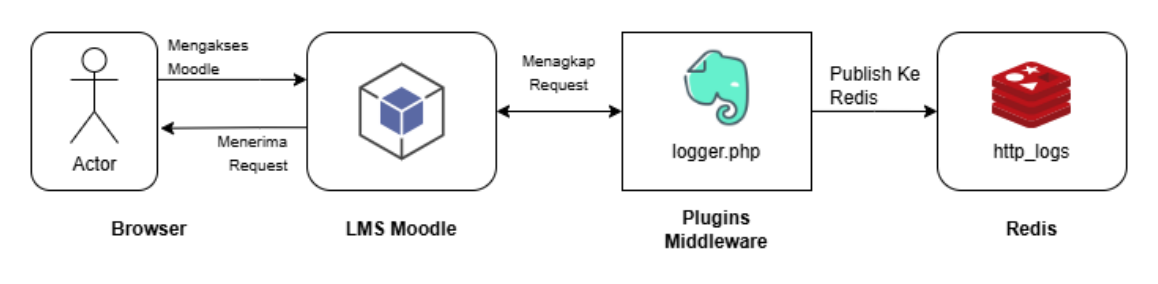
Analisis data dilakukan dengan mengevaluasi hasil prediksi model klasifikasi terhadap data uji menggunakan metrik akurasi, presisi, *recall*, dan *AUC*. Analisis ini bertujuan untuk mengukur performa model dalam membedakan aktivitas normal dan serangan *SQL Injection* maupun *XSS* berdasarkan representasi numerik *payload* yang dihasilkan oleh *TF-IDF*. Evaluasi dilakukan pada data uji terpisah untuk menjaga objektivitas hasil pengujian.

Evaluasi Sistem

Evaluasi dilakukan untuk menguji keandalan dan efektivitas sistem deteksi intrusi yang dibangun. Evaluasi terbagi menjadi dua aspek, yaitu evaluasi fungsional dan evaluasi akurasi. Evaluasi fungsional bertujuan untuk memastikan bahwa seluruh komponen sistem, mulai dari *plugin middleware* hingga proses klasifikasi dan visualisasi *log*, berjalan secara otomatis tanpa gangguan dan intervensi manual. Pengamatan dilakukan terhadap kestabilan plugin dalam menangkap *HTTP request*, kecepatan pengiriman data ke Redis *Channel*, serta konsistensi visualisasi prediksi pada *dashboard* Grafana. Sementara itu, evaluasi akurasi dilakukan untuk mengukur kemampuan model dalam membedakan antara aktivitas normal dan intrusi berdasarkan *log HTTP*. Model dievaluasi menggunakan data uji dan dinilai menggunakan metrik akurasi, *precision*, *recall*, dan *AUC*. Evaluasi ini bertujuan untuk menilai sejauh mana sistem dapat diandalkan sebagai alat bantu dalam pemantauan dan deteksi dini terhadap aktivitas mencurigakan di lingkungan *LMS*.

HASIL DAN PEMBAHASAN

Penelitian ini menghasilkan sistem deteksi intrusi berbasis *supervised machine learning* yang terintegrasi dengan *LMS* Moodle. Untuk memungkinkan sistem mendeteksi aktivitas mencurigakan secara *real-time*, dikembangkan sebuah *plugin middleware* khusus yang diintegrasikan langsung ke dalam *LMS* Moodle yang terinstal secara lokal. *Middleware* ini bertugas menangkap seluruh *HTTP request* dari aktivitas pengguna pada browser, termasuk *method*, *URL*, *payload*, dan *timestamp*, kemudian meneruskannya dalam format JSON ke *channel Redis* melalui mekanisme *publish-subscribe*. Proses ini ditunjukkan pada Gambar 3.



Gambar 3. Alur *Middleware Plugin* dalam Mendistribusikan *HTTP Log* ke Redis

Cuplikan *log* aktivitas yang berhasil ditangkap oleh *middleware* ditampilkan dalam Tabel 1. *Log* ini merepresentasikan struktur data *JSON* yang memuat informasi penting seperti alamat IP pengguna, *endpoint* yang diakses, metode *request*, dan konten *payload*. Data tersebut menjadi fondasi utama dalam membangun dataset aktual untuk pelatihan dan evaluasi model deteksi intrusi.

Tabel 1. Contoh Cup likan *HTTP Log* dari *Plugin Middleware*

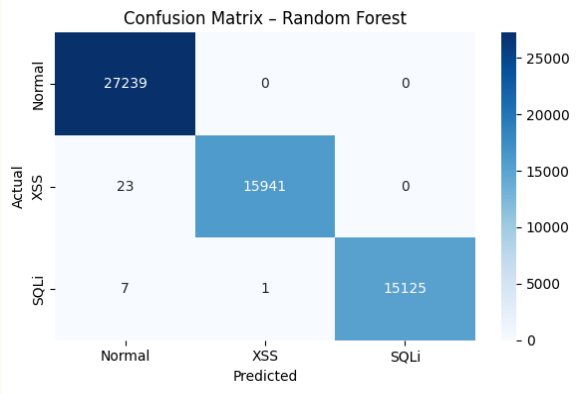
Timestamp	Ip Adress	Method	URL	Payload(Ringkas)
1750745404	127.0.0.1	POST	/moodle/lib/ajax/service.php?...	SELECT ... FROM Orders ... JOIN Shippers ON ...
1750744913	127.0.0.1	GET	/moodle/admin/search.php	[]
1750744563	127.0.0.1	POST	/moodle/course/edit.php?category=1	...fullname=Keamanan+Jaringan...
175074504	127.0.0.1	POST	/moodle/lib/ajax/service.php?...core_calendar_get_action_events...	methodname=core_calendar_get_action_events_by_timesort

