BAB II

LANDASAN TEORI

2.1 Landasan Teori

Landasan teori merupakan komponen fundamental dalam penelitian ilmiah yang memberikan kerangka pemahaman konseptual atas isu atau permasalahan yang diangkat. Dalam penelitian ini, fokus utama adalah penerapan *machine learning* dan *deep learning* dalam mendeteksi anomali pada sistem log keamanan jaringan menggunakan data dari firewall Fortigate. Oleh karena itu, kajian literatur ini difokuskan pada teori-teori yang mendasari teknik pendeteksian anomali, pemrosesan data log, serta pendekatan algoritmik melalui model *Autoencoder* dan *One-Class SVM*.

Perkembangan teknologi jaringan dan keamanan informasi saat ini telah memunculkan tantangan baru dalam mitigasi ancaman siber. Sistem keamanan tradisional yang berbasis aturan (rule-based) dinilai tidak cukup tanggap terhadap ancaman siber modern seperti serangan *zero-day* dan penyusupan bertahap. Hal ini mendorong adopsi metode *machine learning* yang adaptif dan berbasis data untuk meningkatkan kemampuan sistem dalam mengenali pola tak biasa secara otomatis. Fu et al. (2025) menunjukkan bahwa pendekatan pembelajaran mesin dan *unsupervised learning* seperti Autoencoder memberikan solusi efisien dalam menghadapi data log yang besar, tidak terstruktur, dan minim label.

Salah satu bentuk representasi data yang sangat penting dalam domain keamanan jaringan adalah log. Data log firewall berisi informasi yang sangat kaya,

mulai dari asal-usul trafik, port yang digunakan, hingga tindakan keamanan yang diambil. Namun, volume dan kecepatan data log yang sangat tinggi menjadi tantangan tersendiri dalam proses deteksi anomali. Oleh karena itu, keberadaan model *deep learning* yang mampu merepresentasikan pola kompleks seperti autoencoder menjadi sangat strategis. Zeng et al. (2024) menegaskan bahwa rekonstruksi data melalui autoencoder mampu menangkap pergeseran pola perilaku akses, bahkan pada tingkat detail jaringan yang halus.

Tidak kalah penting, sistem deteksi modern perlu diintegrasikan dengan tools dan kerangka kerja teknologi yang mendukung *end-to-end pipeline*, mulai dari akuisisi data hingga visualisasi hasil. Dalam konteks ini, platform seperti Wazuh, Jupyter Notebook, dan Streamlit memberikan lingkungan kerja yang kolaboratif dan transparan. Syed et al. (2024) menunjukkan bahwa keberhasilan sistem keamanan tidak hanya bergantung pada algoritma pendeteksian, tetapi juga pada kemampuan sistem untuk beroperasi secara real-time dan mudah dianalisis oleh administrator. Landasan teori ini membentuk basis konseptual atas strategi teknis yang digunakan dalam penelitian untuk mendeteksi anomali pada log sistem Fortigate.

2.1.1 Konsep Dasar *Machine Learning* dan *Deep Learning*

Machine Learning (ML) dan *Deep Learning* (DL) merupakan dua pendekatan utama dalam *artificial intelligence* yang berfokus pada pembelajaran dari data. Keduanya memainkan peran penting dalam pemrosesan data log dan deteksi anomali. Dalam konteks keamanan jaringan, kemampuan algoritma untuk menemukan pola tidak biasa menjadi kunci utama dalam mitigasi risiko.

Menurut Panagoulias et al. (2025), *machine learning* menyediakan kerangka kerja berbasis statistik yang kuat untuk pengenalan pola dan deteksi anomali dalam sistem kritis, sementara *deep learning* menawarkan kemampuan representasi data yang kompleks melalui arsitektur berlapis seperti autoencoder dan RNN.

Pendekatan *Machine Learning* mengubah paradigma pengembangan sistem dari yang berbasis aturan (*rule-based*) menjadi berbasis data (*data-driven*). Dalam sistem keamanan tradisional, deteksi ancaman sangat bergantung pada *signature* atau pola serangan yang telah diketahui sebelumnya. Keterbatasan utama dari pendekatan ini adalah ketidakmampuannya untuk mengidentifikasi serangan baru yang tidak cocok dengan *signature* yang ada. *Machine Learning* mengatasi kelemahan ini dengan mempelajari pola "normal" dari data historis. Dengan demikian, setiap aktivitas yang secara signifikan menyimpang dari pola normal tersebut dapat ditandai sebagai potensi anomali, memberikan kemampuan deteksi yang lebih proaktif dan adaptif (Xin et al., 2018).

Deep Learning merupakan evolusi dari *Machine Learning* yang menggunakan arsitektur jaringan saraf tiruan berlapis-lapis (*deep neural networks*) untuk mempelajari representasi data secara hierarkis dan otomatis. Kemampuan utamanya terletak pada ekstraksi fitur dari data mentah yang kompleks dan tidak terstruktur, seperti teks, gambar, atau data log sekuensial. Dalam konteks analisis log Fortigate, model *Deep Learning* seperti *Autoencoder* dapat secara otomatis mempelajari hubungan non-linier yang rumit antar fitur log tanpa perlu rekayasa

fitur (*feature engineering*) manual yang ekstensif, yang seringkali menjadi tugas yang sulit dan memakan waktu (LeCun et al., 2015).

2.1.1.1 Pengertian *Machine Learning*

Machine learning merupakan cabang dari kecerdasan buatan (*artificial intelligence*) yang berfokus pada pengembangan algoritma yang dapat belajar dari data tanpa harus diprogram secara eksplisit. Algoritma ini mampu mengidentifikasi pola dan membuat prediksi berdasarkan data historis. Menurut Jumani dan Raza (2025), *machine learning* memungkinkan sistem untuk beradaptasi terhadap data baru secara otomatis melalui proses pelatihan yang berkelanjutan, yang sangat bermanfaat dalam konteks deteksi anomali di sistem keamanan siber.

2.1.1.2 Peran *Machine Learning* dalam Deteksi Anomali

Dalam deteksi anomali, *machine learning* digunakan untuk mengenali aktivitas yang menyimpang dari pola normal. Pendekatan ini efektif dalam mendeteksi serangan *zero-day* dan pola akses yang tidak biasa dalam jaringan. Misalnya, penelitian oleh Frimpong et al. (2025) menunjukkan bahwa integrasi model *machine learning* dalam sistem deteksi intrusi pada jaringan fog computing mampu meningkatkan akurasi dalam mendeteksi anomali real-time secara adaptif.

2.1.1.3 Pengertian dan Arsitektur *Deep Learning*

Deep learning adalah subbidang dari *machine learning* yang menggunakan jaringan saraf tiruan (*artificial neural networks*) dengan banyak lapisan tersembunyi. Arsitektur ini mampu mengekstraksi representasi fitur dari data yang kompleks dan berdimensi tinggi. Yao dan Yang (2025) mengembangkan arsitektur *disentangled dual branch convolutional* untuk mendeteksi anomali pada sistem

server, yang menunjukkan efisiensi tinggi dalam memahami pola kompleks dari data log.

2.1.1.4 Jenis-Jenis Arsitektur *Deep Learning*

Beberapa arsitektur yang sering digunakan dalam deteksi anomali antara lain *Convolutional Neural Networks* (CNN), *Recurrent Neural Networks* (RNN), dan *Autoencoder*. CNN cocok untuk data berbasis citra atau spasial, sementara RNN ideal untuk data sekuensial seperti log sistem. Autoencoder sangat efektif untuk *unsupervised anomaly detection*, seperti dijelaskan oleh Meti et al. (2025), karena dapat merekonstruksi data input dan mengukur deviasi sebagai indikasi anomali.

