PERANCANGAN SISTEM KEAMANAN WEB MENGGUNAKAN METODE RANDOM FOREST (STUDI KASUS : INSTITUSI X)

Muhammad Firman Prayogi, Ahmad Fahrudi Setiawan, Franciscus Xaverius Ariwibisono
Teknik Informatika, Institut Teknologi Nasional Malang
Jalan Raya Karanglo km 2 Malang, Indonesia
muhammadfirmanprayogi@gmail.com

ABSTRAK

Penelitian ini bertujuan untuk merancang dan mengimplementasikan *SecureShield*, sebuah sistem keamanan web yang mampu mendeteksi dan merespons tiga jenis serangan siber utama: *SQL Injection*, *Cross-site Scripting (XSS)*, dan *Webshell*. Sistem ini dibangun secara modular yang terdiri dari tiga komponen utama, yaitu *Go Agent* untuk pemantauan lalu lintas, *Python Engine* untuk pemrosesan data dan klasifikasi serangan, serta *Dashboard Monitoring* sebagai antarmuka pengguna. Metode yang digunakan adalah algoritma *Random Forest*, dengan pendekatan pembelajaran mesin berbasis analisis konten dan trafik HTTP. Hasil pengujian menunjukkan bahwa sistem mampu mengenali serangan dengan tingkat akurasi di atas 99% dan waktu respon rata-rata 1.3 detik. Sistem juga mampu memblokir serangan secara otomatis serta menyimpan log kejadian untuk keperluan audit. Hasil ini menunjukkan bahwa *SecureShield* dapat diandalkan sebagai solusi keamanan aplikasi web yang adaptif dan efisien.

Kata kunci: *Keamanan Web,* SQL Injection, XSS, Webshell, *Random Forest,* Real-Time Monitoring

1. PENDAHULUAN

Meningkatnya ketergantungan terhadap aplikasi berbasis *web* dalam berbagai sektor menjadikan sistem ini rentan terhadap ancaman siber. Serangan seperti *SQL Injection*, *Cross-site Scripting (XSS)*, dan penyisipan *Webshell*, Serangan ini umumnya dilakukan melalui permintaan HTTP yang tampak normal namun mengandung muatan berbahaya, dan terus berkembang secara dinamis sehingga membutuhkan pendekatan adaptif[1],jenis serangan yang sering terjadi dan dapat membahayakan kerahasiaan, integritas, serta ketersediaan layanan digital. Menurut laporan ID-SIRTII/CC tahun 2023[2], serangan terhadap situs institusi pendidikan menempati persentase signifikan dengan pola yang semakin kompleks.

Sayangnya, banyak sistem keamanan yang masih bergantung pada pendekatan berbasis aturan statis. Untuk itu, penelitian ini merancang sistem deteksi serangan otomatis berbasis algoritma *Random Forest* yang terintegrasi dalam satu platform bernama *SecureShield*. Sistem ini bekerja secara *real-time* dengan memanfaatkan *Go Agent* untuk pemantauan lalu lintas, *Python Engine* untuk analisis klasifikasi serangan, dan *dashboard* untuk visualisasi dan pelaporan.

2. TINJAUAN PUSTAKA

2.1. Keamanan Web dan Serangan Siber

Keamanan aplikasi *web* menjadi perhatian utama dalam pengembangan sistem informasi. Beberapa serangan umum yang mengincar aplikasi web antara lain adalah *SQL Injection*, *XSS*, dan penyisipan *Webshell*[3]. Penanganan yang bersifat reaktif tidak cukup dalam menghadapi pola serangan modern yang bersifat dinamis dan adaptif. Berdasarkan studi Riera dan Higuera[4], pendekatan deteksi anomali berbasis *machine learning* mampu mengurangi risiko ancaman yang tidak teridentifikasi oleh sistem berbasis aturan konvensional. pada penelitian sebelumnya Shahid [5] dalam penelitiannya menekankan bahwa algoritma *machine learning*, khususnya *ensemble classifier*, efektif untuk mendeteksi kerentanan web secara proaktif dan mengurangi kesalahan klasifikasi. Oleh karena itu, pendekatan berbasis pembelajaran mesin dinilai lebih efektif karena mampu mengenali pola berdasarkan histori dan struktur permintaan yang mencurigakan.

Penelitian sebelumnya oleh T. H. Nguyen [6],dan S. Leelavathy [7] telah berhasil mendeteksi serangan menggunakan pendekatan pembelajaran mesin, namun sistem yang mereka kembangkan belum dirancang untuk melakukan deteksi secara *real-time*.

2.2. Algoritma *Random Forest*

Random Forest adalah algoritma ensemble learning berbasis sekumpulan *decision tree* yang bekerja dengan prinsip pemungutan suara mayoritas. Algoritma ini dikembangkan oleh Leo Breiman[8], Algoritma ini terbukti efektif dalam klasifikasi multikategori dan tahan terhadap overfitting seperti pada penelitian X.D. Hoang[9]. Dalam penelitian ini, *Random Forest* digunakan untuk mengklasifikasikan permintaan *HTTP* apakah termasuk serangan atau bukan.