Evaluasi kinerja dilakukan pada tiga algoritma, yaitu *Random Forest*, *Decision Tree*, dan *Support Vector Machine (SVM)*, menggunakan *dataset* gabungan (70.336 sampel) yang terdiri atas tiga kelas: Normal, *SQL Injection (SQLi)*, dan *Cross-Site Scripting (XSS)*. Hasil pengukuran akurasi, presisi, *recall*, dan *AUC* ditampilkan pada Tabel 2.

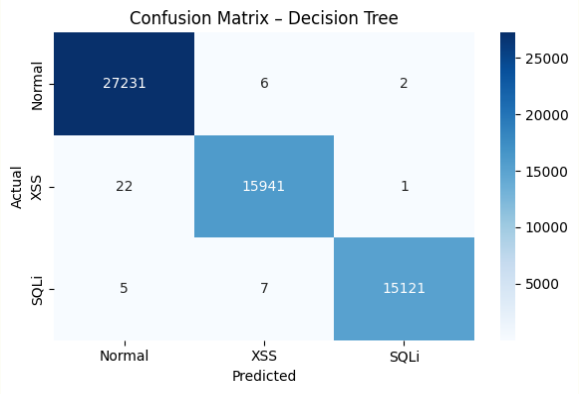
Tabel 2. Hasil Perbandingan Skenario Pengujian

	Skenario 1	Skenario 2	Skenario 3
Algoritma	Random Forest	Decision Tree	SVM
Akurasi %	99,94	99,92	99,94
Precision %	99,94	99,92	99,94
Recall %	99,94	99,92	99,94
AUC	0,9994	0,9992	0,9994

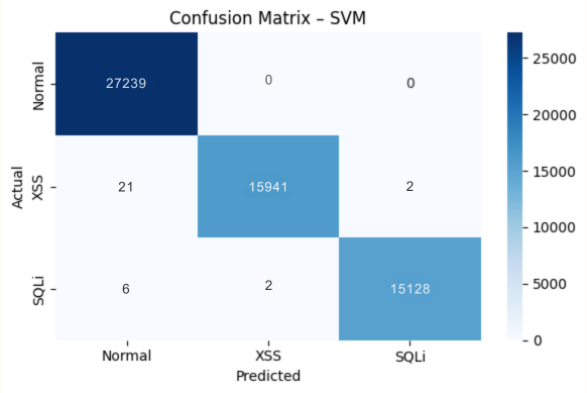
Hasil pada Tabel 2 menunjukkan bahwa *Random Forest* memberikan performa paling stabil, dengan akurasi 99,94% dan *AUC* 0,9994. *Decision Tree* sedikit tertinggal dengan akurasi 99,96%, sedangkan *SVM* setara akurasi namun cenderung lebih lambat dalam inferensi. Keunggulan *Random Forest* selaras dengan hasil studi Garg & Sharma (2024) dan *Nature Scientific Reports* (2025) yang menegaskan bahwa algoritma *ensemble* lebih unggul dalam deteksi intrusi karena ketahanan terhadap *overfitting* dan *noise*.



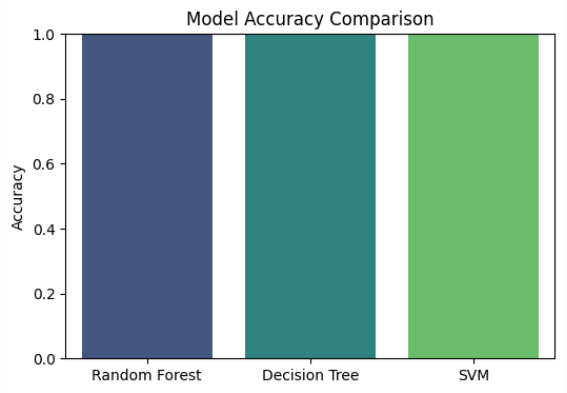
Gambar 4. *Confusion Matrix* Algoritma *Random Forest*