2.1.2 Metodologi CRISP-DM

Untuk memastikan bahwa proyek pengembangan sistem deteksi anomali ini berjalan secara terstruktur, sistematis, dan selaras dengan tujuan yang telah ditetapkan, penelitian ini mengadopsi metodologi *Cross-Industry Standard Process for Data Mining* (CRISP-DM). CRISP-DM adalah model proses yang paling banyak digunakan dalam proyek ilmu data dan *machine learning*, yang menyediakan kerangka kerja non-propietari dan terdokumentasi dengan baik. Metodologi ini bersifat iteratif dan berfokus pada aplikasi praktis, memastikan bahwa hasil akhir tidak hanya akurat secara teknis tetapi juga memberikan nilai nyata bagi pemangku kepentingan (Schröer et al., 2021).

Pentingnya penggunaan metodologi seperti CRISP-DM terletak pada kemampuannya untuk memecah proyek analitik yang kompleks menjadi enam fase utama yang dapat dikelola: *Business Understanding, Data Understanding, Data*

Preparation, Modeling, Evaluation, dan *Deployment*. Setiap fase memiliki tujuan dan tugas spesifik yang saling terkait, memungkinkan tim untuk bergerak maju secara terorganisir dan melakukan iterasi kembali ke fase sebelumnya jika diperlukan. Pendekatan ini membantu mengurangi risiko kegagalan proyek, meningkatkan efisiensi, dan memastikan bahwa solusi yang dikembangkan benar-benar menjawab permasalahan bisnis yang ada (Martínez-Plumed et al., 2021). Dalam konteks penelitian ini, CRISP-DM memandu setiap langkah, mulai dari mendefinisikan ancaman keamanan yang ingin dideteksi hingga mengintegrasikan model ke dalam alur kerja operasional tim keamanan.

2.1.2.1 Business Understanding

Langkah pertama dalam metodologi CRISP-DM adalah memahami tujuan dan kebutuhan bisnis yang ingin dicapai melalui proyek data mining. Tahap ini menentukan arah proyek secara keseluruhan, termasuk definisi masalah dan sasaran operasional yang relevan. Dalam konteks deteksi anomali akses pada log sistem Fortigate, tujuan bisnis adalah meningkatkan ketahanan keamanan jaringan perusahaan dengan mendeteksi akses mencurigakan secara dini.

Muñoz et al. (2025) menekankan bahwa pemahaman konteks bisnis sangat penting dalam memilih variabel data yang relevan, serta dalam menentukan metrik evaluasi yang akan digunakan. Tanpa pemahaman bisnis yang memadai, solusi teknis berpotensi meleset dari tujuan strategis perusahaan.

2.1.2.2 Data Understanding

Setelah memahami tujuan bisnis, tahap berikutnya adalah eksplorasi dan pemahaman data yang tersedia. Proses ini meliputi pengumpulan, deskripsi awal,

dan visualisasi data untuk menemukan potensi masalah kualitas data serta pola awal yang relevan. Data log Fortigate, misalnya, sering kali tidak terstruktur dan mengandung *noise* sehingga perlu dievaluasi terlebih dahulu.

Menurut Neumayer et al. (2024), tahap *data understanding* memungkinkan peneliti mengidentifikasi outlier, nilai hilang, serta struktur korelasi antara fitur. Dalam proyek deteksi anomali, eksplorasi ini penting untuk menentukan apakah anomali dapat dibedakan dari data normal secara statistik atau melalui pemetaan ruang fitur.

2.1.2.3 Data Preparation

Pada tahap ini merupakan fase terpanjang dan paling penting dalam siklus CRISP-DM, di mana data mentah diproses menjadi bentuk yang siap untuk digunakan oleh model pembelajaran mesin. Prosesnya meliputi pembersihan data, transformasi format, *feature engineering*, dan normalisasi. Dalam kasus log Fortigate, data perlu diuraikan dari format syslog menjadi format tabel yang mengandung fitur-fitur seperti IP sumber, waktu akses, dan jenis aktivitas.

Penelitian oleh Ładyżyńska-Kozdraś et al. (2024) menunjukkan bahwa kualitas *data preparation* secara signifikan memengaruhi performa akhir model deteksi, terutama pada sistem real-time. Tanpa normalisasi dan teknik pengurangan dimensi yang tepat, model dapat mengalami *overfitting* atau gagal mengenali anomali.

2.1.2.4 Modeling

Tahap modeling adalah proses membangun dan melatih algoritma pembelajaran mesin atau *deep learning* untuk menemukan pola dari data yang telah

disiapkan. Dalam penelitian ini, model yang digunakan adalah *autoencoder* dan *One-Class SVM*. Pemilihan parameter, arsitektur, dan teknik pelatihan sangat krusial dalam menghasilkan model yang mampu mengenali anomali dengan baik.

Wijaya et al. (2024) menunjukkan bahwa penerapan CRISP-DM pada data akademik dengan algoritma *Naive Bayes* memperlihatkan bahwa setiap fase, terutama *modeling*, harus selalu divalidasi berdasarkan hasil evaluasi sebelumnya agar tidak terjadi kesalahan asumsi. Pada log Fortigate, pendekatan serupa dapat digunakan untuk menentukan batasan deteksi outlier.

2.1.2.5 Evaluation

Setelah model selesai dilatih, tahap evaluasi dilakukan untuk mengukur performa model berdasarkan metrik seperti *precision*, *recall*, *F1-score*, dan *AUC*. Tujuannya adalah untuk mengetahui sejauh mana model mampu membedakan antara data normal dan anomali, serta menghindari kesalahan klasifikasi.

Gonzalez-Jimenez et al. (2025) menggunakan pendekatan serupa untuk mengevaluasi sistem deteksi kesalahan pada motor listrik, menunjukkan pentingnya validasi berbasis metrik untuk menjamin keandalan sistem. Evaluasi pada sistem log juga perlu memperhatikan *false positives* yang terlalu tinggi, karena dapat membanjiri tim keamanan dengan alarm palsu.

2.1.2.6 Deployment

Tahap terakhir adalah *deployment*, yaitu penerapan model ke dalam sistem operasional. Model yang telah dievaluasi akan diintegrasikan ke dalam infrastruktur IT perusahaan agar dapat berjalan secara otomatis dan real-time. Sistem ini harus

dikonfigurasi agar mampu membaca log secara kontinu dan menghasilkan notifikasi bila ditemukan anomali.

Montoya-Murillo et al. (2025) menekankan bahwa keberhasilan deployment bergantung pada kesiapan infrastruktur serta mekanisme pembaruan model yang adaptif terhadap perubahan pola data. Dalam konteks log Fortigate, sistem dapat diintegrasikan dengan tools seperti Wazuh untuk notifikasi otomatis ke tim keamanan siber.

2.1.3 Pemodelan Sistem Menggunakan UML

Pemodelan sistem merupakan tahap penting dalam pengembangan perangkat lunak yang bertujuan untuk merepresentasikan struktur dan alur logika sistem sebelum dibangun secara nyata. Salah satu pendekatan yang paling populer digunakan adalah Unified Modeling Language (UML), sebuah bahasa standar untuk merancang dan memvisualisasikan sistem perangkat lunak secara sistematis dan terstruktur. UML membantu pengembang, analis, dan pemangku kepentingan lainnya untuk memiliki pemahaman yang seragam terhadap sistem yang akan dibangun.

Menurut Bachtiar et al. (2024), penggunaan diagram UML seperti use case, activity, dan sequence berperan penting dalam menggambarkan fungsi dan interaksi antarkomponen dalam sistem, terutama untuk sistem berbasis web dan pemrosesan data log. Diagram tersebut memungkinkan pengujian awal terhadap alur sistem untuk mengidentifikasi potensi kesalahan logika atau redundansi proses sebelum implementasi berlangsung.