Salah satu keunggulan utama *Random Forest* adalah kemampuannya dalam mengurangi risiko overfitting, yang sering terjadi pada model pohon keputusan tunggal. Dengan menggabungkan banyak pohon dan melakukan pengambilan suara mayoritas (*majority voting*),Secara umum, prediksi pada *Random Forest* untuk kasus klasifikasi dapat dirumuskan sebagai berikut:

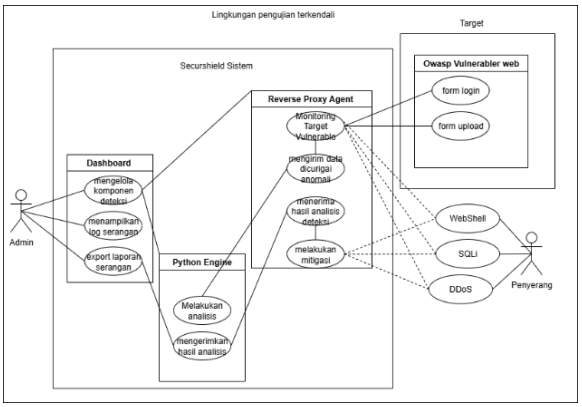
$$\hat{y} = mode\{h1(x), h2(x), \ldots, hk(x)\} \quad (1)$$

Keterangan:
$\hat{y}$ adalah hasil prediksi akhir
$h_i(x)$ merupakan prediksi dari pohon ke - i
k adalah jumlah total pohon dalam random forest
$mode$ pengambilan suara mayoritas

3. METODE PENELITIAN

3.1. Arsitektur Sistem

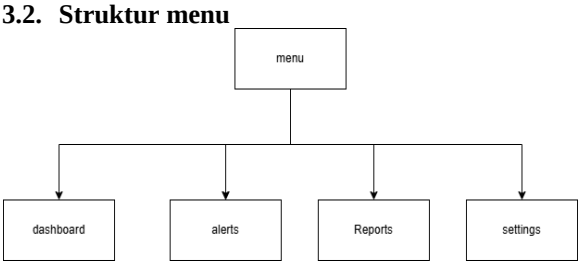
Pendekatan modular diterapkan agar sistem lebih fleksibel dan mudah dikembangkan. Tiga komponen utama yaitu *Go Agent*, *Python Engine*, dan *Dashboard Monitoring* berkomunikasi melalui protokol *REST API*. Integrasi ini memungkinkan sistem melakukan pemantauan, analisis, dan pelaporan secara bersamaan dan terpusat.



Gambar 1. *Use case diagram*

Gambar 1. Diagram ini memperlihatkan bagaimana pengguna (Admin) dan pihak eksternal (Penyerang) berinteraksi dengan komponen sistem SecureShield. Admin bertugas memantau status keamanan dan merespons peringatan melalui *Dashboard Monitoring*. Sementara itu, Penyerang mengirimkan berbagai jenis permintaan HTTP seperti login, unggahan file, dan penyisipan parameter berbahaya. Permintaan ini pertama kali ditangkap oleh *Go Agent*, lalu diteruskan ke *Python Engine* untuk dianalisis menggunakan model *Random Forest*. Berdasarkan hasil klasifikasi, sistem menentukan tindakan lanjut apakah akan memproses atau memblokir permintaan tersebut. Proses ini mencakup deteksi terhadap *SQL Injection*, *XSS*, dan *Webshell*. Seluruh aktivitas tercatat dalam log dan divisualisasikan melalui *dashboard* secara real-time.

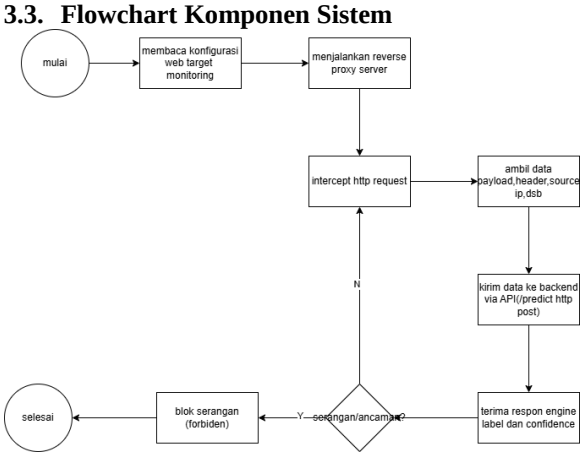
3.2. Struktur menu



Gambar 2. Strukur *menu*

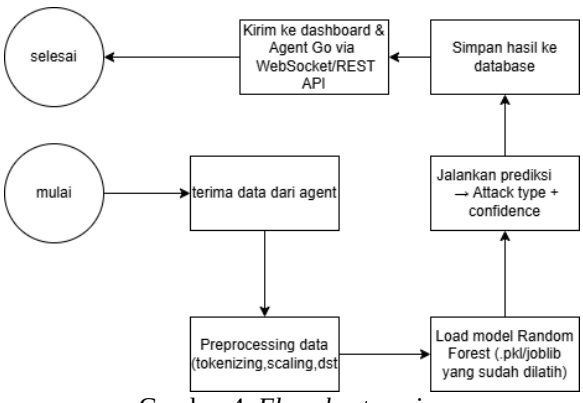
Struktur *menu* pada *dashboard SecureShield* dirancang agar intuitif dan mudah diakses oleh pengguna. Menu utama terdiri dari empat bagian utama yaitu *Dashboard*, *Alerts*, *Reports*, dan Settings. Masing-masing komponen memiliki peran dalam menyajikan informasi serangan, konfigurasi sistem, serta pelaporan aktivitas keamanan