Gambar 5. *Confusion Matrix* Algoritma *Decision Tree*

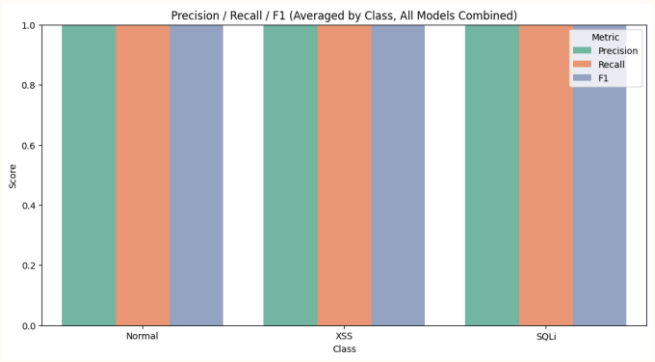


Gambar 6. *Confusion Matrix* Algoritma *SVM*



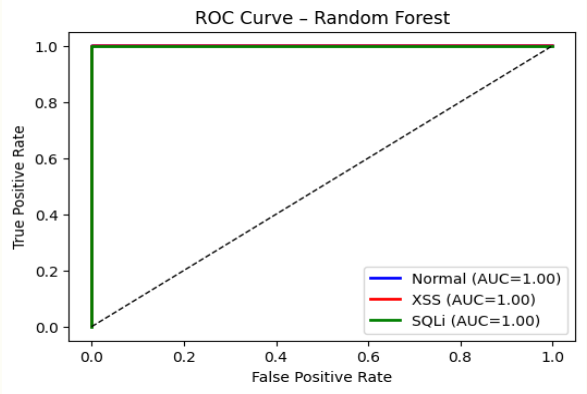
Gambar 7. Perbandingan akurasi antara *Random Forest*, *Decision Tree*, dan *SVM* pada *dataset* uji

Dari Gambar 4. *confusion matrix* tersebut, Gambar 5. *Random Forest* dan *SVM* tidak menghasilkan kesalahan klasifikasi signifikan, sedangkan Gambar 6. *Decision Tree* menunjukkan kesalahan prediksi kecil. Gambar 7. menunjukkan Grafik *Random Forest* dan *SVM* mencapai akurasi ~99.94%, sedangkan *Decision Tree* sedikit lebih rendah (~99.92%). Ini menunjukkan representasi *TF-IDF* berbasis karakter sangat efektif membedakan *payload normal vs XSS vs SQLi*.

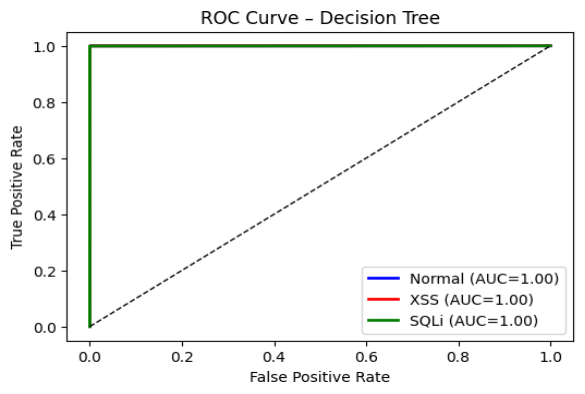


Gambar 8. *Precision, Recall,* dan *F1-score* per kelas (*Normal, XSS, SQLi*) di semua model.

Gambar 8. Menunjukkan nilai metrik mendekati 1.0 untuk tiap kelas, menandakan model memiliki *trade-off* minimal antara *false positives* dan *false negatives* pada data uji.

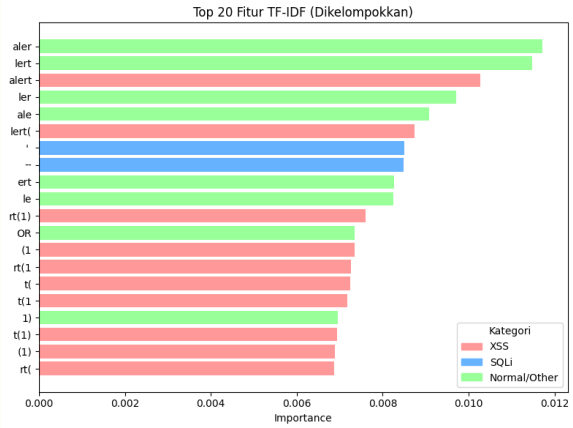


Gambar 9. ROC (one-vs-rest) untuk Random Forest; AUC per kelas



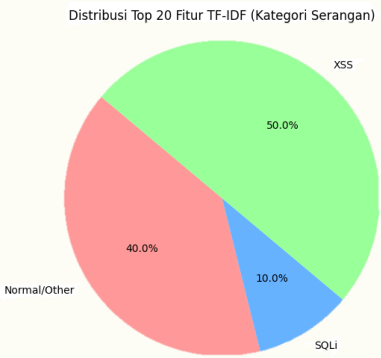
Gambar 10. *ROC* untuk *Decision Tree*; *AUC* per kelas

Gambar 9. Menunjukkan *AUC* sangat tinggi (≈1.00) untuk tiap kelas — *RF* mampu membedakan kelas dengan sangat baik. Gambar 10. Menunjukkan Meski *AUC* juga tinggi, *Decision Tree* tunggal biasanya lebih rentan terhadap *overfitting* dibanding *ensemble* seperti *RF*.



