Implementasi *Middleware* Deteksi Serangan Siber Berbasis *Deep learning* pada Endpoint API

Proposal Tugas Akhir

Kelas MK Penulisan Proposal (CAK4EAB2)

2211102308

Alfanah Muhson Husain Nugroho

Program Studi Sarjana Informatika

Direktorat Kampus Purwokerto

Universitas Telkom

Purwokerto

2025

Lembar Persetujuan

Implementasi *Middleware* Deteksi Serangan Siber Berbasis *Deep learning* pada Endpoint API

Implementation of *Deep learning*-Based Cyber Attack Detection *Middleware* on API Endpoint

NIM : 2211102308

Alfanah Muhson Husain Nugroho

Proposal ini diajukan sebagai usulan pembuatan tugas akhir pada
Program Studi Sarjana Informatika
Direktorat Kampus Purwokerto Universitas Telkom

Purwokerto, 28 Mei 2025
Menyetujui

Calon Pembimbing 1	Calon Pembimbing 2
M. Agung Nugroho, S.Kom., M.Kom.	Yesy Diah R, S.Kom., M.Kom
25850001o	238503339

DAFTAR ISI

Lembar Persetujuan 1
DAFTAR ISI 2
ABSTRAK 3
1. PENDAHULUAN 4
1.1. Latar Belakang 4
1.2. Perumusan Masalah 5
1.3. Tujuan 6
1.4. Batasan Masalah 6
1.5. Rencana Kegiatan 6
1.6. Jadwal Kegiatan 8
2. KAJIAN PUSTAKA 9
2.1. Tinjauan Pustaka 9
2.2. Dasar Teori 14
2.2.1 Serangan siber 14
2.2.2 Intrusion Detection System 15
2.2.3 *Deep learning* 17
2.2.4 Deep Neural Network 17
2.2.5 Application Programming Interface 20
2.2.6 *Middleware* 21
3. PERANCANGAN SISTEM 22
3.1 Gambaran Umum Sistem 22
3.2 Data Set dan Pengumpulan Data 23
3.3 Pembangunan Model 24
3.3.1 *Pre-processing* data 24
3.3.2 Pemodelan IDS 33
3.4 Evaluasi IDS 35
3.5 Skenario Penerapan dan Pengujian *Middleware* IDS 37
DAFTAR PUSTAKA 39
LAMPIRAN 44

ABSTRAK

Pertumbuhan lalu lintas jaringan dan meningkatnya serangan siber seperti DDoS, brute force, dan infiltrasi telah menunjukkan bahwa sistem keamanan konvensional seperti firewall tidak lagi cukup untuk memberikan proteksi menyeluruh. Masalah utama yang dihadapi adalah keterbatasan model deteksi tradisional dalam mengenali pola serangan kompleks dan kemampuan integrasi ke dalam sistem yang berjalan secara real-time. Penelitian ini bertujuan untuk mengembangkan sistem deteksi intrusi (Intrusion Detection System/IDS) berbasis *Deep Neural Network* (DNN) yang dapat diintegrasikan langsung ke dalam middleware API guna menyaring permintaan jaringan sebelum mencapai endpoint utama. Metodologi yang diterapkan meliputi pra-pemrosesan data, pelatihan model DNN, serta integrasi model ke middleware API. Dataset yang digunakan adalah CIC-IDS2017, yang mencakup beragam jenis serangan jaringan dan telah digunakan secara luas dalam penelitian IDS. Sistem akan diuji dalam skenario simulasi serangan menggunakan berbagai tools seperti hping3 dan slowloris untuk mengevaluasi akurasi deteksi, efisiensi pemrosesan, serta dampak terhadap latensi layanan. Hipotesis awal dari penelitian ini menyatakan bahwa integrasi model DNN ke dalam middleware akan menghasilkan deteksi serangan yang lebih cepat dan akurat. Pendekatan ini diharapkan mampu memperkuat sistem keamanan jaringan secara real-time dan adaptif terhadap berbagai ancaman siber.

Kata Kunci: *deep neural network*, *intrusion detection*, *middleware*, API, CIC-IDS2017, keamanan jaringan

.

1. PENDAHULUAN

1.1. Latar Belakang

Laporan statistik Internet global mengungkapkan bahwa pertumbuhan Internet pada periode 2000-2019 mencapai 1.114%, dengan lebih dari 2 *kuintiliun byte* data yang dihasilkan setiap harinya. Hal ini menunjukkan bahwa laju pertumbuhan data dari berbagai sumber meningkat dengan sangat cepat. Di saat yang sama, perkembangan alat dan metode peretasan juga mengalami kemajuan yang pesat [1]. Sepanjang tahun 2024, sistem *Honeynet* BSSN mencatat lebih dari 609 juta serangan siber yang berasal dari 1.232.072 alamat IP, dengan Indonesia menjadi sumber serangan tertinggi sebanyak 158 juta serangan [2]. Selain itu, tercatat pula 1,4 juta serangan *malware* dengan 7.592 jenis *malware* unik, yang sebagian besar menargetkan sektor-sektor penting seperti pemerintahan dan keuangan [2]. Oleh karena itu, diperlukan langkah-langkah keamanan informasi dan analisis data yang efektif untuk melindungi data dari ancaman penyusupan [1].

Teknik pengamanan menggunakan *firewall* tradisional memiliki keterbatasan dalam mendeteksi serangan anomali [3]. Sistem berbasis perilaku tradisional bergantung pada profil standar yang sulit ditentukan karena meningkatnya kompleksitas jaringan dan aplikasi. Akibatnya, sistem tersebut mungkin kurang efektif dalam mendeteksi anomali [4]. Oleh karena itu, *Intrusion Detection System* (IDS) berbasis *Deep learning* menjadi solusi yang semakin banyak dikembangkan [5].

Sebagai perkembangan terkini, pendekatan *Deep learning* menawarkan keunggulan signifikan dibandingkan metode tradisional maupun algoritma *Machine Learning* (ML) klasik [5]. *Deep learning* dapat mengatasi beberapa keterbatasan dari pembelajaran dangkal (*shallow learning*) seperti tingginya tingkat interaksi yang diperlukan dari para ahli. Pengetahuan dari pakar dibutuhkan untuk memproses data, misalnya dalam mengidentifikasi data dan pola yang berguna. Proses ini tidak hanya memakan banyak tenaga dan biaya, tetapi juga rentan terhadap kesalahan. Hingga saat ini, *deep learning* telah menunjukkan bahwa pembelajaran fitur secara bertingkat yang dimilikinya mampu memberikan performa yang lebih baik atau setidaknya setara dengan teknik pembelajaran

dangkal. *Deep learning* mampu memfasilitasi analisis data jaringan yang lebih mendalam serta identifikasi anomali yang lebih cepat [6].

Akan tetapi sebagian besar penelitian IDS berbasis *Deep Learning* masih berfokus pada pengujian secara offline, dan belum banyak yang mengintegrasikan model deteksi anomali secara langsung ke dalam sistem *middleware* sebagai lapisan awal pengamanan permintaan jaringan. Hal ini menunjukkan adanya celah penelitian yang dapat dieksplorasi lebih lanjut melalui pendekatan integratif antara ML dan *middleware* API.

Pada penelitian ini, algoritma yang akan digunakan adalah *Deep Neuron Network* (DNN). DNN mampu mempelajari representasi fitur tingkat tinggi secara otomatis dari data mentah melalui lapisan-lapisan neuron bertingkat sehingga dapat mengenali pola-pola rumit dan variatif dalam trafik jaringan serta perilaku pengguna. Algoritma tersebut akan digunakan untuk melatih sebuah model IDS menggunakan dataset CIC-IDS2017, yang secara luas digunakan dalam penelitian IDS karena mencakup berbagai jenis serangan nyata yang mencerminkan pola serangan terkini, sehingga memberikan lingkungan evaluasi yang representatif bagi model pembelajaran mesin.

Untuk memastikan model dapat beroperasi secara optimal dalam lingkungan jaringan modern, model IDS yang dihasilkan akan diimplementasikan dalam bentuk *middleware* API. *Middleware* ini akan berfungsi sebagai lapisan pengaman tambahan yang secara otomatis menganalisis setiap permintaan (request) yang masuk sebelum diteruskan ke sistem utama. Dengan pendekatan ini, serangan dapat dideteksi dan ditangani lebih cepat tanpa mengganggu performa layanan utama, sehingga meningkatkan ketahanan sistem terhadap ancaman siber.