3.3. Flowchart Komponen Sistem



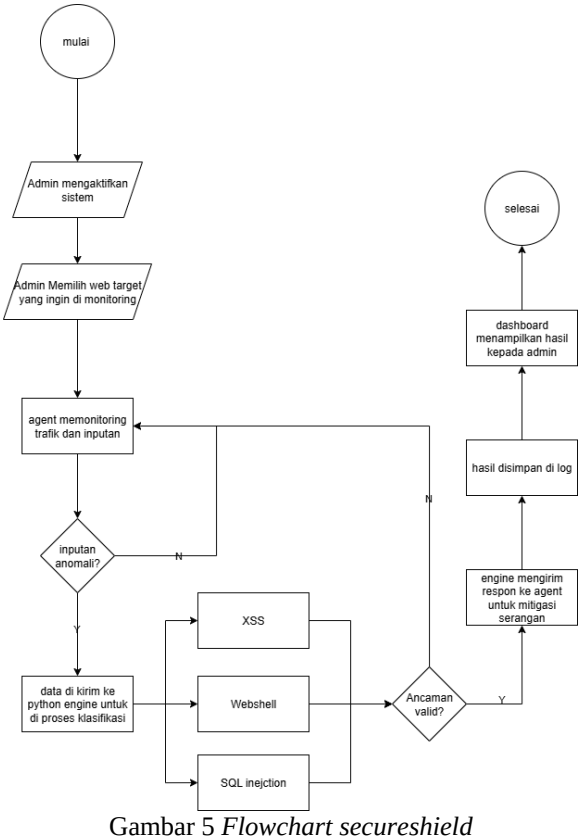
Gambar 3. Flowchart modular *agent* secureshield

Gambar 3 ini menggambarkan alur proses modular yang dijalankan oleh komponen *Go Agent* dalam sistem *SecureShield*. Proses dimulai dengan membaca konfigurasi target web yang akan dipantau, kemudian menjalankan *reverse proxy server* untuk mengintersepsi seluruh permintaan HTTP. Permintaan tersebut kemudian dianalisis: jika terdeteksi sebagai serangan atau ancaman, maka akan diblokir dan tidak diteruskan ke *server* tujuan (*forbidden*). Jika tidak, maka sistem akan mengambil fitur dari permintaan tersebut seperti *payload*, *header*, dan *IP* sumber. Data ini dikirimkan ke *backend Python Engine* melalui *endpoint API/predict* untuk diklasifikasikan oleh model *Random Forest*. Hasil klasifikasi berupa label (aman/serangan) dan nilai *confidence* diterima kembali oleh *Go Agent* untuk menentukan tindakan lanjutan. Diagram ini menunjukkan bagaimana sistem mendeteksi dan merespons serangan secara otomatis sebelum permintaan mencapai *server* aplikasi.



Gambar 4. *Flowchart engine*

Flowchart ini memvisualisasikan alur kerja *Python Engine* dalam sistem SecureShield. Proses dimulai dari penerimaan data yang dikirim oleh *Go Agent*. Data ini kemudian diproses dalam tahap pra-pemrosesan, termasuk tokenisasi, normalisasi, dan transformasi numerik agar sesuai dengan input model pembelajaran mesin. Selanjutnya, sistem memuat model *Random Forest* yang telah dilatih sebelumnya untuk melakukan klasifikasi serangan berdasarkan data yang diterima. Hasil klasifikasi berupa jenis serangan beserta nilai keyakinan (confidence) disimpan dalam basis data untuk pencatatan. Di samping itu, hasil tersebut juga dikirim kembali ke *Go Agent* dan *Dashboard* melalui WebSocket atau REST *API* guna memberikan umpan balik real-time, baik dalam bentuk notifikasi kepada admin maupun tindakan pemblokiran otomatis terhadap permintaan yang terindikasi berbahaya.



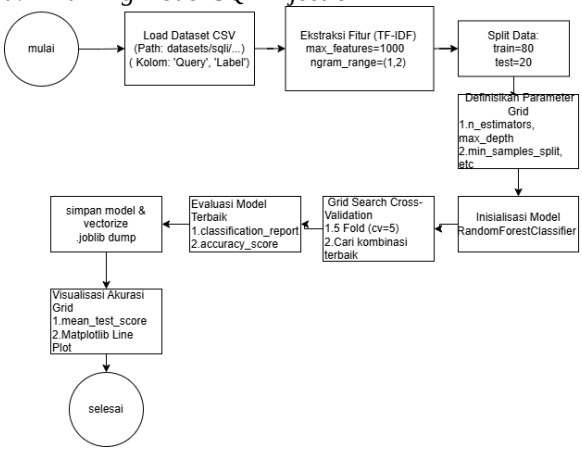
Gambar 5 *Flowchart secureshield*

Ini merupakan alur *Flowchart* sistem,yaitu sistem dimulai dengan admin sebagai aktor utama yang mengaktifkan sistem melalui *dashboard* monitoring *trafic* dan lalu lintas yang di lakukan go agent untuk mendeteksi anomali pada *trafic* lalu lintas,inputan, ataupun *query* yang sekiranya anomali,jika di temukan anomali maka *go agent* akan mengirimkan data anomali tersebut untuk di proses oleh klasifikasi *python secure engine* yang dimana berisikan model *machine learning* yang telah di latih untuk mendeteksi *Webshell,XSS,sql injection*. jika pendeteksian mengeluarkan nilai ancaman valid maka sistem akan melakukan mitigasi otomatis (blokir ip/hapus file atau *query* yang terdeteksi sebagai ancaman) dan sistem akan memberikan notifikasi pada admin dan menyimpan log serangan yang terdeteksi,jika tidak terdeteksi ancamana atau ancamanya = *false* maka sistem kembali memonitoring dan mendeteksi anomali dan mengulangi prosesnya lagi.