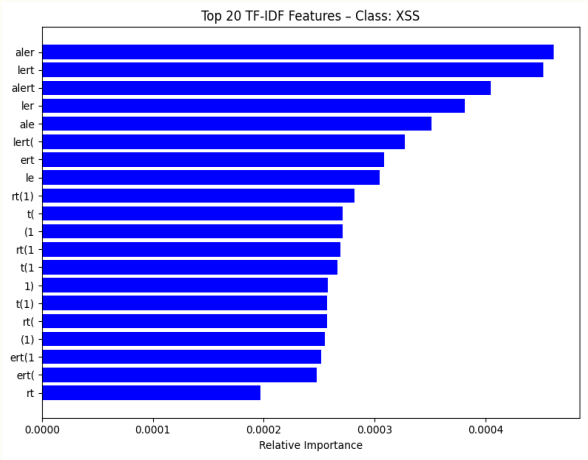
Gambar 11. Top-20 fitur *TF-IDF* yang paling berkontribusi pada prediksi (*XSS* / *SQLi* / *Normal*)

Fitur-fitur seperti *alert*, *script*, *ert(* mengindikasikan *XSS*; token *'*, *--*, *or* mengindikasikan *SQLi*; sedangkan *n-gram* biasa (mis. *al*, *in*) menandakan payload *normal*.

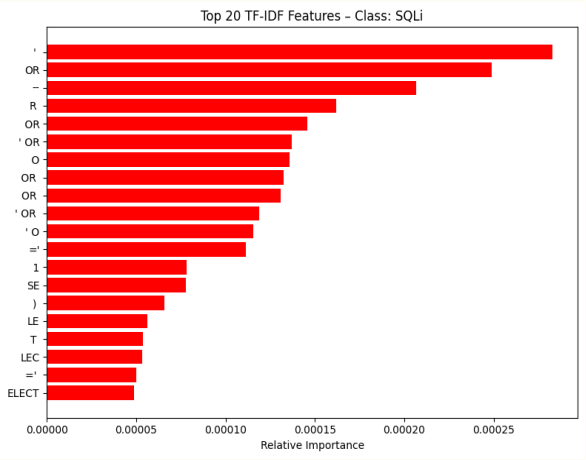


Gambar 12. Proporsi kategori fitur (*XSS* / *SQLi* / *Normal)* di antara 20 fitur teratas

Memudahkan melihat apakah fitur penting lebih banyak berasal dari pola serangan atau pola normal—berguna untuk analisis kebijakan mitigasi.

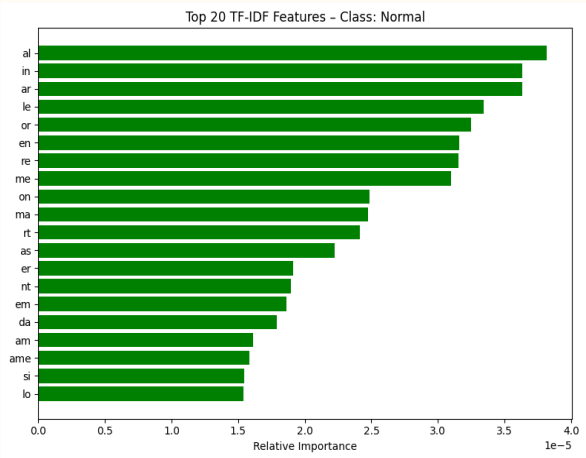


Gambar 13. Fitur paling berpengaruh khusus untuk kelas *XSS*



Gambar 14. Fitur paling berpengaruh untuk kelas *SQL Injection*

Gambar 13. Menunjukkan Token *alert*, *script*, dan *substring* serupa sering muncul pada *payload XSS* (mis. *<script>alert(1)</script>*), sehingga menjadi indikator utama. Gambar 14. Menunjukka Token *', --, select, or* sering menandai *payload* yang mengandung pernyataan *SQL* berbahaya (*' OR 1=1 --*).



Gambar 15. Fitur paling berpengaruh untuk *payload normal*

N-gram umum membantu model mengenali pola wajar dan berfungsi sebagai kontras terhadap pola serangan.

Evaluasi dilakukan pada tiga algoritma: *Random Forest (RF)*, *Decision Tree (DT)*, dan *Support Vector Machine (SVM)*. Pengujian memakai *dataset* gabungan (70.336 sampel) berlabel tiga kelas: *Normal, SQL Injection (SQLi)*, dan *Cross-Site Scripting (XSS)*. Ringkasan metrik (akurasi, presisi, *recall*, *AUC*) disajikan pada Tabel 2. Hasil utama:

1. *Random Forest*: Akurasi 99,94%, *Precision* 99,94%, *Recall* 99,94%, *AUC* 0,9994.
2. *Decision Tree*: Akurasi 99,92%, performa sedikit lebih rendah dibanding *RF*.
3. *SVM*: Akurasi 99,94% (setara *RF*), namun membutuhkan waktu inferensi lebih lama pada konfigurasi saat pengujian.

Dari ketiga *confusion matrix* (Gambar 3–5), *RF* dan *SVM* menunjukkan hampir tidak ada kesalahan klasifikasi signifikan; *Decision Tree* menunjukkan sedikit lebih banyak kesalahan prediksi, konsisten dengan sifat model pohon tunggal yang lebih rentan *overfitting* pada pola *noise*. *Random Forest* unggul karena beberapa alasan inheren arsitektur *ensemble*-nya:

1. *Averaging* dari banyak pohon: dengan menggabungkan prediksi ratusan hingga ribuan pohon, *RF* mengurangi varians model dibandingkan pohon tunggal sehingga lebih tahan terhadap *overfitting*.
2. *Bagging* dan *feature subsetting*: setiap pohon dilatih pada *subset* data dan *subset* fitur yang berbeda; ini membuat *RF* lebih *robust* terhadap *noise* dan *outlier* pada fitur *TF-IDF* (banyak fitur berdimensi tinggi).
3. Kemampuan menangani fitur berdimensi tinggi: *RF* bekerja baik pada representasi *sparse TF-IDF* tanpa memerlukan banyak *tuning kernel* seperti pada *SVM*.
4. Interpretabilitas relatif: *RF* menyediakan metrik kepentingan fitur (*feature importance*) yang membantu mengidentifikasi n-gram karakter yang menjadi indikator *XSS*/*SQLi*—berguna untuk analisis keamanan dan aturan mitigasi.