1.2. Perumusan Masalah

Berdasarkan latar belakang tersebut, penelitian ini berfokus pada:

1. Bagaimana merancang dan mengembangkan model *Deep Neural Network* (DNN) yang efektif dalam mendeteksi dan menahan berbagai jenis serangan siber berdasarkan analisis pola trafik jaringan?
2. Bagaimana kinerja (akurasi deteksi) model *Deep Neural Network* (DNN) ketika diintegrasikan secara langsung ke dalam middleware API untuk

berfungsi sebagai sistem deteksi intrusi secara real-time terhadap permintaan jaringan?

1.3. Tujuan

Adapun tujuan dari penulisan TA ini adalah sebagai berikut:

1. Mengembangkan model *Deep Neural Network* (DNN) yang mampu mendeteksi dan menahan berbagai jenis serangan siber secara efektif berdasarkan pola trafik jaringan.
2. Mengevaluasi performa model DNN yang diintegrasikan dalam *middleware* API dalam hal akurasi deteksi.

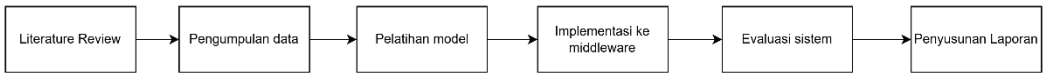
1.4. Batasan Masalah

Batasan masalah merupakan batasan-batasan pembahasan yang akan peneliti masukkan di dalam TA ini. Batasan masalah yang ada adalah sebagai berikut:

1. Dataset yang digunakan adalah CIC-IDS2017.
2. *Middleware* API yang dikembangkan hanya berfungsi untuk mendeteksi dan memitigasi serangan pada permintaan yang masuk sebelum diteruskan ke sistem utama.
3. Evaluasi kinerja menggunakan *matrix evaluation*
4. Model deteksi difokuskan untuk mendeteksi serangan yang memanipulasi lalu lintas jaringan, seperti DDoS, DoS, Port Scanning dan Brute Force.
5. Pengujian dilakukan dalam lingkungan simulasi yang merepresentasikan jaringan modern tanpa implementasi langsung pada sistem produksi.

1.5. Rencana Kegiatan

Pada bagian ini akan dijelaskan secara rinci tahapan-tahapan kegiatan yang direncanakan dalam penelitian ini. Rencana kegiatan disusun untuk memastikan setiap proses berjalan sistematis dan terukur, mulai dari studi literatur, pengumpulan data, pelatihan model hingga implementasi ke *middleware* (lihat Gambar 1.1).



Gambar 1. 1 Alur rencana kegiatan

1. **Literature Review**

   Tahap ini mencakup penelusuran dan kajian terhadap artikel ilmiah mengenai sistem deteksi serangan siber berbasis deep learning, khususnya penggunaan DNN dan dataset CIC-IDS2017. Tujuannya adalah memperoleh pemahaman yang kuat sebagai dasar pengembangan sistem. Targetnya adalah terkumpulnya minimal 30 artikel relevan.

2. **PengumpulanData**

   Dataset CIC-IDS2017 dikumpulkan dalam format .pcap, kemudian dibersihkan dan disiapkan untuk proses pelatihan. Hasil yang diharapkan adalah dataset yang terstruktur dan siap pakai.

3. **Pelatihan Model**

   Data diproses, lalu digunakan untuk melatih model Isolation Forest dalam mendeteksi anomali. Model divalidasi menggunakan metrik seperti akurasi dan F1-score. Target performa minimal adalah akurasi 80%.

4. **Implementasi ke Middleware**

   Model yang telah dilatih diintegrasikan ke dalam middleware API untuk mendeteksi anomali secara real-time pada setiap request masuk. Middleware diuji agar berjalan stabil dan cepat.

5. **Evaluasi Sistem**

   Middleware diuji dengan skenario nyata menggunakan tools simulasi serangan. Evaluasi mencakup akurasi, F1-score, dan latensi sistem. Targetnya akurasi >90% dan inferensi <100 ms.

6. **Penyusunan Laporan**

   Seluruh proses didokumentasikan dalam buku tugas akhir sesuai format akademik. Laporan ini menjadi bukti validitas penelitian dan layak diuji oleh pihak kampus.

Dengan mengikuti rencana kegiatan yang telah disusun secara terstruktur ini, diharapkan penelitian dapat berjalan efektif dan efisien sehingga menghasilkan model deteksi serangan siber yang akurat dan dapat diimplementasikan secara real-time melalui *middleware*.

1.6. Jadwal Kegiatan

Berikut ini disajikan jadwal kegiatan yang menggambarkan alokasi waktu pelaksanaan setiap tahapan dalam penelitian ini.

Tabel 1. 1 Jadwal kegiatan

Kegiatan	**Bulan**					
	1	2	3	4	5	6
Literature review						
Pengumpulan data						
Pelatihan model						
Implementasi ke *middleware*						
Evaluasi Sistem						
Penyusunan Laporan						

*Keterangan: shading warna *grayscale*

Dengan adanya jadwal kegiatan yang terstruktur ini, diharapkan seluruh tahapan penelitian dapat berjalan sesuai rencana sehingga target-target yang telah ditetapkan dapat tercapai secara optimal.

2. KAJIAN PUSTAKA

2.1. Tinjauan Pustaka

Penelitian mengenai sistem deteksi intrusi (Intrusion Detection System/IDS) terus berkembang, khususnya dalam integrasi teknologi pembelajaran mesin (Machine Learning/ML) dan pembelajaran mendalam (Deep Learning/DL) untuk meningkatkan akurasi dan efisiensi deteksi. Selain itu, muncul kebutuhan untuk mengimplementasikan IDS secara langsung dalam arsitektur sistem, seperti middleware atau endpoint API, guna mendeteksi dan mencegah serangan secara real-time. Tinjauan pustaka ini merangkum berbagai studi yang relevan dengan pengembangan IDS berbasis ML/DL serta implementasinya dalam berbagai lingkungan sistem, termasuk jaringan IoT, middleware berbasis blockchain, dan sistem distribusi data. Fokus utama diberikan pada studi-studi yang memanfaatkan dataset publik seperti CICIDS-2017 serta pendekatan model seperti CNN, RNN, LSTM, dan DNN. Tabel berikut menyajikan ringkasan studi terkait, metode yang digunakan, relevansinya dengan topik penelitian ini, serta identifikasi gap sebagai dasar penyusunan kerangka penelitian lanjutan.

No	**Judul Penelitian (dengan sitasi)**	**Studi Kasus**	**Metode yang Digunakan**	**Relevansi**	**Gap**
1	Yin et al. – *A Deep Learning Approach for Intrusion Detection Using RNN* [5]	Deteksi intrusi pada dataset KDD dan NSL-KDD	Menggunakan Recurrent Neural Network (RNN) untuk menangkap urutan temporal dalam data lalu lintas jaringan dan klasifikasi jenis serangan.	Memberikan dasar penggunaan deep learning dalam pendeteksian serangan jaringan.	Tidak mengintegrasikan model ke dalam sistem middleware atau endpoint secara real-time.

No	**Judul Penelitian (dengan sitasi)**	**Studi Kasus**	**Metode yang Digunakan**	**Relevansi**	**Gap**
2	Wu & Guo. *LuNet: A Deep Neural Network for Network Intrusion Detection* [18]	Deteksi intrusi menggunakan dataset CICIDS-2017	Kombinasi CNN (untuk ekstraksi fitur spasial) dan RNN (untuk analisis temporal) dalam satu arsitektur hybrid bernama LuNet.	Relevan dalam mendeteksi pola serangan kompleks berbasis urutan waktu.	Tidak menyebutkan integrasi model dalam sistem API/middleware secara langsung.
3	Munawarah et al. – *Deep Neural Network for DDoS Detection in IoT* [19]	Deteksi DDoS dalam sistem IoT	Pelatihan Deep Neural Network (DNN) dengan parameter tuning untuk klasifikasi traffic DDoS.	Fokus pada DDoS dan konteks IoT, serupa dengan kebutuhan API/middleware.	Belum diuji pada skenario integrasi langsung di middleware atau endpoint.
4	Bororing – *Evaluasi Metode Machine Learning untuk Deteksi Anomali pada Lalu Lintas Jaringan* [20]	Dataset CICIDS-2017 untuk deteksi anomali	Evaluasi beberapa algoritma ML (SVM, Decision Tree, Naïve Bayes) untuk klasifikasi serangan dan benign.	Menyediakan evaluasi awal terhadap efektivitas model ML dalam dataset yang sama.	Tidak fokus pada middleware dan tidak membahas integrasi real-time.
6	Sahren & Lubis – *Intrusion Detection System Berbasis Deep Learning Untuk Peningkatan Mitigasi Sql Injection Dan Syn Flood Attack* [21]	Deteksi serangan berbasis analisis data log jaringan dari dataset publik CICDDoS2019 dan CSE-CIC-IDS2018	CNN (VGG-16, ResNet50, InceptionV3) digunakan untuk klasifikasi serangan; data log diproses melalui konversi atribut, normalisasi, dan augmentasi fitur.	Performa tinggi dalam mendeteksi serangan SQL Injection dan SYN Flood; model deep learning menunjukkan akurasi hingga >99%.	Tidak kompatibel langsung dengan sistem API berbasis text-log atau packet-level.