3.4. Training Model *Random Forest*

Model dilatih secara terpisah untuk masing-masing jenis serangan dengan alur sebagai berikut:

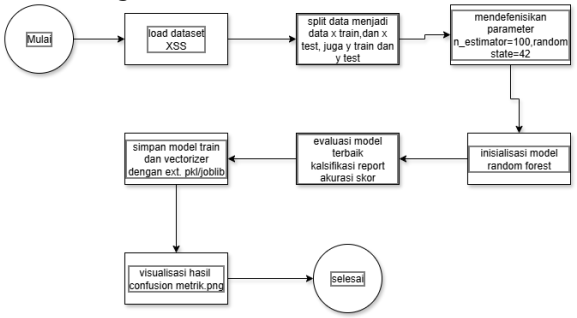
a. Training model SQL Injection



Gambar 6. *Training model SQL Injection*

Alur pelatihan dimulai dari pengambilan dataset lokal, konversi ke *TF-IDF*, evaluasi performa model dengan *GridSearchCV*, dan penyimpanan model terbaik sebagai file *.pkl*

b. Training model XSS

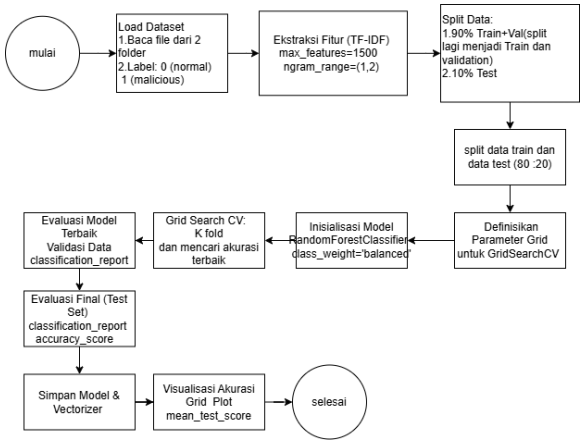


Gambar 7. *Training model XSS*

Data perintah javascript berbahaya dan parameter dianalisis untuk mendeteksi pola XSS. Proses meliputi pelabelan data, pemrosesan, validasi performa, dan pembentukan model *Random Forest*.

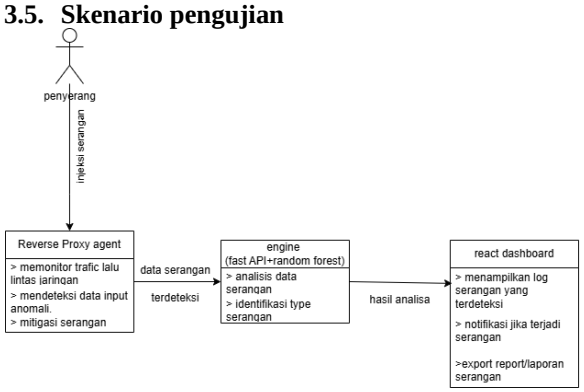
c. Training model Webshell

Dataset berupa perintah sistem diproses dan diubah ke representasi vektor. Model kemudian dievaluasi dengan *cross-validation* sebelum disimpan.



Gambar 8. *Training model XSS*

3.5. Skenario pengujian

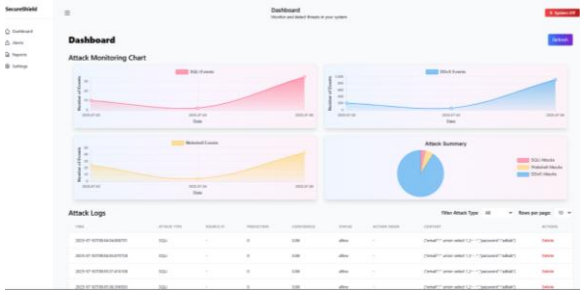


Gambar 9. Skenario pengujian

Gambar 9 menjelaskan alur pengujian sistem keamanan web untuk deteksi dan respons otomatis serangan *(SQLi, Web Shell, DDoS*). Proses diawali simulasi serangan oleh *attacker*. *Reverse Proxy Agent* memantau lalu lintas HTTP, mengekstrak data, dan melakukan mitigasi awal. Data terintercept dikirim ke *Engine* (berbasis *FastAPI* dan *Random Forest*) untuk identifikasi jenis serangan beserta *confidence score*. Hasil analisis kemudian dikirim ke React *Dashboard* guna menampilkan log serangan, memberikan notifikasi *real-time*, dan menyediakan fitur ekspor laporan audit.