Singkatnya, sifat *ensemble RF* (reduksi varians + resistensi terhadap *noise* + informasi kepentingan fitur) membuatnya stabil dan andal dalam skenario deteksi intrusi berbasis *payload*. Pengujian *end-to-end* menunjukkan sistem mampu memproses *request* secara *real-time*: *middleware* menerbitkan *log* ke Redis, model melakukan inferensi, dan hasil dicatat ke *log*/*monitoring*. Contoh *log* hasil klasifikasi disajikan di Tabel 3. *Dashboard* berbasis Grafana + Loki menampilkan ringkasan aktivitas (Gambar 16), termasuk jumlah aktivitas normal dan serangan serta distribusi berdasarkan jenis dan alamat IP. Selama periode pengujian 12 hari tercatat 506 aktivitas *normal*, 52 *XSS*, dan 22 *SQLi*—diagram pai mempermudah pemantauan proporsi serangan.

Walau hasil sangat baik pada *dataset* pengujian, ada beberapa keterbatasan penting yang perlu dicatat:

1. Ketergantungan pada *dataset* berlabel:
   a. Model *supervised* memerlukan data berlabel berkualitas. Jika labelisasi keliru atau *dataset* tidak merepresentasikan variasi serangan nyata, performa di lapangan bisa menurun.
   b. Perlu proses *continuous labeling* / *human-in-the-loop* untuk menangani varian baru.
2. *Generalizability* terhadap varian serangan baru / *obfuscation*: Penyerang dapat mengubah *payload* (*encoding*, *obfuscation*, fragmentasi) sehingga n-gram yang pernah penting menjadi kurang relevan. Model perlu *retraining* berkala dan augmentasi data untuk menanggulangi ini.
3. Performa saat lalu lintas tinggi (skala & latensi)
   a. Inferensi model (terutama *RF* dengan banyak estimator atau *SVM non-linear*) dapat menjadi *bottleneck* pada lalu lintas web sangat tinggi. Solusi produksi memerlukan optimasi: model *distillation*, *pruning*, *quantization*, atau arsitektur asinkron (*batching*, *worker pool*).
   b. Selain itu, *overhead middleware* dan *pipeline* Redis harus diukur untuk memastikan latensi deteksi yang dapat diterima.
4. Kemungkinan *False Positive* / *False Negative*: Metrik *test* tinggi tidak menjamin nol *false positive* pada lingkungan produksi yang heterogen. *False positive* dapat membebani tim operasi, sedangkan *false negative* berisiko keamanan.
5. *Adversarial Evasion*: Teknik *adversarial* (mis. perturbasi *payload*) dapat mengecoh model. Perlu penelitian lebih lanjut pada *robustness* dan *defensive training*.
6. Privasi & Kepatuhan: Pengumpulan *payload HTTP* mengandung data sensitif (*credential*). Sistem harus menerapkan *redaction*, enkripsi, dan kebijakan penyimpanan yang sesuai dengan regulasi.

Perbandingan singkat dengan *IDS* konvensional (Snort, Suricata)

1. Pendekatan: Snort/Suricata adalah *IDS* berbasis *signature*/*rule*; model *ML* adalah pendekatan berbasis pola (*generalization*).
2. Kelebihan Snort/Suricata:
   a. Deterministik dan *explainable* (*rule-based*), mudah dipahami operator; rendah *false positives* untuk serangan yang sesuai *rule*; efisien untuk *throughput* tinggi.
   b. Sudah mapan dalam ekosistem dan terintegrasi dengan banyak alat *SOC*.

3. Kelemahan Snort/Suricata: Rentan terhadap varian baru yang tidak ada pada *rules*; membutuhkan *update rule* terus-menerus; sulit mendeteksi serangan *zero-day* atau varian *obfuscated*.
4. Kelebihan pendekatan *ML* (penelitian ini): Mampu mendeteksi varian baru dan pola *non-trivial* yang tidak mudah diwakili satu *rule*; adaptif bila diberi data baru; dapat mengotomatisasi klasifikasi jenis serangan.
5. Kekurangan *ML* dibanding *signature IDS*: Memerlukan data berlabel dan pemeliharaan model; rawan potensi *false positives*; interpretabilitas lebih rendah (meskipun *RF* masih dapat memberi *importance features*).
6. Rekomendasi praktis: gabungkan keduanya—*hybrid IDS*—di mana *signature IDS* (Snort/Suricata) menangani deteksi deterministik dan *rule-based blocking* sementara model *ML* memonitor anomali/*pattern* baru dan memberi prioritas investigasi. *Hybrid* ini memanfaatkan kekuatan masing-masing pendekatan dan mengurangi kelemahan tunggal.