2.1.3.1 Use Case Diagram

Use case diagram digunakan untuk memodelkan kebutuhan fungsional sistem dari perspektif pengguna. Diagram ini menunjukkan hubungan antara aktor (pengguna atau sistem eksternal) dan *use case* (layanan atau fungsi yang disediakan oleh sistem). Dalam konteks sistem deteksi anomali pada log Fortigate, *use case* diagram dapat mencakup entitas seperti admin keamanan, sistem Wazuh, dan modul deteksi anomali yang mengakses data log.

Mohammed et al. (2024) menyatakan bahwa use case diagram efektif untuk menjembatani komunikasi antara pengembang dan pemilik sistem, karena diagram ini bersifat intuitif dan tidak membutuhkan pemahaman teknis yang mendalam. Diagram ini juga sangat membantu dalam tahap analisis kebutuhan pengguna sebelum sistem dikembangkan lebih lanjut.

2.1.3.2 Activity Diagram

Activity diagram menggambarkan aliran proses dalam sistem, baik secara linier maupun bercabang, dan sangat bermanfaat untuk memodelkan logika bisnis atau *workflow*. Diagram ini mengilustrasikan bagaimana data atau kontrol berpindah dari satu aktivitas ke aktivitas lain, serta bagaimana percabangan keputusan dan paralelisme dapat terjadi dalam proses sistem.

Fu et al. (2025) menekankan bahwa activity diagram cocok digunakan untuk menganalisis proses log sistem, karena mampu memetakan tahapan-tahapan penting dalam pengolahan log seperti pengambilan data, pra-pemrosesan, deteksi anomali, dan notifikasi. Diagram ini juga membantu mendeteksi *bottleneck* dalam alur proses sistem.

2.1.3.3 Sequence Diagram

Sequence diagram menunjukkan interaksi antarobjek dalam sistem berdasarkan urutan waktu. Diagram ini memperlihatkan bagaimana pesan dikirim antar entitas sistem dan urutan respons yang terjadi. Dalam sistem deteksi anomali log Fortigate, sequence diagram dapat digunakan untuk menggambarkan alur komunikasi antara sistem log, model Autoencoder/SVM, dan server notifikasi atau *dashboard* admin.

Fan et al. (2024) menjelaskan bahwa sequence diagram sangat efektif untuk menguji urutan komunikasi dalam sistem berbasis event dan sangat berguna dalam pengembangan sistem keamanan siber karena mampu mengevaluasi waktu respon antar subsistem. Dengan pemodelan ini, proses deteksi anomali bisa divalidasi secara lebih tepat.

2.1.4 Bahasa Pemrograman Python dan Library Terkait

Python merupakan salah satu bahasa pemrograman yang paling banyak digunakan dalam bidang *data science*, *machine learning*, dan *artificial intelligence*. Fleksibilitas sintaksis, dukungan komunitas yang luas, serta keberagaman pustaka (*library*) menjadi alasan utama mengapa Python sangat populer dalam pengembangan sistem deteksi anomali berbasis data log. Dalam konteks penelitian ini, Python digunakan untuk mengolah data syslog dari Fortigate, membangun model autoencoder dan One-Class SVM, serta menyusun pipeline analitik secara otomatis.

Dalam studi mereka, Pacini et al. (2025) menyatakan bahwa Python menjadi inti dalam penerapan algoritma AI, terutama karena mendukung arsitektur modular dan skalabel melalui integrasi dengan pustaka seperti Scikit-learn dan TensorFlow.

2.1.4.1 Alasan Pemilihan Python

Python dipilih karena kemudahan penggunaannya dan kapabilitasnya dalam menangani data besar dan kompleks. Python memiliki struktur kode yang intuitif dan pendek, sehingga mempermudah kolaborasi lintas tim. Budzyński & Sładkowski (2024) menjelaskan bahwa Python menjadi pilihan utama dalam penelitian transportasi berbasis *machine learning* karena kompatibilitasnya dengan beragam modul visualisasi, integrasi dengan basis data, dan kemampuan scripting otomatis.

2.1.4.2 Library untuk Data Processing

Dalam tahapan *data preparation*, pustaka seperti Pandas dan NumPy digunakan untuk manipulasi data, sedangkan Matplotlib dan Seaborn digunakan untuk visualisasi data. Library ini memungkinkan eksplorasi awal seperti distribusi data, pencarian *outlier*, dan transformasi fitur. Palacios (2025) menyebut bahwa Python sangat mendukung komputasi ilmiah karena performanya yang optimal dengan pustaka seperti NumPy dan integrasi YAML untuk pengaturan parameter konfigurasi otomatis.

2.1.4.3 Library Machine Learning

Untuk keperluan *machine learning*, pustaka seperti Scikit-Learn digunakan secara luas karena menyediakan berbagai model klasik seperti SVM, k-NN, dan Random Forest dengan antarmuka sederhana. Lemenkova (2025) menunjukkan

bahwa Scikit-Learn memungkinkan otomatisasi proses pelabelan dan klasifikasi citra satelit dalam sistem GIS, yang serupa dengan deteksi anomali berbasis log dari segi kompleksitas data.

2.1.4.4 Library Deep Learning

Dalam implementasi *deep learning*, pustaka seperti TensorFlow dan PyTorch merupakan dua *framework* paling dominan. Keduanya memungkinkan pembuatan dan pelatihan arsitektur neural network secara fleksibel, serta mendukung GPU untuk komputasi intensif. Dhumal et al. (2024) memanfaatkan kombinasi Keras dan PyTorch dalam proyek rekonstruksi data CAD berbasis *deep learning*, menunjukkan bahwa Python mendukung pembangunan sistem kompleks dengan efisiensi tinggi.

2.1.5 Sistem Log Keamanan Jaringan

Sistem log memainkan peranan krusial dalam keamanan jaringan karena menjadi sumber utama informasi mengenai aktivitas sistem, lalu lintas jaringan, dan jejak audit pengguna. Data log mencatat setiap interaksi yang terjadi dalam infrastruktur jaringan, baik yang sah maupun berpotensi sebagai ancaman. Dengan memanfaatkan log sebagai sumber data utama, berbagai sistem deteksi ancaman, baik berbasis aturan maupun pembelajaran mesin, dapat dikembangkan untuk mendeteksi aktivitas abnormal atau anomali.

Fu et al. (2025) menjelaskan bahwa analisis data log menjadi salah satu pendekatan paling andal dalam sistem keamanan, karena log menyediakan konteks historis dan real-time untuk mendeteksi anomali dan serangan tersembunyi.

2.1.5.1 Pengertian Log dan Perannya dalam Keamanan

Log adalah rekaman waktu nyata dari aktivitas sistem atau perangkat jaringan, termasuk aktivitas login, akses file, koneksi jaringan, dan penggunaan sumber daya sistem. Dalam konteks keamanan, log menjadi alat utama dalam mendeteksi potensi serangan siber, baik dari luar maupun dalam jaringan. Chuah et al. (2025) dalam tinjauannya terhadap alat korelasi log untuk mendeteksi serangan menyatakan bahwa log dari sistem Linux, NetFlow, dan Windows dapat dikombinasikan untuk membentuk gambaran menyeluruh tentang status keamanan jaringan.

Balani dan Mohammed (2025) menjelaskan bahwa integrasi log dengan teknologi AI mampu memperkuat firewall dalam mengenali pola serangan baru secara proaktif. Hal ini dilakukan dengan cara memonitor log yang terus berkembang dan mengadaptasi model deteksi terhadap perubahan perilaku pengguna jaringan.