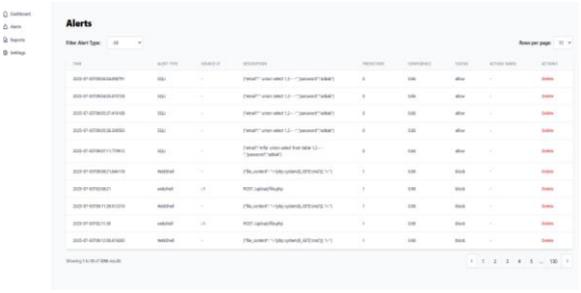
4. HASIL DAN PEMBAHASAN

4.1. Implementasi Hasil



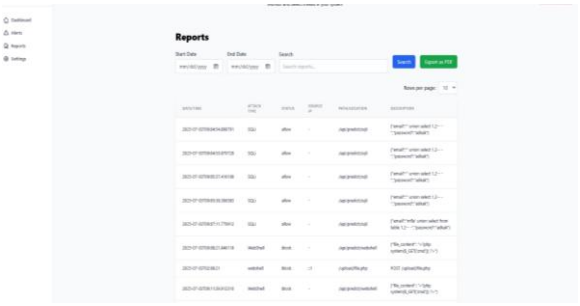
Gambar 10. Tampilan *dashboard monitoring*

Pada gambar 10 ini dapat dilihat bahwa *dashboard* berfungsi sebagai antarmuka untuk menampilkan hasil deteksi serangan yang di lengkapi dengan



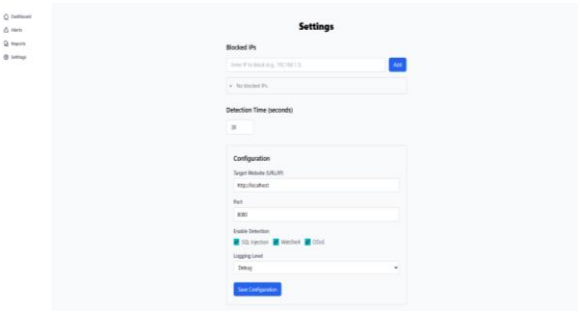
Gambar 11. Tampilan *alerts*

Pada Gambar 11 ini dapat di lihat bahwa *page alerts* berhasil menampilkan notifakasi/pemberitahuan bahwa adanya serangan



Gambar 12 Tampilan *Reports*

Pada Gambar 12 ini dapat dilihat bahwa system ini sudah di lengkapi dengan fitur *reports* untuk ekspor laporan terkait serangan.



Gambar 13. Tampilan settings

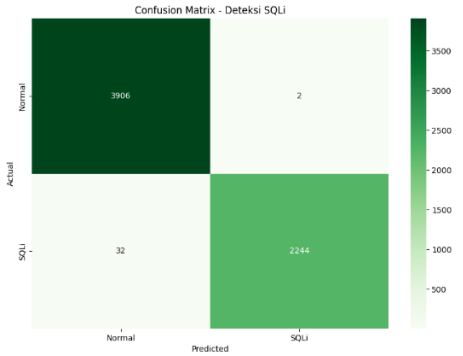
Pada Gambar 13 bisa di lihat bahwa system telah menyediakan fungsi blok ip manual dan juga konfigurasi untuk *web* yang ingin di monitoring.

4.2. Evaluasi Kinerja Machine Learning

Dalam menilai kinerja sistem klasifikasi, khususnya untuk pendeteksian serangan siber, terdapat empat indikator penting yang digunakan, yaitu *True Positive (TP), True Negative (TN), False Positive (FP), dan False Negative (FN). True Positive (TP)* mengacu pada jumlah serangan yang berhasil dikenali dan diklasifikasikan secara akurat sebagai serangan oleh sistem. Sebaliknya, *True Negative (TN)* menunjukkan seberapa banyak trafik yang memang

normal dan diidentifikasi dengan benar sebagai non-ancaman. Sementara itu, *False Positive (FP)* terjadi ketika trafik yang seharusnya tidak berbahaya justru dideteksi sebagai serangan—fenomena ini dikenal juga sebagai alarm palsu, karena memberikan peringatan yang tidak tepat. Yang paling berisiko adalah *False Negative (FN)*, yaitu kondisi saat serangan nyata tidak terdeteksi dan justru dianggap aman, yang dapat membuka celah bagi ancaman untuk lolos dari sistem keamanan

4.3. *SQL Injection*



Gambar 14.*Confusion matrix SQLI*

Inisialisasi: TP = 1993, TN = 3860, FP = 40, FN = 137

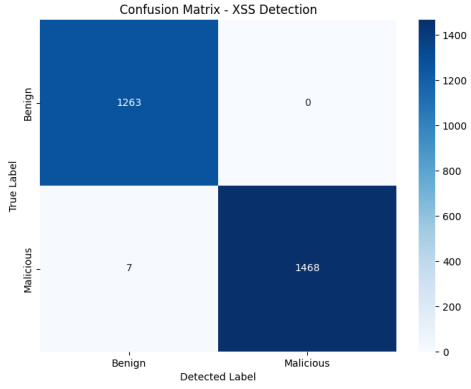
$$Akurasi = \frac{TP+TN}{\text{TP+TN+FP+FN}} = \frac{2244+3906}{2244+3906+2+32} = \frac{6184}{6150} \approx 0.9945 \quad (2)$$