Implikasi, Rekomendasi & Arah Penelitian Lanjutan

1. Produksi & optimasi: sebelum *deployment* skala besar, lakukan pengujian beban (*load testing*), optimasi inferensi (model *compression* / *distillation*), dan arsitektur asinkron (*queueing, worker autoscaling*).
2. *Pipeline* pembaruan model: sediakan mekanisme *retraining* terjadwal dan *human-in-the-loop* untuk label baru. Sertakan metrik *drift detection* agar terdeteksi saat distribusi *payload* berubah.
3. *Hardening* terhadap *adversarial*: terapkan *adversarial training* dan *sanitization/normalization payload* sebelum vektorisasi.
4. Privasi: redaksi otomatis nilai sensitif (mis. sesi, password) sebelum penyimpanan.
5. Integrasi *SOC*: sambungkan *alert* ke *workflow* tiket/*incident response* sehingga memangkas *false positives* dan respon cepat bisa dioperasikan.

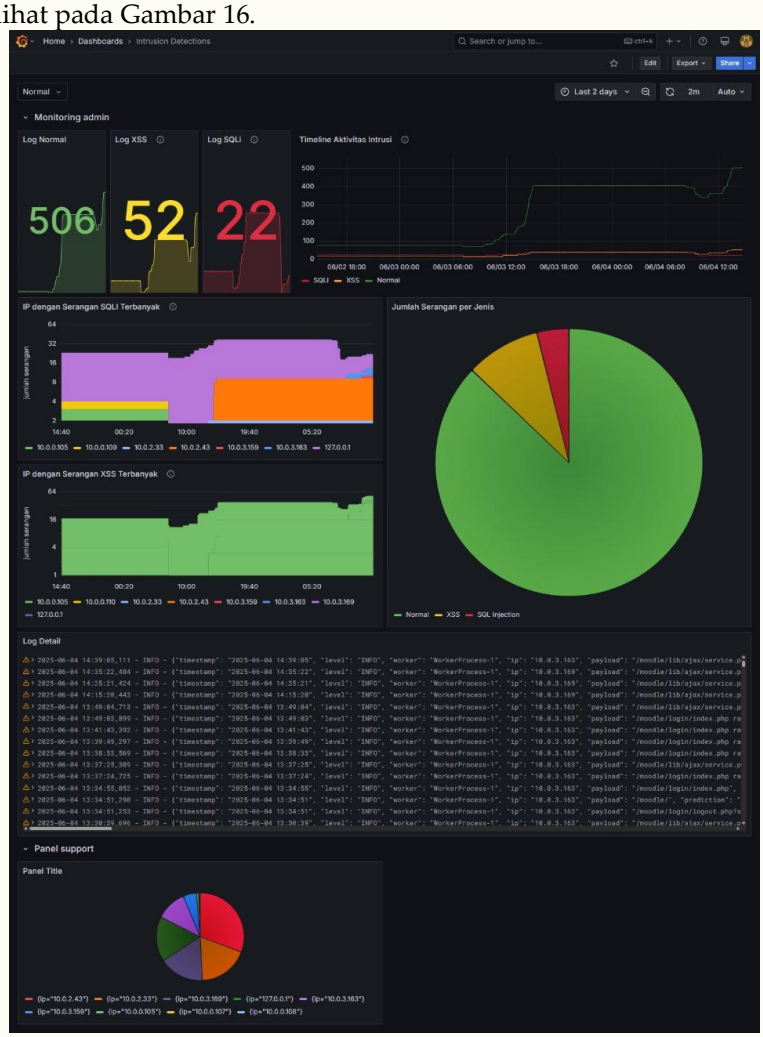
Sistem deteksi intrusi berbasis *supervised ML* yang diusulkan (dengan *middleware* Redis dan integrasi Grafana-Loki) menunjukkan akurasi dan kemampuan *real-time* yang tinggi pada *dataset* pengujian. *Random Forest* muncul sebagai solusi paling stabil berkat sifat *ensemble*-nya, sementara *SVM* mencapai performa serupa namun dengan biaya inferensi lebih tinggi. Untuk implementasi produksi yang andal, direkomendasikan pendekatan *hybrid* (*ML + signature IDS*), *pipeline retraining*, optimasi inferensi, dan perhatian terhadap privasi serta *robustness* terhadap teknik evasi. Selain evaluasi model, pengujian sistem secara *end-to-end* menunjukkan kinerja deteksi *real-time* yang andal. Cuplikan *log* pada Tabel 3 menunjukkan sistem mampu memproses *HTTP request*, mengklasifikasikan aktivitas, dan mencatat prediksi secara otomatis.