No	**Judul Penelitian (dengan sitasi)**	**Studi Kasus**	**Metode yang Digunakan**	**Relevansi**	**Gap**
7	Silva et al. – *MADCS: A Middleware for Anomaly Detection and Content Sharing for Blockchain-Based Systems* [10]	Deteksi anomali dalam blockchain menggunakan middleware khusus	Mengembangkan middleware khusus untuk mendeteksi anomali pada ledger blockchain dengan rule-based dan analisis pola.	Menunjukkan integrasi IDS dalam middleware berbasis domain khusus (blockchain).	Tidak menggunakan ML/DL, sehingga tidak optimal untuk generalisasi pola kompleks.
8	Asjad – *Intrusion Detection and Cyber Attack Classification for Encrypted DDS Communication Middleware in OT Networks Using Machine Learning* [22]	Sistem Distributed Data Service (DDS)	Implementasi model DL (LSTM) ke dalam sistem middleware DDS untuk memantau dan mendeteksi aktivitas mencurigakan.	Mendemonstrasikan integrasi IDS ke middleware berbasis DDS.	Lingkup terbatas pada DDS; tidak menggeneralisasi ke REST API atau endpoint modern.
9	Bhandari et al. – *Distributed Deep Neural-Network-Based Middleware for Cyber-Attacks Detection in Smart IoT Ecosystem* [7]	Deteksi ancaman IoT menggunakan distributed learning	Model DNN dilatih dan dijalankan pada node IoT terdistribusi, fokus pada kecepatan dan efisiensi.	Menunjukkan pendekatan DL yang efisien dan bisa diadaptasi untuk edge/middleware.	Tidak membahas middleware API secara eksplisit.
10	Bhandari et al. – *Middleware-level Intrusion Detection with Edge-AI for IoT*	Integrasi IDS ke middleware IoT edge	Edge AI digunakan untuk deteksi serangan langsung pada level middleware dengan pendekatan ringan.	Relevan untuk sistem edge berbasis middleware API.	Fokus masih pada IoT; tidak diuji pada API konvensional (web/HTTP).

No	**Judul Penelitian (dengan sitasi)**	**Studi Kasus**	**Metode yang Digunakan**	**Relevansi**	**Gap**
11	Tambi & Singh – *A New Framework and Performance Assessment Method for Distributed Deep Neural Network-Based Middleware for Cyberattack Detection in the Smart IoT Ecosystem* [23]	Keamanan middleware edge dalam IoT	Pengembangan arsitektur DNN ringan untuk implementasi real-time pada edge-middleware.	Menunjukkan bahwa DL bisa diterapkan dalam middleware dengan keterbatasan resource.	Studi masih konseptual, tanpa simulasi nyata pada REST API berbasis publik.
12	Ali et al. – *Secure Middleware Model For Public Restful Apis* [9]	Middleware API untuk web service	Menggunakan rules dinamis untuk menyaring request berbahaya berdasarkan pola dan reputasi IP.	Fokus pada keamanan API middleware secara langsung.	Tidak menggunakan machine learning atau deep learning.
14	Liu & Lang – *Machine learning and deep learning methods for intrusion detection systems: A survey* [24]	Literatur tentang DL untuk IDS	Ulasan sistematis tentang metode DL seperti RNN, CNN, AE, LSTM dalam konteks IDS.	Referensi menyeluruh untuk pendekatan DL dan arsitektur dalam IDS.	Tidak spesifik pada middleware atau API.

Berdasarkan tinjauan pustaka, penggunaan algoritma machine learning dan deep learning telah menunjukkan hasil yang menjanjikan dalam sistem deteksi intrusi (IDS), khususnya dalam meningkatkan akurasi deteksi dan kemampuan identifikasi pola serangan yang kompleks. Sejumlah penelitian telah berhasil mengimplementasikan model-model seperti CNN, RNN, dan DNN pada sistem IDS menggunakan dataset seperti CICIDS-2017. Selain itu, pendekatan integrasi keamanan ke dalam middleware seperti pada MADCS dan

sistem berbasis blockchain membuktikan bahwa penyisipan mekanisme proteksi secara langsung pada jalur komunikasi data mampu memperkuat sistem dari sisi arsitektur. Namun demikian, masih terdapat gap signifikan terkait penerapan arsitektur deep learning, khususnya DNN, yang diintegrasikan secara langsung ke dalam middleware endpoint API untuk mendeteksi serangan secara real-time. Sebagian besar pendekatan saat ini masih memisahkan proses deteksi dan perlindungan, atau menggunakan teknik ML konvensional tanpa eksplorasi mendalam terhadap efektivitas arsitektur DNN di lingkungan middleware. Oleh karena itu, penelitian ini diarahkan untuk mengisi celah tersebut dengan mengembangkan dan menguji model DNN yang tertanam dalam middleware API.

2.2. Dasar Teori

2.2.1 Serangan siber

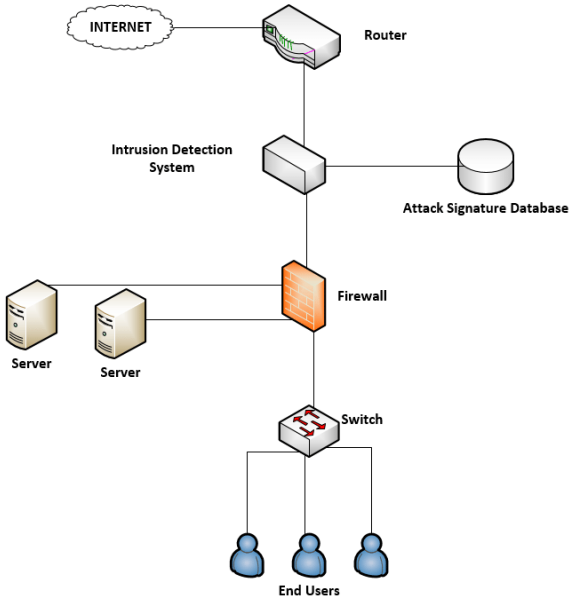
Serangan siber merupakan tindakan yang disengaja untuk mengakses sistem atau data tanpa izin dengan tujuan mencuri, mengubah, merusak, atau menghancurkan informasi. Pelaku serangan ini bisa berasal dari individu, kelompok kriminal, hingga aktor negara yang memiliki beragam motif, seperti keuntungan finansial, kepentingan politik, atau alasan pribadi. Dampak yang ditimbulkan tidak hanya merugikan secara ekonomi, tetapi juga dapat mengganggu operasional organisasi, di mana rata-rata biaya akibat pelanggaran data mencapai angka yang sangat tinggi.

Terdapat berbagai teknik yang digunakan dalam serangan siber, di antaranya malware, rekayasa sosial, serangan *denial-of-service* (DoS), peretasan akun, serta serangan *man-in-the-middle*. *Malware* seperti *trojan*, *ransomware,* dan *spyware* dapat menyebabkan kerusakan sistem serta pencurian data sensitif. Teknik rekayasa sosial, seperti *phishing*, memanipulasi korban agar memberikan informasi penting secara sukarela. Sementara itu, serangan DoS dan DDoS berupaya melumpuhkan sistem dengan membanjiri lalu lintas jaringan. Selain itu, serangan terhadap rantai pasokan dan eksploitasi celah keamanan *zero-day* juga sering dimanfaatkan oleh peretas untuk mengakses sistem tanpa terdeteksi.

Target utama serangan siber mencakup data keuangan, informasi pelanggan, kredensial *login*, hingga infrastruktur penting. Dampak dari serangan ini dapat berupa pencurian data, gangguan operasional, bahkan kerugian reputasi yang signifikan. Beberapa insiden besar, seperti serangan *ransomware* pada *Colonial Pipeline*, telah menunjukkan bagaimana serangan siber dapat mengganggu kehidupan sehari-hari, termasuk pasokan bahan bakar di suatu wilayah. Oleh sebab itu, meningkatkan keamanan siber menjadi hal yang sangat penting guna melindungi individu maupun organisasi dari ancaman yang terus berkembang.