$$Precision = TP / (TP + FP) = 1993 / (1993 + 40) \approx 0.9803 \quad (3)$$

$$Recall = TP / (TP + FN) = 1993 / (1993 + 137) \approx 0.9357 \quad (4)$$

$$F1 = 2 * (0.9803 * 0.9357) / (0.9803 + 0.9357) \approx 0.9576 \quad (5)$$

4.4. *Cross Site Scripting (XSS)*



Gambar 15. *Confusion matrix XSS*

Inisialisasi: TP = 1468, TN = 1263, FP = 0, FN = 7

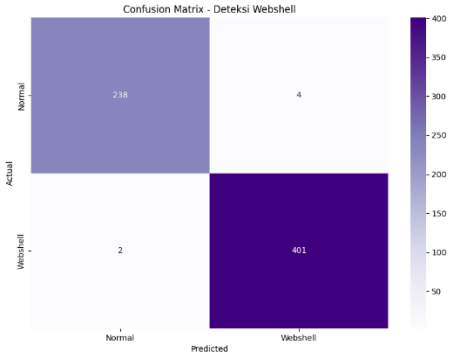
$$Accuracy = \frac{TP + TN}{\text{TP + TN + FP + FN}} = \left(\frac{1468 + 1263}{1468 + 1263 + 0 + 7}\right) = \frac{2731}{2738} \approx 0.9974 \quad (6)$$

$$Precision = \frac{1468}{1468 + 0} = \frac{1468}{1468} = 1.0000 \quad (7)$$

$$Recall = \frac{1468}{1468 + 7} = \frac{1468}{1475} \approx 0.9953 \quad (8)$$

$$F1 - Score = 2 \times (1.0000 \times 0.9953) / (1.0000 + 0.9953) = 2 \times 0.9953 / 1.9953 \approx \mathbf{0.9976} \quad (9)$$

4.5. *WebShell*



Gambar 15. *Confusion matrix Webshell*

Inisialisasi: TP = 401, TN = 238, FP = 4, FN = 2

$$Accuracy = \frac{TP + TN}{\text{TP + TN + FP + FN}} = \frac{401 + 238}{401 + 238 + 4 + 2} = \frac{639}{645} \approx 0.9907 \quad (10)$$

$$Precision = \frac{TP}{TP + FP} = \frac{401}{401+4} = \frac{401}{405} \approx 0.9901 \quad (11)$$

$$Recall = \frac{TP}{TP + FN} = \frac{401}{401+2} = \frac{401}{403} \approx 0.9950 \quad (12)$$

$$F1 = 2 \times (0.9901 \times 0.9950) / (0.9901 + 0.9950) = 2 \times 0.9851 / 1.9851 \approx 0.9926 \quad (13)$$

Tabel 1. Performa Model Pada Data Uji Statis

Model Deteksi	Akurasi	Presisi	Recall	F1-Score
SQL Injection	0.9945	0.9991	0.9859	0.9925
XSS	0.9974	1.0000	0.9953	0.9976
Webshell	0.9907	0.9901	0.9950	0.9926

Pada Tabel 1. Ini ringkasan hasil pengujian model deteksi serangan berbasis *Random Forest*. Seluruh model menunjukkan performa yang sangat baik, ditandai dengan akurasi yang melampaui angka 99%, yang berarti kemampuan klasifikasi sistem tergolong tinggi. Presisi tertinggi ditunjukkan oleh model deteksi *XSS* yang mencapai nilai sempurna (1.0000), tanpa terjadi kesalahan dalam klasifikasi positif, sementara model *Webshell* unggul dalam hal *recall*. Secara keseluruhan, nilai *F1-Score* yang

mendekati angka maksimal pada seluruh model menunjukkan konsistensi dan keandalan sistem dalam mendeteksi serangan secara efektif dan efisien di lingkungan keamanan *web*, temuan ini sejalan dengan Leelavathy et al. [7] yang menunjukkan bahwa kombinasi deteksi otomatis dan *real-time response* dapat meningkatkan efektivitas sistem dalam mencegah serangan *SQL Injection* dan *XSS*.

Tabel 2. Pengujian zero day (serangan yang belum dikenali)

Jenis Serangan	*Total Payload*	Status HTTP	Deteksi oleh *SecureShield*	Keterangan Teknis
SQL Injection	50 *Payload*	403 *Forbidden* (35 kasus),dan 15 kasus 500 internal server error.	Terdeteksi sebagian (70%)	Sebagian besar *Payload* berhasil diblokir oleh mekanisme filtering HTTP SecureShield. Sisanya gagal eksploitasi karena error struktur SQL dari aplikasi.
XSS	50 *Payload*	403 *Forbidden* (22 kasus)200 OK (28 kasus)	Terdeteksi sebagian (44%)	Deteksi berbasis keyword XSS mencegah beberapa *Payload* masuk. Namun, *Payload* yang lolos memerlukan analisis klien-*side* untuk konfirmasi eksekusi skrip.
Webshell	50 *Payload*	403 *Forbidden* (40 kasus)204 *No Content* (10 kasus)	Terdeteksi mayoritas (80%)	*Payload* berbasis PHP yang mengandung *signature* umum berhasil dikenali.