Tabel 3. Contoh *Log* Hasil Deteksi Intrusi

Timestamp	IP	HTTP Payload (Ringkas)	Prediksi
2025-07-16 09:30:34	127.0.0.1	POST /lib/ajax/service.php?sesskey=*****&info=core_message_get_unsent_message	Normal
2025-07-16 09:30:36	127.0.0.1	GET /lib/ajax/service-nologin.php?info=core_output_load_template_with_dependencies&cachekey=*****	Normal
2025-07-16 09:30:37	127.0.0.1	GET /lib/ajax/service-nologin.php?info=core_output_load_template_with_dependencies,...	Normal
2025-07-14 14:30:02	127.0.0.1	POST /moodle/lib/ajax/service.php?info=mod_assign_submit_gra	XSS

		ding_form&jsonformdata=...<iframe src="javascript:alert(1)"></iframe>...	
2025-07-14 14:25:18	127.0.0.1	POST /moodle/lib/ajax/service.php?info=core_calendar_submit_create_update_form&formdata=...SELECT COUNT(id), country FROM mdl_user...	SQL Injection

Hasil pengujian membuktikan bahwa sistem deteksi intrusi dapat bekerja akurat, efisien, dan *real-time*, sekaligus memberikan transparansi hasil melalui *dashboard* monitoring (Grafana + Loki) dan *log tracking* yang mudah dianalisis. Selain hasil pengujian model dan *confusion matrix*, sistem juga dilengkapi dengan dashboard monitoring berbasis Grafana yang terintegrasi dengan Loki dan Promtail. *Dashboard* ini menampilkan ringkasan *log* deteksi intrusi, jumlah aktivitas normal dan serangan (*XSS* dan *SQL Injection*), serta distribusi serangan berdasarkan jenis dan alamat IP. Tampilan *dashboard* dapat dilihat pada Gambar 16.



Gambar 16. *Dashboard Monitoring* Intrusi Menggunakan Grafana-Loki

Dashboard ini memungkinkan administrator untuk memantau aktivitas intrusi secara *real-time*, termasuk tren aktivitas (*timeline*), distribusi serangan, IP sumber serangan terbanyak, dan rincian *log* yang dapat ditelusuri lebih lanjut. Dari data yang ditampilkan pada periode pengujian (12 hari terakhir), sistem mencatat 506 aktivitas *normal*, 52 serangan *XSS*, dan 22 serangan *SQL Injection*. Distribusi ini divisualisasikan dalam grafik pai yang mempermudah pemantauan proporsi serangan dibanding aktivitas normal.

Dengan adanya integrasi *dashboard* ini, sistem tidak hanya mampu mendeteksi ancaman secara akurat, tetapi juga memberikan transparansi dan kemudahan analisis sehingga dapat mendukung respon cepat oleh *administrator* keamanan jaringan.

KESIMPULAN

Penelitian ini menyimpulkan bahwa sistem deteksi intrusi berbasis *supervised machine learning* berhasil dirancang dan diimplementasikan secara efektif pada platform *Learning Management System* (LMS) Moodle untuk mendeteksi serangan *SQL Injection (SQLi)* dan *Cross-Site Scripting (XSS)* secara *real-time*. Dengan memanfaatkan algoritma *Random Forest*, representasi fitur *payload* berbasis *TF-IDF*, serta arsitektur terdistribusi yang melibatkan *plugin middleware*, Redis, *worker* paralel, dan *dashboard* Grafana, sistem ini mampu menganalisis *log HTTP* pengguna secara otomatis dan menghasilkan deteksi intrusi dengan akurasi, presisi, *recall*, dan *AUC* yang sangat tinggi tanpa mengganggu kinerja *LMS*.

Hasil pengujian menunjukkan bahwa sistem mampu memproses data *log* secara cepat dan stabil, sehingga efektif mendukung pemantauan keamanan aplikasi web berbasis pendidikan. Implementasi sistem ini memberikan kontribusi praktis yang signifikan terhadap peningkatan keamanan siber di lingkungan lembaga pendidikan, karena mampu:

1. Menyediakan deteksi ancaman secara langsung pada aktivitas pengguna *LMS* tanpa perlu perangkat tambahan eksternal.
2. Mengurangi risiko eksploitasi data dan kerusakan sistem akibat serangan injeksi atau skrip berbahaya.
3. Meningkatkan kesadaran dan kesiapsiagaan administrator terhadap pola serangan yang sering terjadi di sistem pembelajaran daring.

Sistem ini berpotensi dikembangkan lebih lanjut dengan memperluas cakupan deteksi ke serangan lain seperti *Remote Code Execution (RCE)* atau *zero-day exploits*, serta mengintegrasikan pendekatan *unsupervised learning* atau *deep learning* untuk meningkatkan adaptivitas terhadap ancaman baru. Selain itu, sistem ini dapat dijadikan modul keamanan mandiri untuk berbagai platform web, dengan penambahan fitur seperti notifikasi otomatis dan respon aktif, sehingga mampu berkontribusi pada pembangunan ekosistem keamanan siber yang lebih proaktif dan berkelanjutan — khususnya di sektor pendidikan.