2.2.2 Intrusion Detection System

Intrusion Detection System (IDS) merupakan sistem yang berfungsi untuk mendeteksi berbagai bentuk intrusi atau aktivitas mencurigakan pada jaringan. Sistem ini bekerja dengan cara memantau lalu lintas jaringan dan mengidentifikasi aktivitas yang bersifat mencurigakan maupun pelanggaran kebijakan keamanan, sehingga memungkinkan administrator jaringan untuk secara aktif memantau ancaman yang sedang berlangsung [11].



Gambar 2. 1 Arsitektur penerapan IDS [14]

Seperti yang dapat diilustrasikan dalam arsitektur jaringan pada Gambar 2.1, IDS umumnya diposisikan untuk memantau trafik yang masuk dari internet melalui *router* dan *firewall*, sebelum mencapai server internal atau pengguna akhir (*end users*). IDS akan menganalisis paket data yang lewat, membandingkannya dengan basis data tanda tangan serangan (*attack signature database*) yang telah diketahui, atau mendeteksi anomali perilaku jaringan yang menyimpang dari pola normal. Ketika terdeteksi adanya potensi ancaman, IDS dapat mengirimkan peringatan kepada administrator atau bahkan dalam beberapa konfigurasi, secara otomatis memicu respons seperti memblokir alamat IP penyerang melalui *firewall*.

Beberapa tools IDS yang banyak digunakan saat ini antara lain Snort, Suricata, dan Zeek. Tools tersebut umumnya bekerja menggunakan pendekatan signature-based atau rule-based dan dijalankan di level jaringan atau host.

Pendekatan ini efektif dalam mengenali serangan yang telah dikenal, namun memiliki keterbatasan dalam menangani serangan baru (zero-day) dan anomali kompleks. Selain itu, sebagian besar *tools* tersebut belum terintegrasi langsung dalam jalur komunikasi aplikasi seperti middleware API.

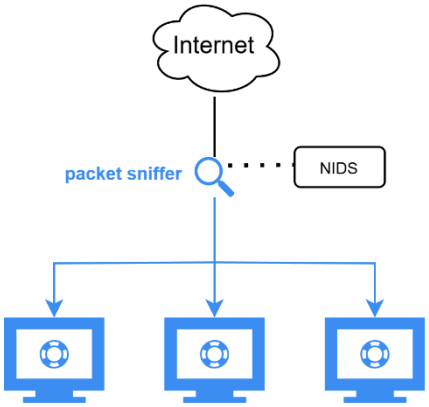
Dalam konteks arsitektur sistem modern, terdapat pendekatan IDS yang menempatkan modul deteksi langsung di dalam *middleware* API, bertujuan menyaring permintaan jaringan sebelum mencapai endpoint utama. Pendekatan ini memungkinkan deteksi yang lebih cepat dan dapat bekerja secara real-time. Integrasi ini umumnya memanfaatkan algoritma deteksi berbasis perilaku seperti *Deep Neural Network* (DNN) yang mampu mengenali pola serangan berdasarkan trafik jaringan tanpa perlu tanda tangan tetap, sehingga dapat melengkapi fungsi IDS konvensional yang sudah ada.

2.2.2.1 Anomaly-based detetction

Deteksi berbasis anomali adalah sebuah metode deteksi yang membandingkan pola atau perilaku sebuah host atau jaringan dengan pola atau perilaku yang diharapkan dari sebuah host atau jaringan tersebut. Pola atau perilaku yang tidak diharapkan tersebut yang nanti akan dideteksi sebagai anomali. Tidak seperti inspeksi paket di mana deteksi eksfiltrasi data didasarkan pada pemantauan konten (data aktual), teknik deteksi berbasis anomali menganalisis konteks seperti sumber, tujuan, ukuran, waktu, dan format untuk mendeteksi eksfiltrasi data [13].

2.2.2.2 Network-based anomaly detection

Seperti yang diilustrasikan pada gambar 2.2, *Network-based anomaly* (NBA) *detection* memonitor lalu lintas jaringan untuk menentukan apakah arus komunikasi berbeda dari kondisi awal dalam hal volume lalu lintas, pasangan alamat sumber/tujuan, keragaman alamat tujuan, dan waktu, atau (kesalahan) penggunaan protokol jaringan tertentu [13]. Setiap perbedaan yang dideteksi akan diidentifikasi sebagai anomali yang bisa menuju ke serangan atau aktivitas berbahaya.



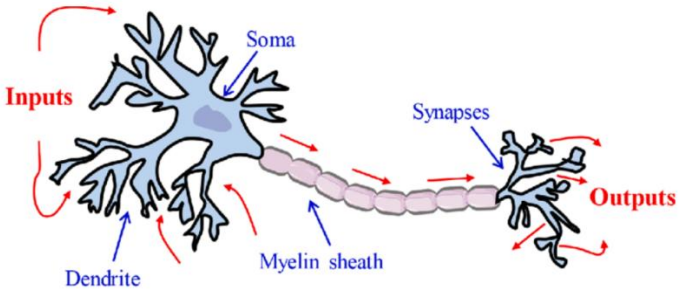
Gambar 2. 2 Network-based IDS

2.2.3 *Deep learning*

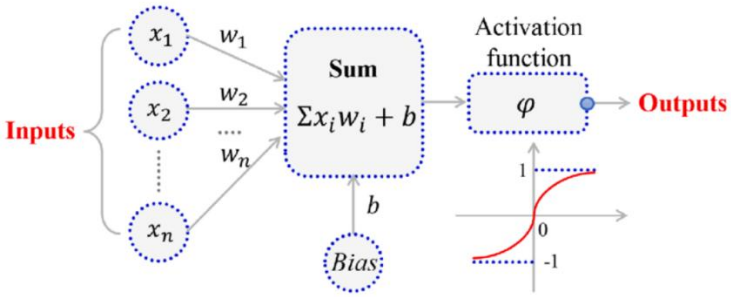
Deep learning merupakan cabang lanjutan dari *machine learning* yang mendorong perkembangan machine learning semakin mendekati kecerdasan buatan (*Artificial Intelligence*). Pendekatan ini memungkinkan pemodelan hubungan dan konsep yang kompleks melalui penggunaan beberapa tingkat representasi fitur. Proses pembelajaran dilakukan baik secara terawasi (*supervised*) maupun tak terawasi (*unsupervised*), di mana setiap tingkat abstraksi yang lebih tinggi dibangun berdasarkan keluaran dari tingkat sebelumnya [6].

2.2.4 Deep Neural Network

Deep Neural Network (DNN) merupakan model jaringan saraf tiruan yang dikembangkan dengan arsitektur berlapis-lapis untuk meniru cara kerja otak manusia dalam mengenali pola. Lapisan dalam arsitektur DNN terdiri dari unit-unit buatan yang menyerupai neuron (seperti gambar 2.3 dan gambar 2.4) yang terhubung satu sama lain dan bobot koneksi menentukan apa yang dihitung oleh sebuah unit, yaitu fitur *input* apa yang direpresentasikan olehnya[17].

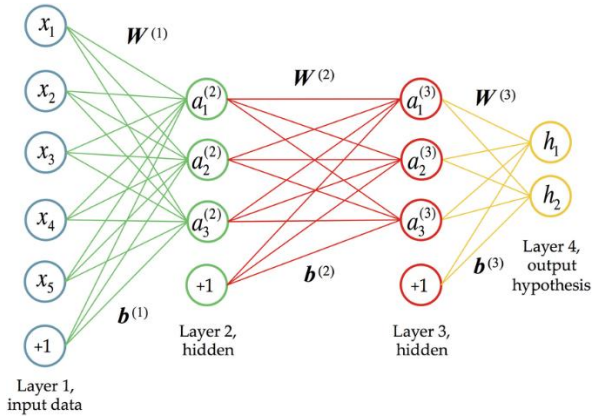


Gambar 2. 3 *Neuron* dalam tubuh manusia [15]



Gambar 2. 4 *Neuron* dalam *deep learning* [15]

Unit *neuron* biasanya dibagi dalam beberapa lapis seperti pada gambar 2.5



Gambar 2. 5 Arsitektur lapisan *deep learning* [16]

Lapisan *input* diaktifkan secara langsung oleh masukan (dalam konteks penelitian ini, data lalu lintas jaringan). Lalu data dari *input layer* diteruskan ke dalam *hidden layer* untuk diproses lebih lanjut oleh *neuron-neuron* dengan bias dan bobot tertentu. Jumlah hidden layer dan jumlah neuron dipengaruhi oleh fungsi aktivasi yang digunakan. Fungsi aktivasi adalah sebuah fungsi penting dalam neural network yang berperan dalam menambahkan karakteristik non-linear pada hasil *output hidden layer*, sehingga *neural network* mampu mempelajari hubungan

kompleks antara data masukan dan target keluaran. Fungsi aktivasi yang umum digunakan diantara lain adalah :

a. ReLU (*Rectified Linear Activation*)

   fungsi ReLU menghasilkan *output* yang sama dengan *input* jika *input*nya positif, dan menghasilkan *output* 0 jika *input*nya negatif. Fungsi ini digunakan untuk masalah regresi dan klasifikasi.