Pada table 2.Pengujian zero-day dilakukan untuk mengevaluasi kinerja sistem SecureShield dalam menghadapi serangan yang belum dikenali sebelumnya. Dari total 150 payload yang terdiri dari serangan SQL Injection, Cross-Site Scripting (XSS), dan Webshell, sistem menunjukkan kemampuan deteksi awal yang cukup signifikan. Tingkat keberhasilan deteksi tertinggi terdapat pada serangan Webshell (80%), disusul SQL Injection (70%) dan XSS (44%). Hasil ini menunjukkan bahwa SecureShield berpotensi berperan sebagai lapisan pertahanan awal terhadap ancaman web, meskipun masih diperlukan pengembangan lebih lanjut untuk meningkatkan efektivitas pada payload yang lebih kompleks dan dinamis

4.6. Pengujian Fungsional Dan Skenario serangan *end to end.*

Pengujian ini meliputi uji Kesehatan layanan sistem,uji aktivasi dan deaktivasi sistem,dan uji deteksi serangan secara *end to end.*

Tabel 2. Hasil pengujian fungsional dan end to end

Kategori	Skenario	Hasil	Keterangan
Kesehatan Layanan	Status *Agent* dan *Engine*	Lulus	Komponen aktif dan responsif
Aktivasi Sistem	API *On*/*Off*	Lulus	Sistem dapat dikendalikan melalui antarmuka
Serangan *SQLi*	Pemblokiran dan pencatatan	Lulus	Diblock, dan pencatatan sesuai
Serangan *XSS*	Pemblokiran dan pencatatan	Lulus	Deteksi, blok, dan pencatatan sesuai
Serangan *Webshell*	Pemblokiran dan pencatatan	Lulus	Semua fungsi berjalan sesuai harapan

Pada Tabel 2. Pengujian dilakukan untuk mengevaluasi kinerja sistem dalam aspek fungsionalitas dan deteksi serangan secara menyeluruh. Sistem berhasil merespons aktivasi serta deaktivasi layanan dengan baik, dan menunjukkan performa yang stabil dalam pemantauan status komponen. Selain itu, sistem mampu mendeteksi dan memblokir serangan *SQLi, XSS, dan Webshell* secara efektif, disertai pencatatan yang berjalan sebagaimana mestinya. Hasil ini mengindikasikan bahwa sistem berfungsi sesuai dengan skenario yang dirancang.

4.7. Pengujian ketahanan sistem (*stress test*)

Pengujian ini ditujukan untuk menilai stabilitas sistem saat menerima beban tinggi secara bersamaan.

Tabel 3. Pengujian ketahanan sistem

Total Request	Sukses	Gagal	Waktu Total (s)	RPS	Status
10	10	0	0.43	23.13	Lulus
100	100	0	2.48	40.37	Lulus
1000	1000	0	29.89	33.26	Lulus
10000	10000	0	325.00	30.67	Lulus

Pengujian ketahanan sistem dilakukan untuk mengevaluasi stabilitas layanan saat menerima permintaan dalam jumlah besar secara simultan. Hasil pengujian menunjukkan bahwa sistem mampu menangani hingga 10.000 *request* tanpa mengalami kegagalan, dengan waktu eksekusi yang masih dalam batas wajar. Nilai *Requests* Per *Second* (RPS) cenderung stabil, meskipun jumlah *request* meningkat secara signifikan. Hal ini mengindikasikan bahwa sistem memiliki ketahanan dan performa yang baik terhadap beban tinggi.

4.8. Pengujian *Blackbox* lengkap Setiap Komponen Sistem *SecureShield*.

Pengujian ini di lakukan untuk menganalisis secara keseluruhan kinerja dari sistem secureshield.

Tabel 4. Hasil pengujian *blackbox secureshield*

No	Komponen	Fungsi di uji	Aksi pengujian	Hasil	Status
1	*Python Engine*	Pemeriksaan status layanan	Akses endpoint /api/status, pastikan sistem merespons dengan status 200 OK	Respons berhasil	Lulus
2	*Python Engine*	Aktivasi dan deaktivasi engine deteksi	Kirim *request* ke /engine/activate dan /engine/deactivate, amati status engine	Status berubah sesuai	Lulus
3	*Python Engine*	Pengambilan data laporan serangan	Akses endpoint /api/report, pastikan data JSON laporan ditampilkan	Data tampil lengkap	Lulus
4	*Python Engine*	Menampilkan data notifikasi serangan	Akses endpoint /api/*alerts* dan verifikasi format JSON	Struktur JSON valid	Lulus
5	*Python Engine*	Menampilkan dan mengatur IP terblokir	Uji endpoint /api/settings/blockedIPs, tambahkan dan cek status IP yang diblokir	Fungsi berjalan normal	Lulus
6	*Go Agent*	Meneruskan trafik HTTP normal	Kirim *request* biasa ke *server* target dan pastikan tidak terblokir	Trafik aman diteruskan	Lulus
7	*Go Agent*	Deteksi serangan *SQL Injection* (*SQLi*)	Kirim payload *SQL Injection*, amati apakah klasifikasi dilakukan dan status diblokir	Serangan terdeteksi	Lulus
8	*Go Agent*	Deteksi serangan XSS	Kirim *request* javascript yang teridentifikasi berbahaya	Permintaan diblokir	Lulus
9	*Go Agent*	Deteksi upload *Webshell*	Upload *file* .php mencurigakan, amati log engine dan *dashboard*	*File* diblokir dan dicatat	Lulus
10	*Dashboard Monitoring*	Menampilkan log serangan	Cek halaman log setelah simulasi serangan, pastikan data baru muncul	Log tercatat di *dashboard*	Lulus
11	*Dashboard Monitoring*	Peringatan *real-time*	Cek apakah notifikasi muncul otomatis saat ada deteksi serangan	Notifikasi muncul otomatis	Lulus
12	*Dashboard Monitoring*	Fitur ekspor laporan ke PDF	Klik tombol "Export as PDF", pastikan *file* laporan dapat diunduh	*File* berhasil disimpan	Lulus
13	*Dashboard Monitoring*	Pengaturan manual blokir IP & *threshold*	Tambahkan IP secara manual ke daftar blokir, simpan dan uji respon terhadap IP tersebut	IP berhasil diblokir	Lulus