DAFTAR PUSTAKA

Ali, A. H., Charfeddine, M., Ammar, B., Hamed, B. B., Albalwy, F., Alqarafi, A., Hussain, A. (2024). Unveiling Machine Learning Strategies And Considerations In Intrusion Detection Systems: A Comprehensive Survey. *Frontiers in Computer Science*, 6. https://doi.org/10.3389/fcomp.2024.1387354

Amirah, A., & Sanmorino, A. (2023). Deteksi Intrusi Siber Pada Sistem Pembelajaran Elektronik Berbasis Machine Learning. *Jurnal Ilmiah Informatika Global*, 14(2), 12–16. https://doi.org/10.36982/jiig.v14i2.3227

Aulianoor, A. B., & Koprawi, M. (2024). Comparative Analysis Of The Performance Of Decision Tree And Random Forest Algorithms In SQL Injection Attack Detection. *JAIC: Journal of Applied Informatics and Computing*, 8(1), 194–202. https://doi.org/10.30871/jaic.v8i1.8136

Fauzi, R. M., Hermawan, R., Adhy, D. R., & Maesaroh, S. (2024). Analisis Kerentanan Keamanan Web Menggunakan Metode OWASP dan PTES di Web Pemerintahan Desa XYZ. *Power Elektronik: Jurnal Orang Elektro*, 13(2), 225–231. https://doi.org/10.30591/polektro.v13i2.6711

Febriyani, F., Pramukantoro, E. S., & Bachtiar, F. A. (2019). Perbandingan Kinerja Redis, Mosquitto, Dan Mongodb Sebagai Message Broker Pada Iot Middleware. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 3(7), 6816–6823.

Garg, A., & Sharma, R. (2024). Machine Learning For Detecting SQL Injection In Web Applications. *International Journal of Cybersecurity*, 12(3), 45–60.

Genuario, F., Santoro, G., Giliberti, M., Bello, S., Zazzera, E., & Impedovo, D. (2024). Machine Learning-Based Methodologies For Cyber-Attacks And Network Traffic Monitoring: A Review And Insights. *Information MDPI*, 15(11), 741. https://doi.org/10.3390/info15110741

Hubballi, N., & Suryanarayanan, V. (2014). False-Alarm Minimization Techniques In Signature-Based Intrusion Detection Systems: A Survey. *Computer Communications*, 49(1), 1-17. https://doi.org/10.1016/j.comcom.2014.04.012

Husna, M. A., & Rosyani, P. (2021). Implementasi Sistem Monitoring Jaringan Dan Server Menggunakan Zabbix Yang Terintegrasi Dengan Grafana Dan Telegram. *JURIKOM: Jurnal Riset Komputer*, 8(6), 247–255. https://doi.org/10.30865/jurikom.v8i6.3631

Prasetyo, S. E., Haeruddin, H., & Ariesryo, K. (2024). Website Security System From Denial Of Service Attacks, SQL Injection, Cross Site Scripting Using Web Application Firewall. *Antivirus: Jurnal Ilmiah Teknologi Informasi*, 18(1), 27–36. https://doi.org/10.35457/antivirus.v18i1.3339

Pinto, A., Herrera, L.-C., Donoso, Y., & Gutierrez, J. A. (2023). Survey On Intrusion Detection Systems Based On Machine Learning Techniques For The Protection Of Critical Infrastructure. *Sensors*, 23(5), 2415. https://doi.org/10.3390/s23052415

Rahayu, A., Yulyanti, E., & Ghalib, M. (2025). Systematic Literature Review: SQL Injection Detection Vulnerability Using Machine Learning. *Jurnal Media Infotama*, 21(1), 15–20. https://doi.org/10.37676/jmi.v21i1.6702

Rattrout, A., Al-Hawawreh, M., & Sitnikova, E. (2023). Machine Learning Advancements In SQL Injection Detection: NLP And Hybrid Models. *Research Square*, 1. https://doi.org/10.21203/rs.3.rs-3446830/v1

Rasyidi, B., & Pratama, F. (2024). Sistem Monitoring Server Di PT. XYZ Media Indonesia Berbasis Grafana Dan Prometheus. *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, 4(4), 1456–1465. https://doi.org/10.57152/malcom.v4i4.1546

Sangadji, V. I., Muhammad, A. H., & Gunawan, E. (2023). Penerapan Metode Signature-Based Berbasis IDS Snort Dan IDS Suricata Pada Keamanan Jaringan Laboratorium Komputer. *J-Tifa: Jurnal Teknik Informatika,* 6(1), 18–22. https://doi.org/10.52046/j-tifa.v6i2.1678

Setiawan, H.; Munandar, M. A.; Astuti, L. W. (2021). Penggunaan Metode Signature Based dalam Pengenalan Pola Serangan di Jaringan Komputer. *JTIIK: Jurnal Teknologi Informasi dan Ilmu Komputer*. https://doi.org/10.25126/jtiik.2021834200

Sugara, V. I., & Sriyasa, I. W. (2024). Analisis keamanan web menggunakan Open Web Application Security Project (OWASP). *Indonesian Journal of Computer Science*, 13(2), 3315–3327. https://doi.org/10.33022/ijcs.v13i2.3736

Tan, T., Sama, H., Wijaya, G., & Aboagye, O. E. (2023). Studi Perbandingan Deteksi Intrusi Jaringan Menggunakan Machine Learning (Metode SVM dan ANN). *Jurnal Teknologi dan Informasi*, 13(2), 152–164. https://doi.org/10.34010/jati.v13i2.10484

Veluvali, P., & Surisetti, J. (2021). Learning Management System For Greater Learner Engagement In Higher Education — A Review. *Higher Education for the Future*. https://doi.org/10.1177/23476311211049855

Wulandari, J. R., Pramukantoro, E. S., & Nurwasito, H. (2018). Implementasi Cluster Message Broker Sebagai Solusi Skalabilitas Middleware Berbasis Arsitektur Publish-Subscribe pada Internet of Things (IoT*). Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 2(12), 6861–6867.