$$f(x) = Max(0, x) \tag{2.1}$$

   Persamaan diatas berarti *output* dari ReLU adalah nilai *input*nya jika nilai *input* tersebut lebih besar dari atau sama dengan nol, dan *output*-nya adalah nol jika nilai *input*nya kurang dari nol

b. Leaky ReLU

$$f(x) = Max(0.1x, x) \tag{2.2}$$

   Fungsi ReLU dapat menyebabkan masalah "*neuron* mati" karena gradiennya nol pada nilai negatif, sehingga beberapa *neuron* berhenti belajar selama pelatihan. Untuk mengatasinya, dikembangkan Leaky ReLU yang memberikan kemiringan kecil pada sisi negatif agar *neuron* tetap dapat diperbarui.

c. Tanh

$$f(x) = \tanh(x) \tag{2.3}$$

   Fungsi tanh menghasilkan *output* dalam rentang -1 hingga 1. Fungsi ini sering digunakan pada lapisan *hidden* karena bisa menghasilkan representasi yang simetris terhadap nilai 0.

Setelah data diproses dalam *hidden layer*, data diteruskan ke *output* layer untuk selanjutnya diproses menggunakan fungsi aktivasi untuk mengubah *output* numerik mentah *neuron* menjadi bentuk yang dapat dimaknai. Fungsi aktivasi yang umum digunakan pada *output* layer adalah sebagai berikut :

a. Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.4}$$

   fungsi sigmoid menghasilkan *output* dalam rentang 0 hingga 1. Fungsi ini digunakan untuk masalah klasifikasi biner.

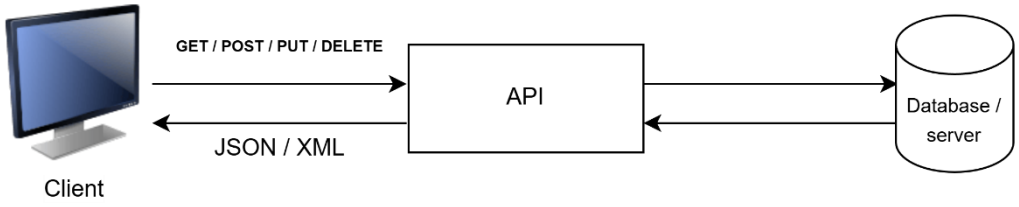
b. Softmax

$$Softmax(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad (2.5)$$

Softmax adalah jenis fungsi aktivasi yang umum digunakan pada lapisan *output* dari jaringan saraf, terutama untuk tugas klasifikasi *multiclass*. Fungsi *softmax* mengubah nilai *input* menjadi distribusi probabilitas yang memetakan *output* pada rentang (0, 1) sehingga total probabilitas *output* menjadi 1.

2.2.5 Application Programming Interface

API (*Application Programming Interface*) adalah sekumpulan aturan dan protokol yang memungkinkan aplikasi perangkat lunak untuk saling berkomunikasi. API memberikan cara bagi pengembang untuk mengakses fungsi tertentu dari suatu sistem atau layanan tanpa perlu memahami rincian implementasinya. Seperti yang diilustrasikan pada Gambar 2.6, API bertindak sebagai perantara antara *client* (pengguna atau aplikasi lain) dan *database/server*. *Client* mengirimkan permintaan ke API, kemudian API meneruskan permintaan tersebut ke *server* atau *database* untuk diproses, dan selanjutnya mengembalikan respons dari *server* ke *client*.



Gambar 2. 6 diagram sederhana API

Dengan memanfaatkan API, pengembang dapat menggunakan layanan yang sudah ada, seperti database atau layanan web, untuk menciptakan aplikasi baru dengan lebih efisien. Selain itu, API juga mendukung interoperabilitas antara berbagai platform dan bahasa pemrograman, sehingga memudahkan integrasi dan pertukaran data di antara sistem yang berbeda.

2.2.6 *Middleware*

Middleware adalah perangkat lunak yang berfungsi sebagai jembatan penghubung antara berbagai aplikasi atau sistem, memungkinkan mereka untuk saling berkomunikasi dan bertukar data dengan cara yang efisien. Seperti diilustrasikan pada Gambar 2.7, *middleware* bertindak sebagai perantara yang menerima permintaan (misalnya, *HTTP Request*) dari satu sisi klien (*client side*), memproses atau meneruskannya, dan kemudian mengirimkannya ke sisi klien lainnya atau ke sistem tujuan. Proses serupa terjadi untuk respons (*HTTP Response*) yang dikirim kembali. Peran *middleware* sangat penting dalam arsitektur perangkat lunak modern karena ia menyediakan layanan yang mendukung pengembangan aplikasi dengan menyederhanakan proses integrasi dan interaksi antar komponen perangkat lunak.

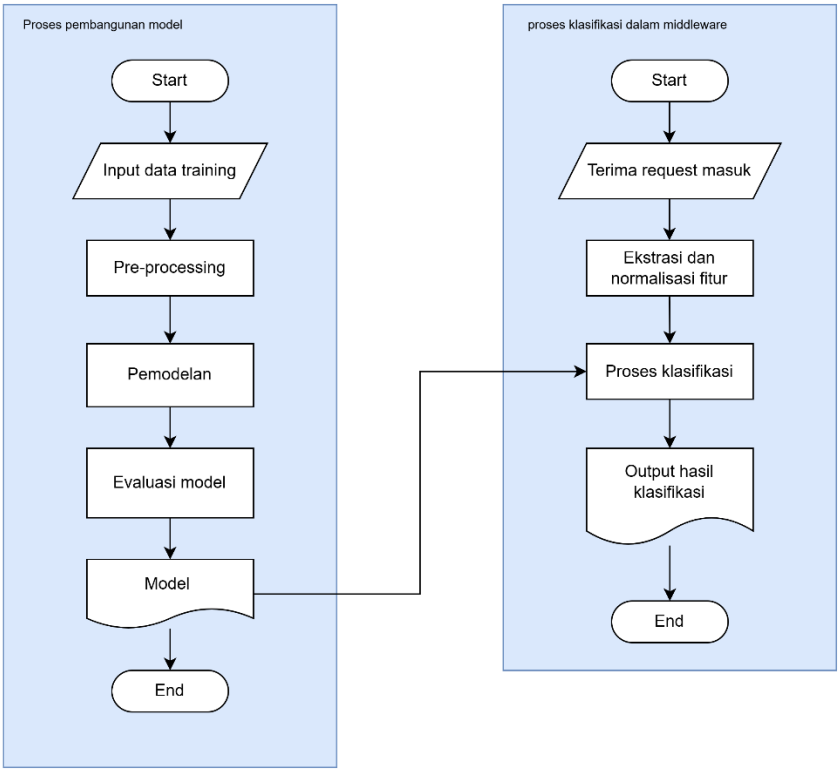
Gambar 2. 7 Diagram sederhana *middleware*

Middleware dapat digunakan baik dalam lingkungan lokal maupun di cloud, sehingga memberikan fleksibilitas bagi pengembang untuk membangun solusi yang lebih terintegrasi. Dengan memanfaatkan *middleware*, pengembang tidak hanya dapat mengurangi kompleksitas dalam mengelola komunikasi antar sistem, tetapi juga meningkatkan responsivitas dan kinerja aplikasi secara keseluruhan. Selain itu, *middleware* sering kali mencakup fitur tambahan seperti manajemen transaksi, keamanan data, dan pemantauan kinerja, menjadikannya alat yang sangat berguna dalam menciptakan ekosistem aplikasi yang kohesif dan efektif.

3. PERANCANGAN SISTEM

3.1 Gambaran Umum Sistem

Pada gambar 3.1, secara umum menggambarkan alur sistem yang akan dirancang. Sistem yang dirancang pada penelitian ini merupakan sistem deteksi intrusi (Intrusion Detection System/IDS) berbasis *Deep Neular Network* yang mampu mengklasifikasikan jenis serangan siber secara otomatis dan terintegrasi dalam *middleware* API. Sistem ini memanfaatkan dataset CIC-IDS2017 sebagai sumber data pelatihan dan pengujian model, serta dirancang untuk memproses lalu lintas jaringan secara real-time. Setiap permintaan yang masuk akan dianalisis oleh *middleware* menggunakan model klasifikasi, sehingga dapat diidentifikasi apakah permintaan tersebut tergolong normal atau merupakan salah satu jenis serangan siber.