2.1.5.2 Tantangan Analisis Data Log

Meskipun log menawarkan potensi besar dalam penguatan sistem keamanan, terdapat berbagai tantangan teknis dalam mengelolanya. Salah satu tantangan utama adalah volume data yang sangat besar dan tidak terstruktur, sehingga membutuhkan proses pra-pemrosesan yang cermat. Selain itu, log sering kali tidak memiliki konsistensi format, misalnya antara syslog Fortigate, NetFlow, dan audit log dari sistem operasi, yang membuat proses parsing menjadi kompleks.

Liu et al. (2025) menyoroti bahwa kualitas dataset log sangat berpengaruh terhadap hasil deteksi anomali. Dataset yang terlalu *regular* atau sudah dibersihkan

secara berlebihan justru dapat menurunkan kemampuan model dalam mendeteksi serangan nyata yang bersifat *subtle* atau jarang terjadi.

Xie dan Ni (2025) juga memperkenalkan model LogMT berbasis *multi-task self-supervised learning* yang secara khusus dikembangkan untuk mendeteksi anomali dalam sekuens log yang kompleks. Mereka menekankan pentingnya teknik pembelajaran mandiri (*self-supervised learning*) untuk mengatasi keterbatasan pelabelan dalam data log.

2.1.6 Wazuh

Wazuh merupakan platform open-source yang digunakan untuk deteksi ancaman, integritas file, respons insiden, dan pemantauan keamanan secara real-time. Wazuh dikembangkan sebagai pengganti OSSEC dan kini telah menjadi salah satu solusi SIEM (*Security Information and Event Management*) yang banyak digunakan di lingkungan enterprise maupun pemerintahan karena bersifat modular dan dapat diintegrasikan dengan sistem log seperti ELK stack (Elasticsearch, Logstash, dan Kibana). Awaisi et al. (2025) menunjukkan bahwa sistem SIEM seperti Wazuh yang digabungkan dengan teknik pembelajaran mesin berbasis attention dapat meningkatkan akurasi deteksi dan efisiensi sistem analitik log pada industri besar..

2.1.6.1 Arsitektur dan Fungsi Wazuh

Wazuh memiliki arsitektur berbasis agen dan server. Agen diinstal pada endpoint (seperti server, firewall, dan perangkat pengguna) untuk mengumpulkan data log dan informasi sistem. Data ini kemudian dikirimkan ke server Wazuh untuk diproses, dianalisis, dan diklasifikasikan. Komponen utama dalam arsitektur

Wazuh meliputi modul deteksi anomali berbasis aturan (*rules engine*), modul analisis integritas, serta pemantauan kerentanan sistem.

Sarker et al. (2024) dalam studi mereka menjelaskan bahwa Wazuh dapat dioptimalkan sebagai solusi SIEM berbasis open-source yang efisien dan skalabel. Platform ini mampu memproses log dalam jumlah besar serta mendeteksi pola-pola akses mencurigakan, terutama bila dikombinasikan dengan teknologi visualisasi seperti Kibana. Dalam sistem mereka, Wazuh berhasil digunakan untuk pemantauan endpoint dan integrasi dengan sistem *threat intelligence*.

2.1.6.2 Peran Wazuh dalam Sistem Penelitian

Dalam penelitian deteksi anomali pada log Fortigate, Wazuh berperan sebagai sistem pengumpul dan pengelola log (*log collector*) yang menerima input dari firewall, mengklasifikasikannya berdasarkan aturan yang sudah ditentukan, serta mengirimkannya ke sistem analitik berbasis *machine learning*.

2.1.7 Fortigate dan Format Syslog

Fortigate adalah solusi firewall berbasis *unified threat management* (UTM) yang disediakan oleh Fortinet. Perangkat ini menggabungkan berbagai fungsi keamanan jaringan seperti filtering, IDS/IPS, antivirus, VPN, dan manajemen bandwidth dalam satu platform terintegrasi. Salah satu keunggulan Fortigate adalah kemampuannya dalam menghasilkan log sistem yang terstruktur secara konsisten, yang kemudian dapat digunakan untuk kebutuhan monitoring, audit, maupun deteksi anomali berbasis *machine learning*. Hornyák (2025) mencatat bahwa log firewall Fortigate dapat dikombinasikan dengan teknik prediktif dan *statistical*

trend analysis untuk memperkirakan risiko keamanan dan menghasilkan alert otomatis yang lebih akurat.

2.1.7.1 Fungsi Fortigate

Sebagai *next-generation firewall* (NGFW), Fortigate menyediakan proteksi terhadap lalu lintas jaringan internal dan eksternal secara menyeluruh. Fortigate mampu mendeteksi dan memblokir serangan berbasis signature maupun perilaku (*behavior-based detection*), menjadikannya solusi utama dalam banyak sistem keamanan perusahaan. Menurut Komadina et al. (2024), Fortigate menghasilkan log yang sangat kaya akan konteks, mencakup metadata seperti alamat IP sumber dan tujuan, protokol, port, waktu, serta tindakan keamanan yang diambil.

Dalam implementasi sistem deteksi anomali, log dari Fortigate menjadi *raw input* yang sangat penting untuk diolah dan dianalisis. Berkat format dan tingkat detail yang dimiliki log ini, proses pelatihan model *machine learning* seperti autoencoder atau One-Class SVM dapat dilakukan secara lebih efisien.

2.1.7.2 Struktur Format Syslog Fortigate

Dalam studi oleh Mohandas et al. (2021), log Fortigate menunjukkan karakteristik yang khas seperti penanda subtype, kode aksi (action=accept atau action=deny), dan identifikasi sesi. Penulis menggarisbawahi bahwa analisis log Fortigate memerlukan *log parser* yang tangguh agar dapat menangani variasi struktur yang muncul karena perbedaan konfigurasi perangkat.

Selain itu, penelitian oleh Meng et al. (2023) menyebutkan bahwa dari 99 atribut log yang dikumpulkan dari Fortigate, hanya 42 yang umumnya relevan untuk deteksi anomali. Sisanya bersifat administratif atau terkait pengaturan

internal, sehingga perlu dibersihkan saat *data preprocessing* untuk meminimalkan kebisingan dalam model.

2.1.8 Deteksi Anomali

Deteksi anomali merupakan teknik penting dalam keamanan siber dan *data analytics* yang digunakan untuk mengidentifikasi pola atau kejadian yang menyimpang dari perilaku normal. Anomali dapat mengindikasikan adanya serangan, kesalahan sistem, atau aktivitas tidak sah yang memerlukan tindakan cepat. Pendekatan ini sering digunakan dalam sistem *intrusion detection*, deteksi penipuan finansial, serta analisis log jaringan dan firewall seperti Fortigate.

Jinpeng et al. (2025) menggunakan pendekatan hybrid autoencoder untuk mendeteksi penipuan pada sistem tenaga listrik, yang menegaskan pentingnya deteksi anomali dalam berbagai sektor termasuk energi, keuangan, dan jaringan.

2.1.8.1 Pengertian dan Tujuan Deteksi Anomali

Deteksi anomali (*anomaly detection*) adalah proses untuk mengenali observasi, kejadian, atau data yang tidak sesuai dengan pola yang diharapkan dalam dataset. Tujuan utama dari pendekatan ini adalah mengidentifikasi kejadian yang langka namun berpotensi signifikan terhadap keamanan atau operasional sistem. Dalam konteks jaringan, deteksi anomali dapat mengungkap aktivitas seperti serangan brute force, login mencurigakan, atau eksfiltrasi data.