Pengujian *blackbox* dilakukan untuk mengevaluasi respons masing-masing komponen sistem *SecureShield* berdasarkan masukan yang diberikan, tanpa memperhatikan struktur internalnya. Uji coba mencakup 14 skenario pengujian terhadap *engine API*, agen monitoring, hingga komponen *dashboard*, termasuk proses deteksi terhadap serangan *SQL injection*, *XSS*, dan *Webshell*. Setiap komponen diuji untuk memastikan dapat merespons input secara tepat, memproses data dengan format yang valid, serta menangani serangan atau anomali secara efektif. Seluruh skenario berhasil dijalankan dengan status *lulus*, yang menunjukkan bahwa sistem telah memenuhi fungsionalitas sesuai spesifikasi

.

5. KESIMPULAN DAN SARAN

Penelitian ini telah berhasil merancang serta menerapkan sistem keamanan web bernama *SecureShield* yang menggunakan pendekatan *machine learning* berbasis algoritma *Random Forest* untuk mengidentifikasi serangan *SQL Injection*, *Cross-site Scripting (XSS)*, dan *Webshell* secara *real-time*. Arsitektur sistem terdiri atas tiga bagian utama, yakni *Go Agent* untuk memantau lalu lintas HTTP, *Python Engine* sebagai modul klasifikasi serangan, serta *Dashboard Monitoring* untuk penyajian informasi dan laporan keamanan. Berdasarkan hasil pengujian secara fungsional, *blackbox*, dan uji ketahanan (*stress test*), sistem menunjukkan performa tinggi dengan tingkat akurasi melebihi 99%, waktu respons yang cepat, serta kemampuan menangani beban akses secara simultan tanpa error. Capaian ini mendukung hasil penelitian sebelumnya mengenai keunggulan metode *ensemble learning* seperti *Random Forest* dalam penerapan keamanan siber [4], [5], [8].

Selain itu, sistem mampu menghasilkan notifikasi otomatis dan mencatat aktivitas serangan ke dalam log, sehingga sangat potensial digunakan dalam skenario institusional seperti perguruan tinggi maupun sistem layanan publik digital [2],[3].

Untuk pengembangan di masa mendatang, disarankan agar sistem diperluas cakupannya dalam mendeteksi jenis serangan lainnya seperti *Remote Code Execution (RCE)*, *Directory Traversal*, serta *zero-day attack* yang lebih sulit dikenali [9]. Penggabungan fitur *adaptive learning* atau pelatihan ulang otomatis (*online retraining*) dari data baru juga dapat dipertimbangkan guna mempertahankan akurasi sistem seiring evolusi teknik serangan [10].

DAFTAR PUSTAKA

[1] M. Nasereddin, A. ALKhamaiseh, M. Qasaimeh, and R. Al-Qassas, "A systematic review of detection and prevention techniques of SQL injection attacks," 2023, *Taylor and Francis Ltd.* doi: 10.1080/19393555.2021.1995537.

[2] BADAN SIBER DAN SANDI NEGARA RI, hss.gov, and Id-SIRTII /CC, "Lanskap Keamanan Siber Indonesia," *Id-SIRTII /CC*, no. 70, pp. 1–21, 2023.

[3] E. Zayid, I. Isah, N. Farah, Y. Adam, and O. Alshehri, "Exploiting the Capabilities of

Classifiers to Examine a Website Defacement Data Set," *Int. J. Comput. Informatics*, vol. 3, no. 3, pp. 9–41, 2024, doi: 10.59992/ijci.2024.v3n3p1.

[4] T. S. Riera, J. R. B. Higuera, J. B. Higuera, J. J. M. Herraiz, and J. A. S. Montalvo, "Prevention and fighting against web attacks through anomaly detection technology. A systematic review," *Sustain.*, vol. 12, no. 12, Jun. 2020, doi: 10.3390/su12124945.

[5] M. Shahid, "Machine Learning for Detection and Mitigation of Web Vulnerabilities and Web Attacks," 2023, [Online]. Available: http://arxiv.org/abs/2304.14451

[6] T. H. Nguyen, X. Dau Hoang, and D. D. Nguyen, "Detecting Website Defacement Attacks using Web-page Text and Image Features," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 7, pp. 215–222, 2021, doi: 10.14569/IJACSA.2021.0120725.

[7] S. Leelavathy, R. Jaichandran, R. Shobana, and S. Bhaskaran, "A Secure Methodology to Detect and Prevent Ddos and Sql Injection Attacks," vol. 12, no. 2, pp. 341–346, 2021.

[8] L. Breiman, Scikit-Learn, and G. Developers, "RandomForestClassifier — scikit-learn 1.3.1 documentation," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001, [Online]. Available: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

[9] X. D. Hoang and N. T. Nguyen, "Detectingwebsite defacements based on machine learning techniques and attack signatures," *Computers*, vol. 8, no. 2, 2019, doi: 10.3390/computers8020035.

[10] X. D. Hoang, T. H. Nguyen, and H. D. Pham, "A novel model for detecting web defacement attacks transformer using plain text features," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 37, no. 1, pp. 232–240, 2025, doi: 10.11591/ijeecs.v37.i1.pp232-240.