Gambar 3. 1 Alur sistem

Dalam penelitian ini, seluruh proses pembuatan sistem dilakukan di dalam lingkungan *cloud*, yaitu *Microsoft Azure*. Untuk proses pembuatan model IDS

menggunakan DNN, peneliti menggunakan fitur *notebook Azure AI* dengan *compute instance processor Intel(R) Xeon(R)* CPU E5-2673 v3 @ 2.40GHz 4 *core* 32GiB *memory* dan 150 GB *disk*. Untuk menyimpan dataset dan juga hasil pelatihan, digunakan *Azure cloud storage* dengan jenis *storageV2*. Sedangkan untuk skenario pengujian, akan menggunakan layanan *Azure App Service* yang merupakan layanan hosting terkelola untuk aplikasi web, termasuk situs web dan API web.

3.2 Data Set dan Pengumpulan Data

Dataset yang digunakan pada penelitian ini adalah CIC-IDS2017. Dataset CIC-IDS2017 berisi data lalu lintas jaringan yang digunakan untuk pengembangan dan evaluasi IDS. Dataset ini dirancang agar merepresentasikan lalu lintas jaringan modern dan mencakup lebih dari 2,8 juta paket jaringan yang direkam selama periode tujuh hari dalam lingkungan jaringan nyata. Dataset ini mencakup lalu lintas normal serta tujuh skenario serangan yang berbeda, dijelaskan pada tabel 3.1. Dataset ini memiliki ketidakseimbangan yang tinggi, di mana sebagian besar data termasuk dalam kelas 'Benign' dan hanya sebagian kecil data yang termasuk dalam kelas-kelas lainnya.

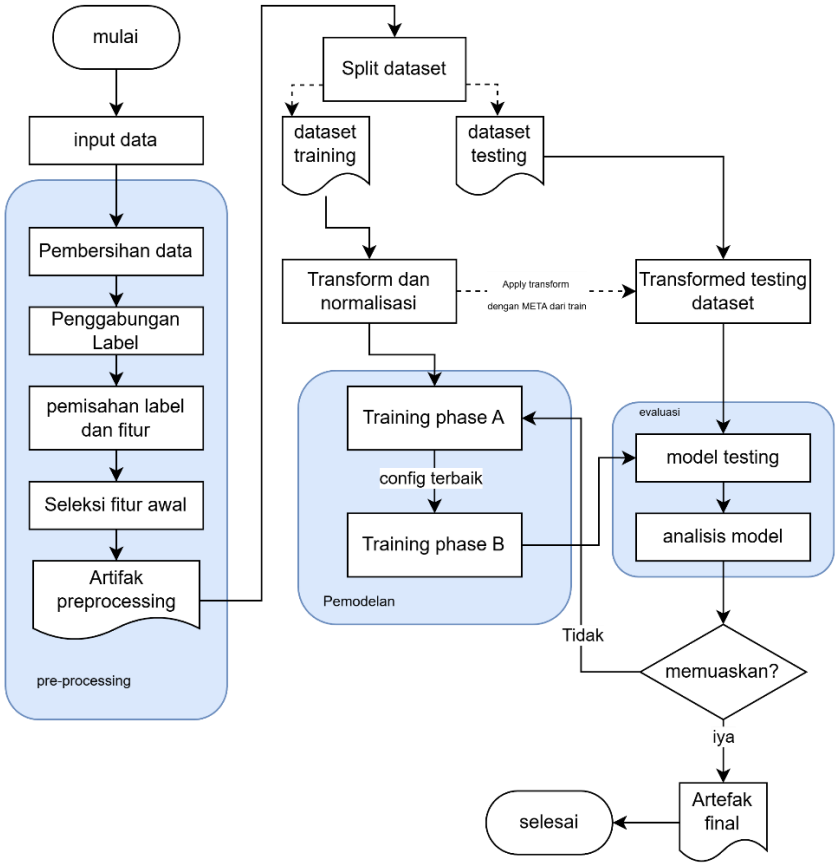
Tabel 3. 1 Distribusi serangan dataset CIC-IDS2017

Attack Type	Jumlah Data	Deskripsi Singkat
BENIGN	2,273,097	Lalu lintas jaringan normal tanpa aktivitas berbahaya atau serangan.
DoS	252,661	Serangan *Denial of Service* yang bertujuan membuat layanan tidak tersedia dengan membanjiri trafik.
Port Scan	158,930	Upaya pemindaian port pada target untuk menemukan layanan terbuka dan potensi celah keamanan.
DDoS	128,027	Serangan *Distributed Denial of Service*, DoS yang dilakukan dari banyak sumber secara bersamaan.
Brute Force	13,835	Serangan mencoba menebak kredensial (password) dengan mencoba banyak kombinasi secara otomatis.
Web Attack	2,180	Serangan terhadap aplikasi web seperti *SQL Injection*, *Cross-Site Scripting* (XSS), dan *brute force web*.
Bot	1,966	Trafik dari perangkat terinfeksi *malware botnet* yang dikendalikan untuk melakukan serangan atau spam.
Infiltration	36	Upaya penyusupan ke dalam jaringan setelah melewati pertahanan awal seperti firewall atau IDS/IPS.

Attack Type	Jumlah Data	Deskripsi Singkat
Heartbleed	11	Eksploitasi kerentanan *Heartbleed* di *OpenSSL* yang memungkinkan pencurian data sensitif dari memori server.

3.3 Pembangunan Model

Pada gambar 3.2 ditunjukan gambaran proses dari pembangunan model IDS. Tahap ini meliputi beberapa proses diantaranya :



Gambar 3. 2 workflow pemodelan

3.3.1 *Pre-processing* data

Seperti yang ditunjukkan pada gambar 3.2, proses pra-pemrosesan (pre-processing) data merupakan tahap awal yang sangat penting sebelum pembangunan model machine learning dilakukan. Tahap ini terdiri dari beberapa langkah sistematis untuk mempersiapkan dataset agar siap digunakan dalam pelatihan dan pengujian model, seperti yang dijelaskan pada tabel 3.2

Tabel 3. 2 Tahap pre-processing data

Proses	Deskripsi
Pembersihan data	duplicate data : menghapus data yang duplikat infinites value : merubah menjadi NaN NaN value : diubah menjadi median dari atribut tersebut
Penggabungan label (label collapsing)	Menggabungkan sub-tipe serangan menjadi 9 kelas (mis. semua DoS → "DoS", Web Attack → "Web Attack", FTP/SSH-Patator → "Brute Force") untuk meningkatkan stabilitas pembelajaran.
Pemisahan label dan fitur	Memisahkan antara kolom label dan kolom fitur. Mengubah label kategori menjadi nilai numerik (encoding). Melakukan downcast tipe data untuk efisiensi memori.
Seleksi fitur awal	Variance Threshold: menghapus kolom yang tidak memiliki variasi (nilai sama semua). Pruning korelasi: menghapus salah satu dari pasangan fitur dengan korelasi sangat tinggi (> 0.98).

3.3.1.1 Pembersihan data

Proses ini bertujuan untuk memastikan integritas dan konsistensi data sebelum digunakan untuk pelatihan model. Pembersihan data dilakukan dengan menghapus nilai duplikat, menghapus baris dengan durasi negatif, mengganti nilai infinite (±∞) dengan nilai NaN, dan mengganti nilai NaN dengan nilai median dari fitur tersebut.

Algoritma 3.1: Data Cleaning

```
1:  Deklarasi:
2:  Data ← Dataset mentah
3:  n ← jumlah baris dalam Data
4:  Implementasi Algoritma:
5:  BEGIN
6:  ....// Hapus duplikat
7:  ....FOR i ← 1 TO n DO
```

```
8:  ........IF Data[i] adalah duplikat THEN
9:  ............ Hapus Data[i]
10: ........ENDIF
11: ....ENDFOR
12:
13: ....// Hapus baris dengan durasi negatif
14: ....FOR i ← 1 TO n DO
15: ........IF Data[i]['Flow Duration'] < 0 THEN
16: ............ Hapus Data[i]
17: ........ENDIF
18: ....ENDFOR
19:
20: ....// Ganti nilai infinite dengan NaN
21: ....FOR setiap kolom j dalam Data DO
22: ........FOR setiap baris i DO
23: ............IF Data[i][j] == ∞ OR Data[i][j] == -∞ THEN
24: ................Data[i][j] ← NaN
25: ............ ENDIF
26: ........ ENDFOR
27: ....ENDFOR
28:
29: ....// Ganti NaN dengan median
30: ....FOR setiap kolom j dalam Data DO
31: ........ Median_j ← HITUNG_MEDIAN(Data[:,j] TANPA NaN)
32: ........GANTI semua NaN di kolom j dengan Median_j
33: ....ENDFOR
34: END
35: Output:
36: Dataset bersih tanpa duplikat, durasi negatif, dan nilai tak
hingga
```

3.3.1.2 Penggabungan Label (Label Collapsing)

Proses ini bertujuan untuk menggabungkan sub-tipe serangan yang memiliki karakteristik serupa menjadi satu kelas yang lebih umum. Penggabungan dilakukan karena beberapa sub-tipe serangan memiliki jumlah sampel yang sangat sedikit (ultra-rare), sehingga sulit untuk dipelajari oleh model secara stabil. Dengan

penggabungan, jumlah sampel per kelas meningkat dan pembelajaran menjadi lebih stabil.