Kaliyaperumal et al. (2024) dalam studi mereka mengembangkan pendekatan *hybrid unsupervised learning* dengan kombinasi *autoencoder* dan *One-Class SVM* untuk deteksi anomali di lingkungan IoT. Mereka menyatakan bahwa metode ini mampu mendeteksi berbagai bentuk serangan siber dengan akurasi

tinggi, meskipun tanpa label eksplisit, menjadikannya cocok untuk sistem log seperti Fortigate yang cenderung menghasilkan data dalam jumlah besar namun tidak selalu diberi anotasi.

2.1.8.2 Strategi Deteksi Anomali

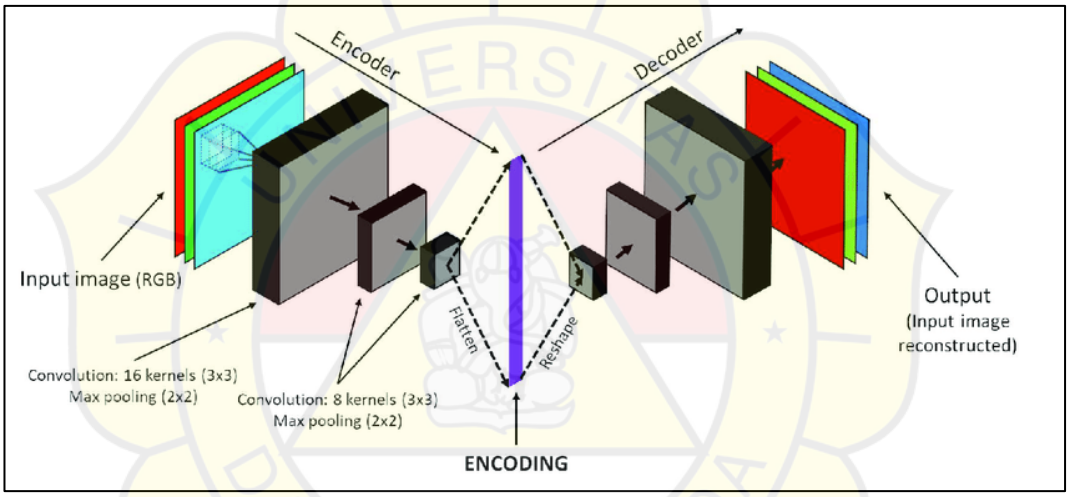
Model *unsupervised* seperti autoencoder dan One-Class SVM sangat relevan untuk log jaringan karena umumnya data log tidak memiliki label. Autoencoder bekerja dengan merekonstruksi data input, dan jika rekonstruksi gagal (dengan error tinggi), maka data tersebut dianggap anomali. Sementara itu, One-Class SVM mempelajari batasan ruang fitur dari data normal dan menolak observasi yang berada di luar batas tersebut. Gabungan dari dua pendekatan ini dapat meningkatkan sensitivitas dan mengurangi kesalahan deteksi.

2.1.9 Autoencoder

Autoencoder adalah jenis jaringan saraf tiruan (*artificial neural network*) yang dilatih secara *unsupervised* dengan tujuan utama untuk mempelajari representasi data yang efisien, biasanya untuk reduksi dimensi atau ekstraksi fitur. Arsitekturnya yang unik terdiri dari dua komponen utama: *encoder* dan *decoder*. *Encoder* mengambil data input dan mengompresnya menjadi representasi laten (*latent representation*) berdimensi lebih rendah. Sebaliknya, *decoder* mengambil representasi laten ini dan mencoba merekonstruksi data input asli kembali. Model ini dilatih dengan meminimalkan perbedaan antara input asli dan output yang direkonstruksi, yang dikenal sebagai *reconstruction error* (LeCun et al., 2015).

2.1.9.1 Arsitektur Autoencoder

Secara umum, autoencoder terdiri dari dua komponen utama: encoder dan decoder. Encoder bertugas mengubah data input ke dalam bentuk representasi tersembunyi (*latent space*), sedangkan decoder mencoba merekonstruksi kembali data input dari representasi tersebut. Arsitektur ini dapat dikembangkan menjadi berbagai bentuk seperti *variational autoencoder*, *convolutional autoencoder*, dan *sparse autoencoder* tergantung pada kompleksitas data dan tujuan model.



Gambar 2. 1 Arsitektur Autoencoder(Jimenez et al., 2020)

Nayak et al. (2024) mengembangkan arsitektur *attention-enabled convolutional autoencoder* untuk mendeteksi anomali pada data citra industri. Penelitian tersebut menunjukkan bahwa penambahan lapisan *attention* pada encoder meningkatkan efisiensi ekstraksi fitur penting dan menghasilkan akurasi deteksi anomali yang lebih tinggi.

2.1.9.2 Mekanisme Deteksi Anomali

Dalam konteks deteksi anomali, autoencoder dilatih hanya menggunakan data normal. Setelah pelatihan, setiap input yang tidak dapat direkonstruksi dengan baik—ditandai oleh *reconstruction error* yang tinggi—dianggap sebagai anomali.

Oleh karena itu, proses inferensi tidak memerlukan label, menjadikan autoencoder sangat efisien untuk dataset besar seperti log Fortigate yang tidak diberi anotasi.

Zeng et al. (2024) menggunakan pendekatan autoencoder untuk deteksi anomali dalam sistem microservice berbasis *multidimensional feature reconstruction*. Mereka menunjukkan bahwa membandingkan *reconstruction error* dari input terhadap ambang batas (threshold) memberikan hasil deteksi yang dapat diandalkan dalam sistem real-time.

2.1.9.3 Formula dan Threshold

Proses deteksi anomali menggunakan autoencoder umumnya didasarkan pada nilai *reconstruction error* yang dihitung antara input asli x dan output $\hat{x}$ rekonstruksi . Formula umumnya adalah:

$$\mathrm{RE} = |x - \hat{x}|^2 \tag{1}$$

Di mana RE adalah *reconstruction error*, dan biasanya dihitung menggunakan jarak Euclidean atau *mean squared error (MSE)*. Jika nilai RE melebihi batas ambang (threshold) yang ditentukan berdasarkan distribusi error pada data pelatihan, maka data tersebut diklasifikasikan sebagai anomali.

Markovic et al. (2024) melakukan evaluasi terhadap berbagai arsitektur autoencoder dan menyatakan bahwa pemilihan threshold optimal dapat dilakukan melalui analisis distribusi error pada data validasi. Mereka menggunakan *percentile-based thresholding* untuk menentukan batas error yang masih bisa dianggap normal.

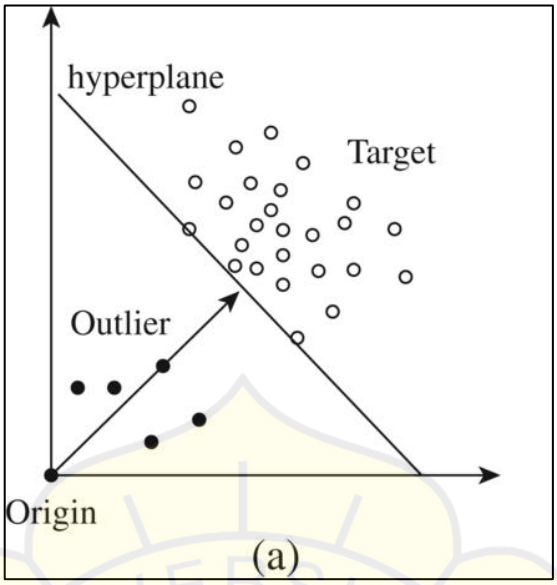
2.1.10 One-Class SVM

One-Class Support Vector Machine (OC-SVM) merupakan algoritma *unsupervised learning* yang digunakan secara luas untuk deteksi anomali. Tidak seperti SVM biasa yang memisahkan dua kelas, OC-SVM dirancang untuk mempelajari distribusi data normal dan mengidentifikasi data yang berada di luar *decision boundary* sebagai anomali. OC-SVM sangat cocok diterapkan dalam analisis sistem log seperti Fortigate karena umumnya data log tidak memiliki label dan bersifat sangat tidak seimbang.

Cao et al. (2025) menerapkan SVM dalam mitigasi serangan APT dan menekankan keunggulan OC-SVM dalam mendeteksi pola yang sangat jarang muncul, terutama dalam skenario multistage attack.

2.1.10.1 Konsep OC-SVM

OC-SVM bekerja dengan memetakan data input ke ruang berdimensi lebih tinggi menggunakan fungsi kernel, lalu mencari hiperplane yang memisahkan semua data dari asal (*origin*) dengan margin maksimum. Titik data yang berada di luar margin dianggap sebagai *outlier*. Fungsi kernel memungkinkan OC-SVM menangani data non-linear dan mendeteksi anomali pada berbagai bentuk distribusi data.



Gambar 2. 2 Arsitektur One-Class SVM (Liang, et al, 2018)

Ghiasi et al. (2024) menerapkan OC-SVM untuk mendeteksi anomali pada sistem pemantauan rel kereta menggunakan data geometri trek. Mereka menunjukkan bahwa OC-SVM efektif dalam mengenali kondisi aneh meskipun data latih hanya berisi data normal, menjadikannya cocok untuk aplikasi keamanan jaringan dan sistem log.

2.1.10.2 Parameter dan Interpretasi Output

Beberapa parameter penting dalam OC-SVM meliputi:

1. **Kernel**: Fungsi pemetaan data ke ruang fitur yang lebih tinggi, misalnya *Radial Basis Function (RBF)*, linear, atau polynomial. Kernel RBF adalah yang paling umum digunakan untuk deteksi anomali.
2. **ν (nu)**: Parameter yang mengontrol jumlah maksimum *outlier* dan jumlah *support vectors* minimum. Nilai ν berada di antara 0 dan 1.
3. **γ (gamma)**: Parameter kernel RBF yang mengatur seberapa jauh pengaruh satu titik data terhadap bentuk *decision boundary*.

Avola et al. (2021) menjelaskan bahwa pemilihan parameter ini sangat mempengaruhi sensitivitas model terhadap anomali. Dalam eksperimen mereka untuk pengawasan video UAV berbasis tekstur, penyesuaian parameter kernel dan ν terbukti krusial dalam menurunkan tingkat false positive.

2.1.10.3 Formula Matematis

Secara matematis, OC-SVM menyelesaikan permasalahan optimisasi sebagai berikut:

minimize

$$\frac{1}{2}|\mathrm{w}|^2 + \frac{1}{\nu \mathrm{n}}\sum \xi_{\mathrm{i}} - \quad (1)$$

dengan

$$\left(w \cdot \phi(x_i)\right) \geq \rho - \xi_i, \qquad \xi_i \geq 0 \quad (2)$$

di mana *w* adalah vektor bobot, ξ_i adalah slack variable untuk pelanggaran margin, *v* adalah parameter nu, *n* adalah jumlah data, dan ρ adalah threshold. Fungsi kernel $\varphi(x_i)$ digunakan untuk memetakan data dari ruang input ke ruang fitur non-linier. Kernel RBF yang umum digunakan memiliki rumus:

$$K(x, x') = \exp(-\gamma|x - x'|^2) \quad (1)$$

Formulasi matematis dari OC-SVM diawali dengan memetakan data $\phi(\mathrm{xi})$ ke ruang fitur $\phi(\mathrm{xi})\backslash\mathrm{phi}(\mathrm{x\_i})\phi(\mathrm{xi})$ menggunakan fungsi kernel $\mathrm{K}(\mathrm{x\_i}, \mathrm{x\_j}) =$

$\langle \phi(x\_i), \phi(x\_j) \rangle$ Tujuan utama adalah mencari fungsi keputusan f(x) yang memisahkan mayoritas data dari origin.

Fungsinya dapat dituliskan sebagai:

$$f(x) = \text{sign}\left(\sum \alpha_i K(x_i, x) - \rho\right) \quad (1)$$

di mana:

1. w adalah vektor bobot di ruang fitur,
2. ρ adalah *offset* dari hyperplane,
3. Fungsi *sign* menentukan apakah suatu titik data diklasifikasikan sebagai normal (positif) atau anomali (negatif).

Persoalan optimisasi disusun sebagai

$$\min_{w,\rho,\xi} \quad \frac{1}{2}|w|^2 + \frac{1}{\nu n}\sum_{i=1}^{n} \xi_i - \rho \quad (1)$$

$$\text{subject to: } \langle w, \phi(x_i)\rangle \geq \rho - \xi_i, \qquad \xi_i \geq 0 \quad (2)$$

Penelitian oleh Machado et al. (2022) menggunakan formulasi ini dalam sistem deteksi anomali sumur minyak, membuktikan bahwa pengaturan nilai γ dan ν secara adaptif mampu meningkatkan akurasi deteksi.

2.1.11 Evaluasi Model

Evaluasi model merupakan tahapan penting dalam pengembangan sistem berbasis *machine learning*, karena menentukan sejauh mana model yang dibangun dapat bekerja secara optimal dan akurat dalam lingkungan nyata. Pada sistem deteksi anomali, evaluasi dilakukan terhadap performa model dalam

mengklasifikasikan data normal dan anomali, terutama karena umumnya dataset bersifat tidak seimbang.

Menurut Kumar dan Soundarapandiyan (2024), evaluasi model anomali harus mempertimbangkan metrik yang memperhitungkan ketidakseimbangan kelas, karena data anomali sering kali jauh lebih sedikit dibandingkan data normal.

2.1.11.1 Confusion Matrix

Confusion matrix merupakan matriks dua dimensi yang menunjukkan jumlah klasifikasi benar dan salah berdasarkan label aktual dan prediksi model. Matriks ini terdiri dari empat komponen utama: *True Positive (TP)*, *True Negative (TN)*, *False Positive (FP)*, dan *False Negative (FN)*. Dalam konteks deteksi anomali, confusion matrix memberikan gambaran visual tentang bagaimana model membedakan data normal dari data anomali.

Yamazaki et al. (2024) menggunakan *confusion matrix* dalam evaluasi model deteksi intrusi menggunakan dataset CSE-CIC-IDS2018, yang memperlihatkan pentingnya evaluasi berdasarkan FP dan FN karena konsekuensi kesalahan deteksi sangat besar dalam sistem keamanan.

2.1.11.2 Precision

Precision (presisi) adalah metrik yang mengukur akurasi dari prediksi positif. Metrik ini menjawab pertanyaan: "Dari semua instans yang diprediksi oleh model sebagai anomali, berapa persen yang benar-benar merupakan anomali?".

Precision adalah *rasio* antara jumlah prediksi positif yang benar terhadap semua prediksi positif. Rumusnya adalah:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (1)$$

Precision tinggi menunjukkan bahwa model jarang salah dalam mendeteksi anomali, yang sangat penting dalam sistem *real-time* untuk menghindari *false alarm*. Dalam sistem log Fortigate, precision yang rendah dapat menyebabkan tim keamanan kewalahan menangani banyak alarm palsu.

Vadisetty dan Polamarasetti (2024) menekankan pentingnya precision dalam sistem deteksi *real-time* karena kesalahan prediksi dapat menimbulkan reaksi otomatis yang salah, seperti pemblokiran akses sah.