Tabel 3. 3 Skema Penggabungan Label

Label Asli	Label Gabungan	Alasan
BENIGN	BENIGN	Kelas mayoritas, tidak digabung
DDoS	DDoS	Jumlah sampel besar, tidak digabung
DoS Hulk, DoS GoldenEye, DoS Slowloris, DoS Slowhttptest	DoS	Semua varian serangan DoS digabung
PortScan	Port Scan	Jumlah sampel besar, tidak digabung
FTP-Patator, SSH-Patator	Brute Force	Keduanya serangan brute force pada protokol berbeda
Bot	Drop	Bukan network attack
Web Attack – Brute Force, Web Attack – XSS, Web Attack – Sql Injection	Drop	Bukan network attack
Infiltration	Drop	Bukan network attack
Heartbleed	Drop	Bukan network attack

Algoritma 3.2: Label collapsing

```
Deklarasi:
Map_collapse ← kamus pemetaan label asli → label gabungan
Implementasi Algoritma:
BEGIN
....// Definisi mapping
....Map_collapse ← {
........'DoS Hulk' → 'DoS',
........ 'DoS GoldenEye' → 'DoS',
........'DoS Slowloris' → 'DoS',
........ 'DoS Slowhttptest' → 'DoS',
```

```
11: ........'FTP-Patator' → 'Brute Force',
12: ........'SSH-Patator' → 'Brute Force',
16: ........ // Label lain tetap sama
17: ....}
18:
19: ....// Terapkan mapping
20: ....FOR setiap baris i dalam Data DO
21: ........label_asli ← Data[i]['Label']
22: ........IF label_asli ADA di Map_collapse THEN
23:.........Data[i]['Label_collapsed'] ← Map_collapse[label_asli]
24: ........ELSE
25: ............Data[i]['Label_collapsed'] ← label_asli
26: ........ ENDIF
27: .... ENDFOR
28: END
29: Output:
30: Dataset dengan kolom Label_collapsed berisi 5 kelas
```

3.3.1.3 Pemisahan Label dan Fitur

Proses ini bertujuan untuk memisahkan fitur input (X) dan label output (y) agar bisa digunakan dalam pelatihan supervised learning. Label yang berupa kategori akan diubah menjadi nilai numerik. Selain itu, dilakukan downcast tipe data dari float64 ke float32 untuk menghemat penggunaan memori.

Algoritma 3.3: Pemisahan Label dan Fitur

```
1:  Deklarasi:
2:  Fitur ← seluruh kolom numerik selain kolom label
3:  Label ← kolom 'Label_collapsed'
4:  Implementasi Algoritma:
5:  BEGIN
6:  .... X ← Data[Fitur]
7:  ....y ← Data[Label]
8:
9:  ....// Encoding label ke numerik
10: ....Label_Unik ← daftar unik dari y (diurutkan)
11: ....label_map ← {}
12: ....FOR i ← 0 TO jumlah(Label_Unik) - 1 DO
```

```
13: ........ label_map[Label_Unik[i]] ← i
14: ....ENDFOR
15: ....y_encoded ← MAP(y, label_map)
16:
17: ....// Downcast tipe data untuk efisiensi memori
18: ....FOR setiap kolom j dalam X DO
19: ........X[j] ← CONVERT(X[j], float32)
20: .... ENDFOR
21: END
22: Output:
23: X sebagai fitur (float32), y_encoded sebagai label numerik,
label_map
```

3.3.1.4 Seleksi Fitur Awal

Proses seleksi fitur bertujuan untuk menghapus fitur yang tidak memberikan informasi berguna bagi pembelajaran model. Terdapat dua langkah dalam seleksi fitur awal:

1. Variance Threshold: Menghapus kolom yang tidak memiliki variasi (semua nilainya sama). Fitur tanpa variasi tidak dapat membedakan antar kelas.
2. Pruning Korelasi: Menghapus salah satu dari pasangan fitur yang memiliki korelasi sangat tinggi (> 0.98). Fitur dengan korelasi tinggi bersifat redundan karena membawa informasi yang hampir identik.

Algoritma 3.4: Seleksi fitur variance threshold

```
1:  Deklarasi:
2:  Threshold_variance ← 0 (minimal harus ada variasi)
3:  Implementasi Algoritma:
4:  BEGIN
5:  ....FOR setiap kolom j dalam X DO
6:  ........variance_j ← HITUNG_VARIANCE(X[:,j])
7:  ........IF variance_j ≤ Threshold_variance THEN
8:  ............Hapus kolom j dari X
9:  ........ ENDIF
10: ....ENDFOR
11: END
```

```
12: Output:
13: X dengan fitur yang memiliki variasi
```

Algoritma 3.4: Seleksi fitur pruning korelasi

```
1:  Deklarasi:
2:  Threshold_corr ← 0.98
3:  Implementasi Algoritma:
4:  BEGIN
5:  .... Corr_matrix ← HITUNG_KORELASI_ABSOLUT(X)
6:  ....drop_cols ← []
7:
8:  ....FOR setiap pasangan kolom (i, j) dimana i < j DO
9:  ........ IF Corr_matrix[i][j] > Threshold_corr THEN
10: ............ TAMBAHKAN kolom j ke drop_cols
11: ........ENDIF
12: .... ENDFOR
13:
14: ....FOR setiap kolom dalam drop_cols DO
15: ........ Hapus kolom dari X
16: ....ENDFOR
17: END
18: Output:
19: X dengan fitur tanpa redundansi tinggi
```

3.3.1.5 Artefak Preprocessing

Setelah seluruh proses pre-processing selesai, beberapa artefak penting disimpan untuk digunakan pada tahap selanjutnya dan untuk keperluan inference di masa depan:

- feature_cols: Daftar nama fitur yang digunakan
- label_map: Mapping label kategori ke numerik
- inv_label_map: Mapping numerik ke label kategori (kebalikan)
- original_to_collapsed: Mapping label asli ke label gabungan

3.3.2 Split Dataset

Setelah pre-processing selesai, dataset dibagi menjadi dua bagian menggunakan metode stratified split:

- Dataset Training (80%): Digunakan untuk melatih model
- Dataset Testing (20%): Disimpan terpisah (hold-out) dan hanya digunakan sekali di akhir untuk evaluasi final

Stratified split memastikan proporsi setiap kelas tetap terjaga di kedua bagian. Hal ini dilakukan agar kelas minoritas tetap terwakili di dataset training maupun testing.

Algoritma 3.5 : Straified split

```
1:  Deklarasi:
2:  Rasio_test ← 0.20 (20%)
3:  Implementasi Algoritma:
4:  BEGIN
5:  ....// Hitung jumlah sampel per kelas
6:  .... FOR setiap kelas k dalam y DO
7:  ........count_k ← jumlah sampel kelas k
8:  ........n_test_k ← ROUND(count_k × Rasio_test)
9:  ........n_train_k ← count_k - n_test_k
10: .... ENDFOR
11:
12: ....// Bagi data per kelas
13: ....X_train ← [], y_train ← []
14: ....X_test ← [], y_test ← []
15: ....FOR setiap kelas k DO
16: ........indices_k ← indeks semua sampel kelas k (diacak)
17: ........X_train. APPEND(X[indices_k[:n_train_k]])
18: ........X_test. APPEND(X[indices_k[n_train_k:]])
19: .... ENDFOR
20: END
21: Output:
22: X_train, y_train (80%) dan X_test, y_test (20%) dengan
proporsi kelas terjaga
```

3.3.3 Transform dan Normalisasi

Proses transformasi dan normalisasi dilakukan setelah split dataset. Hal ini dilakukan untuk mencegah data leakage, yaitu kondisi di mana informasi dari data testing "bocor" ke proses pelatihan. Prinsip utama pada proses ini adalah Fit hanya di data training, kemudian parameter yang didapat digunakan untuk Transfom data training dan testing. Parameter transform disimpan sebagai Transform META untuk digunakan pada inference data baru.