2.1.11.3 *Recall*

Recall adalah rasio antara jumlah prediksi positif yang benar terhadap semua kasus aktual positif. Rumusnya adalah:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{1}$$

Recall tinggi berarti model mampu menangkap sebagian besar anomali yang ada. Dalam sistem keamanan, recall penting karena kegagalan dalam mendeteksi anomali bisa menyebabkan kerugian besar. Namun, peningkatan *recall* sering kali berdampak pada penurunan precision, sehingga perlu keseimbangan.

2.1.11.4 *F1-Score*

Dalam banyak kasus, terdapat pertukaran (*trade-off*) antara *precision* dan *recall*. Meningkatkan salah satu metrik seringkali menyebabkan penurunan pada metrik lainnya. *F1-Score* diperkenalkan sebagai cara untuk menyeimbangkan kedua metrik tersebut. *F1-Score* adalah rata-rata harmonik (*harmonic mean*) dari *precision* dan *recall*, yang memberikan bobot yang sama pada keduanya. Metrik ini sangat berguna untuk dataset yang tidak seimbang karena ia akan bernilai rendah jika salah satu dari *precision* atau *recall* bernilai rendah. Nilai *F1-Score* yang tinggi

menunjukkan bahwa model memiliki keseimbangan yang baik antara tidak menghasilkan banyak alarm palsu dan tidak melewatkan banyak ancaman nyata (Abdullatif & Al-Sartawi, 2023). *F1-Score* adalah rata-rata harmonis dari *precision* dan *recall*, memberikan keseimbangan antara keduanya. Rumusnya:

$$F1\text{-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (1)$$

F1-Score ideal digunakan saat distribusi kelas tidak seimbang. Kadri dan Singla (2025) menampilkan bahwa *F1-score* memberikan gambaran paling adil terhadap performa model dalam kondisi data langka seperti anomali jaringan.

2.1.11.5 ROC Curve dan AUC (Area Under the Curve)

ROC (*Receiver Operating Characteristic*) *curve* adalah grafik antara *True Positive Rate* (TPR) dan False Positive Rate (FPR). AUC adalah area di bawah kurva tersebut, dan digunakan untuk menilai performa klasifikasi secara menyeluruh. Nilai AUC mendekati 1 menunjukkan model sangat baik dalam membedakan dua kelas.

Matsuo et al. (2025) mengembangkan pendekatan *AUC-based anomaly scoring* untuk meningkatkan performa deteksi berbasis jaringan saraf dalam konteks pembelajaran positif-negatif tidak seimbang, dengan hasil ROC yang lebih stabil terhadap outlier.

2.1.12 *Tools Website* dan Sistem Integrasi

Dalam pengembangan sistem deteksi anomali berbasis *machine learning*, penggunaan platform pendukung seperti *notebook environment*, *deployment framework*, serta version control sangat berperan dalam membangun alur kerja yang

terstruktur, kolaboratif, dan mudah diakses. Sistem yang baik tidak hanya bergantung pada performa model, tetapi juga pada seberapa efisien model tersebut diintegrasikan dalam sistem operasional nyata. Oleh karena itu, pemilihan *tools* seperti Jupyter Notebook, GitHub, dan Streamlit memberikan fondasi penting dalam *end-to-end deployment* aplikasi keamanan berbasis data log.

2.1.12.3 Alur Kerja Sistem End-to-End

Alur kerja sistem secara umum dimulai dari eksplorasi dan pemrosesan data log menggunakan Jupyter Notebook. *Platform* ini menyediakan lingkungan interaktif untuk menulis kode, melakukan eksperimen, serta dokumentasi teknis dalam satu antarmuka yang mudah ditelusuri. Jupyter memungkinkan visualisasi hasil evaluasi model secara instan, sehingga sangat cocok untuk iterasi dan validasi cepat selama fase pengembangan.

Menurut Syed et al. (2024), Jupyter Notebook membantu mempercepat proses pelatihan dan evaluasi model pada proyek *cyber threat intelligence*, karena mendukung format *reproducible research* dan dapat diintegrasikan langsung dengan pustaka seperti Scikit-learn, Pandas, dan TensorFlow.

Selanjutnya, kode proyek dan model disimpan serta dilacak menggunakan GitHub Repository. GitHub tidak hanya berfungsi sebagai tempat penyimpanan versi, tetapi juga memfasilitasi kolaborasi antar tim, *issue tracking*, serta integrasi berkelanjutan (*CI/CD*) melalui GitHub Actions. Dengan menyimpan seluruh proses pipeline di GitHub, pengembangan sistem dapat dilakukan secara transparan, terstruktur, dan dapat direplikasi oleh tim lain.

Untuk visualisasi dan penyajian antarmuka pengguna, sistem di-*deploy* menggunakan Streamlit, sebuah framework *open-source* berbasis Python yang ringan dan cepat untuk membangun aplikasi web berbasis data sains. Pengguna dapat melakukan *upload* log, menjalankan proses deteksi anomali, serta melihat hasil evaluasi melalui visualisasi metrik seperti confusion matrix, ROC curve, dan grafik distribusi log. Raghavendran dan Elragal (2023) menunjukkan bahwa Streamlit menjadi solusi efektif dalam *prototyping AI tools* di sektor publik karena sifatnya yang *interactive* dan dapat diintegrasikan dengan backend analitik secara langsung.

Dengan integrasi ketiga komponen ini—Jupyter untuk pengembangan dan eksperimen, GitHub untuk manajemen proyek, dan Streamlit untuk antarmuka pengguna—sistem deteksi anomali berbasis log Fortigate menjadi sistem yang lengkap dan siap diterapkan dalam lingkungan produksi perusahaan.

2.2 Kajian Penelitian Terdahulu

Setiap penelitian ilmiah dibangun di atas fondasi pekerjaan yang telah dilakukan oleh para peneliti sebelumnya. Oleh karena itu, tinjauan pustaka atau kajian penelitian terdahulu merupakan langkah fundamental yang bertujuan untuk memetakan lanskap pengetahuan yang ada terkait topik penelitian. Tujuan dari kajian ini adalah untuk mengidentifikasi teori-teori kunci, metodologi yang relevan, temuan-temuan penting, serta kesenjangan (*gaps*) dalam penelitian yang ada. Dengan memahami keadaan terkini (*state-of-the-art*), peneliti dapat memposisikan

kontribusi studinya secara akurat dan menunjukkan kebaruan (*novelty*) dari pendekatan yang diusulkan (Snyder, 2019).

Dalam bab ini, disajikan rangkuman dari sejumlah penelitian terdahulu yang relevan dengan deteksi anomali pada log keamanan jaringan menggunakan *machine learning*. Kajian ini mencakup studi yang berfokus pada penggunaan *Autoencoder* dan *One-Class SVM*, integrasi dengan sistem seperti Wazuh dan Fortigate, serta penerapan metodologi serupa dalam konteks keamanan siber. Setiap penelitian dianalisis berdasarkan tujuan, metodologi yang digunakan, temuan utama, dan kesimpulannya. Analisis ini tidak hanya memberikan konteks bagi penelitian saat ini tetapi juga membantu memvalidasi pemilihan metode dan pendekatan yang diadopsi.

2.2.1 Paper 1: *Syscall Security Components*, HP Léo, dipublikasikan di Repositorio Aberto, University of Porto, 2024. Klasifikasi jurnal: Repositori akademik terakreditasi universitas Eropa.

2.2.1.1 Tujuan Penelitian

Penelitian ini bertujuan untuk merancang dan mengimplementasikan sistem deteksi anomali keamanan berbasis syscall monitoring pada infrastruktur log terdistribusi. Penelitian mengintegrasikan Wazuh sebagai agen keamanan untuk pengumpulan data log dan menerapkan pendekatan berbasis pembelajaran mesin dalam mendeteksi anomali sistem.

2.2.1.2 Metodologi yang Digunakan

Penulis menggunakan pendekatan *machine learning* berbasis *Autoencoder* untuk melakukan *unsupervised anomaly detection*. Wazuh diintegrasikan sebagai

komponen pengumpulan log, sementara data syscall diproses untuk pelatihan model Autoencoder dalam mengenali aktivitas sistem yang menyimpang.

2.2.1.3 Temuan Utama

Hasil penelitian menunjukkan bahwa model Autoencoder yang dilatih hanya dengan data normal mampu mendeteksi penyimpangan perilaku sistem dengan akurasi di atas 92%. Sistem juga berhasil mendeteksi anomali yang tidak dikenali oleh sistem signature-based seperti ClamAV. Efektivitas model meningkat saat log dikonversi menjadi representasi numerik yang dinormalisasi.

2.2.1.4 Kesimpulan Penelitian

Studi ini menyimpulkan bahwa Autoencoder dapat digunakan sebagai alat pelengkap untuk sistem deteksi berbasis signature dengan akurasi tinggi, dan integrasi dengan Wazuh mendukung otomasi pengumpulan dan manajemen data log.

2.2.2 Paper 2: *Anomaly Detection Through User Behaviour Analysis*, V. Dumitrasc, dipublikasikan di Universitat Politècnica de Catalunya (UPC), 2023. Klasifikasi jurnal: Repositori TFM terakreditasi EHEA (European Higher Education Area).

2.2.2.1 Tujuan Penelitian

Penelitian ini ditujukan untuk mengembangkan sistem deteksi perilaku mencurigakan pengguna melalui pendekatan berbasis analitik log dari sistem Wazuh. Tujuannya adalah memetakan perilaku normal pengguna dan mengidentifikasi outlier sebagai anomali.

2.2.2.2 Metodologi yang Digunakan

Penelitian memanfaatkan Autoencoder dan *One-Class* SVM secara terpisah dan gabungan. Data log dikumpulkan menggunakan agen Wazuh dan diekstrak untuk fitur perilaku seperti waktu *login*, alamat IP, dan frekuensi akses.

2.2.2.3 Temuan Utama

Model Autoencoder menunjukkan performa deteksi anomali sebesar 94%, sementara One-Class SVM berkinerja lebih tinggi untuk data berdimensi rendah. Penggabungan keduanya menghasilkan sistem hybrid yang mampu *menekan false positives* tanpa mengorbankan *recall*.

2.2.2.4 Kesimpulan Penelitian

Studi ini menunjukkan bahwa pendekatan *hybrid* mampu mengatasi keterbatasan masing-masing algoritma, dan Wazuh dapat digunakan sebagai *platform* efektif dalam pengumpulan data perilaku.

2.2.3 Paper 3: *Survey on Unified Threat Management (UTM) Systems for Home Networks*, Siddiqui et al., dipublikasikan di IEEE Communications Surveys & Tutorials, 2024. Klasifikasi jurnal: Q1 (Scopus), IEEE.

2.2.3.1 Tujuan Penelitian

Tujuan utama studi ini adalah memberikan tinjauan komprehensif terhadap *Unified Threat Management* (UTM), termasuk integrasi deteksi anomali dalam firewall seperti Fortigate.

2.2.3.2 Metodologi yang Digunakan

Penulis melakukan *systematic literature review* terhadap sistem UTM dan mengevaluasi integrasi log DNS, HTTP, dan NetFlow menggunakan algoritma berbasis deteksi statistik dan pembelajaran mesin.

2.2.3.3 Temuan Utama

Penelitian ini menemukan bahwa pendekatan *anomaly detection* berbasis model hybrid seperti Autoencoder dan SVM sangat diperlukan dalam konteks perangkat rumahan. Fortigate dapat digunakan sebagai *source data* untuk pipeline log anomali.

2.2.3.4 Kesimpulan Penelitian

Studi ini menyimpulkan pentingnya integrasi log multi-sumber ke dalam sistem UTM untuk meningkatkan visibilitas ancaman, termasuk integrasi Fortigate ke dalam pipeline analitik.

2.2.4 Paper 4: *Deteksi Anomali Jaringan Menggunakan Isolation Forest pada Log Wazuh dengan Pemberitahuan WhatsApp di PT XYZ*, R.P. Dewa & W. Windarto, dipublikasikan di KRESNA: Jurnal Riset dan Inovasi Teknik Informatika, Universitas Budi Luhur, 2024. Klasifikasi jurnal: SINTA 5 (Indonesia).

2.2.4.1 Tujuan Penelitian

Penelitian ini bertujuan untuk mengembangkan sistem deteksi anomali jaringan berbasis log dari Wazuh dengan penerapan algoritma *Isolation Forest*. Tujuan utamanya adalah mendeteksi aktivitas tidak biasa dalam sistem dan mengirimkan notifikasi otomatis kepada administrator melalui WhatsApp.

2.2.4.2 Metodologi yang Digunakan

Data log dikumpulkan dari Wazuh dan diekstrak menggunakan pipeline berbasis Python. Penelitian menggunakan algoritma *unsupervised Isolation Forest* untuk mendeteksi penyimpangan dari pola perilaku normal dalam log. Selain itu, hasil anomali dikonversi menjadi pesan teks dan dikirim ke WhatsApp API melalui skrip Python.

2.2.4.3 Temuan Utama

Model mampu mendeteksi serangkaian aktivitas yang tidak lazim seperti login dari IP asing atau penggunaan port tidak standar. Dengan menggunakan teknik *ensemble* yang mendasari *Isolation Forest*, hasil deteksi menunjukkan akurasi cukup tinggi dalam memisahkan outlier log dari aktivitas reguler. Integrasi notifikasi mempercepat proses mitigasi oleh tim IT.

2.2.4.4 Kesimpulan Penelitian

Penelitian menyimpulkan bahwa algoritma berbasis *tree ensemble* seperti Isolation Forest cocok untuk data log Wazuh dan notifikasi otomatis dapat meningkatkan efektivitas sistem keamanan internal perusahaan.

2.2.5 Paper 5: *ML-Driven Log Analysis for Real-Time Cyber Threat Detection in Security Operation Centers*, S.A. Chamkar et al., dipublikasikan di *Preprints*, 2025. Klasifikasi: Pra-cetak akademik dari penelitian Universitas Maroko.

2.2.5.1 Tujuan Penelitian

Studi ini bertujuan untuk meningkatkan *sistem Security Operation Center* (SOC) dengan menggunakan AI dalam menganalisis log keamanan berbasis Wazuh untuk mendeteksi anomali secara *real-time.*

2.2.5.2 Metodologi yang Digunakan

Data log dikumpulkan dari berbagai endpoint dan dianalisis menggunakan pendekatan K-Nearest Neighbors (KNN) *Classifier* Penulis juga memanfaatkan *Elastic Stack* untuk penyimpanan dan visualisasi data log.

2.2.5.3 Temuan Utama

Model yang dikembangkan berhasil mengidentifikasi perubahan mendadak dalam pola log jaringan yang mengindikasikan serangan. Selain itu, sistem real-time memberikan peringatan dini kepada operator SOC dengan sensitivitas tinggi.

2.2.5.4 Kesimpulan Penelitian

Studi ini menunjukkan bahwa integrasi antara AI dan log framework seperti Wazuh dan ELK Stack dapat meningkatkan kapasitas deteksi anomali jaringan secara otomatis dan real-time.