Proses transformasi meliputi:

1. Identifikasi Heavy Columns: Menandai kolom dengan distribusi sangat miring (skewness > 1. 5)
2. Transformasi Log1p: Menerapkan log(1+x) pada heavy columns untuk mengurangi rentang nilai ekstrem
3. Normalisasi dengan RobustScaler: Menormalkan data menggunakan median dan IQR (Interquartile Range). RobustScaler dipilih karena dataset memiliki outlier ekstrem (misalnya flow duration yang sangat panjang atau packet size yang sangat besar).

Algoritma 3.6 : Transform dan normalisasi

```
1:  Deklarasi:
2:  Skew_threshold ← 1.5
3:  Implementasi Algoritma:
4:  BEGIN
5:  .... // LANGKAH 1: Identifikasi heavy columns (FIT di train)
6:  ....heavy_cols ← []
7:  ....FOR setiap kolom j dalam X_train DO
8:  ........ skew_j ← HITUNG_SKEWNESS(X_train[:,j])
9:  ........IF |skew_j| > Skew_threshold THEN
10: ............ TAMBAHKAN j ke heavy_cols
11: ........ ENDIF
12: .... ENDFOR
13:
14: ....// LANGKAH 2: Transformasi log1p pada heavy columns
```

```
15: .... FOR setiap kolom j dalam heavy_cols DO
16: ........X_train[:,j] ← LOG(1 + MAX(0, X_train[:,j]))
17: ........X_test[:,j] ← LOG(1 + MAX(0, X_test[:,j]))
18: .... ENDFOR
19:
20: ....// LANGKAH 3: Normalisasi dengan RobustScaler (FIT di train)
21: ....scaler ← RobustScaler()
22: ....scaler.FIT(X_train)  // Hitung median & IQR dari train
23: ....
24: ....X_train ← scaler. TRANSFORM(X_train)
25: ....X_test ← scaler.TRANSFORM(X_test)  // Pakai parameter dari
train!
26:
27: .... // Simpan Transform META untuk inference
28: ....META ← {cols_final, heavy_cols, medians, scaler}
29: END
30: Output:
31: X_train_transformed, X_test_transformed, Transform META
```

Transform META yang disimpan meliputi:

- cols_final: Daftar kolom fitur yang digunakan
- heavy_cols: Daftar kolom yang diterapkan log1p
- medians: Nilai median per kolom untuk imputasi NaN
- scaler: Objek RobustScaler yang sudah di-fit

META ini akan digunakan untuk transform data testing dengan parameter yang sama dari training dan transform data baru saat inference/deployment

3.3.4 Pemodelan IDS

Setelah data selesai ditransformasi, dilakukan pemodelan dalam dua fase yaitu:

a. Phase A (Eksplorasi Hyperparameter): Menggunakan subset data training untuk mencari kombinasi parameter terbaik secara efisien
b. Phase B (Training Final): Menggunakan seluruh data training dengan konfigurasi terbaik dari Phase A.

3.3.4.1 Phase A (Eksplorasi Hyperparameter)

Phase A menggunakan subset dari data training untuk mempercepat proses pencarian hyperparameter. Subset dibuat dengan melakukan sampling pada kelas BENIGN (kelas mayoritas) sambil mempertahankan semua sampel kelas serangan. Pada Phase A, tidak dilakukan balancing data (oversampling/undersampling). Sebagai gantinya, digunakan class weight pada fungsi loss untuk memberikan bobot lebih pada kelas minoritas. Hal ini dilakukan agar sinyal hyperparameter tidak bias oleh sampel hasil oversampling. Data subset kemudian dibagi menjadi sub-train (80%) dan validation (20%) untuk evaluasi performa setiap kombinasi hyperparameter.

Tabel 3. 4 parameter yang akan diuji

Batch	Parameter yang Diuji	Variasi yang Dicoba
1	Split Data (Train:Test)	70:30, 80:20, 90:10
2	Hidden Layer	[256,128,128], [256,256,128], [256,256,128,128], [512,256,128]
3	Activation Function	relu, leaky_relu
4	Epochs	60, 100, 140 (dengan early stopping)
5	Learning Rate	0.001, 0.003
6	Batch Size	32, 64, 128

Seperti pada table 3.3, eksperimen dilakukan secara bertahap dalam 6 batch pengujian untuk menemukan kombinasi parameter terbaik pada model *Deep Neural Network* (DNN). Ketika sudah menemukan hasil terbaik dari variasi yang diuji

maka akan dilanjutkan ke *batch* selanjutnya. Pemilihan konfigurasi terbaik didasarkan pada nilai Macro F1 pada data validation.

3.3.4.2 Phase B (Training Final)

Phase B menggunakan seluruh data training dengan konfigurasi terbaik dari Phase A. Pada fase ini menerapkan mild adaptive balancing untuk meningkatkan representasi kelas ultra-rare.

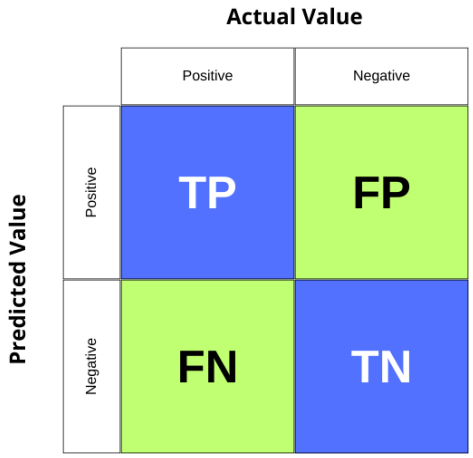
Mild Adaptive Balancing meliputi:

1. Pembatasan BENIGN: Membatasi jumlah sampel BENIGN agar tidak terlalu mendominasi
2. Oversampling kelas kecil: Menduplikasi sampel kelas ultra-rare hingga mencapai ambang minimum tertentu

Metode ini berbeda dengan SMOTE yang mensintesis sampel baru. Mild adaptive hanya menduplikasi sampel yang sudah ada, sehingga tidak menimbulkan noise dari sampel sintetis.

3.3.5 Evaluasi Model

Setiap pengujian yang dilakukan akan dievaluasi dengan menggunakan *evaluation matrix*. Seperti ditunjukan pada gambar3.3, dalam matriks tersebut terdapat beberapa istilah penting seperti *True Positive* (TP) yaitu jumlah data positif yang berhasil diprediksi dengan tepat; *True Negative* (TN) adalah data negatif yang juga berhasil diprediksi secara akurat; *False Positive* (FP) adalah kesalahan ketika data negatif diprediksi sebagai positif; serta *False Negative* (FN) yakni kesalahan saat data positif tidak terdeteksi atau salah diklasifikasikan menjadi negatif.



Gambar 3. 3 visualisasi confusion matrix

Dengan adanya nilai TP, FP, FN, TN tersebut, dapat dicari nilai dari *performance metrics* utama yang akan digunakan dalam evaluasi model di penelitian ini, yaitu seperti *accuracy* untuk mengukur seberapa baik keseluruhan hasil klasifikasi; *precision* untuk mengetahui proporsi prediksi positif tepat di antara semua hasil prediksi positif; *recall* untuk melihat seberapa banyak kasus positif sebenarnya berhasil ditemukan oleh model; serta *F1-score* sebagai rata-rata harmonis antara precision dan *recall* guna memberikan gambaran keseimbangan keduanya.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.1)$$

$$Precision = \frac{TP}{TP+FP} \quad (3.2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3.3)$$

$$F1\ score = \frac{recall \times precision}{recall+precision} \quad (3.4)$$

Namun, Pada kondisi data yang sangat tidak seimbang (imbalanced) seperti dataset CIC-IDS2017, accuracy dapat menyesatkan karena didominasi oleh kelas mayoritas (BENIGN). Sebagai contoh, model yang memprediksi semua data sebagai BENIGN akan mendapatkan accuracy > 80%, padahal tidak mampu mendeteksi serangan sama sekali. Oleh karena itu, Macro F1 digunakan sebagai metrik utama. Macro F1 adalah rata-rata F1-score dari semua kelas tanpa pembobotan:

$$Macro\ F1 = \frac{1}{K}\sum_{k=1}^{K} F1_k \quad (3.5)$$

di mana K adalah jumlah kelas. Macro F1 memberikan bobot yang sama pada setiap kelas, sehingga performa pada kelas minoritas tetap diperhitungkan secara adil.

3.3.6 Artefak Final

Jika hasil evaluasi memuaskan, seluruh artefak disimpan untuk keperluan deployment dan inference:

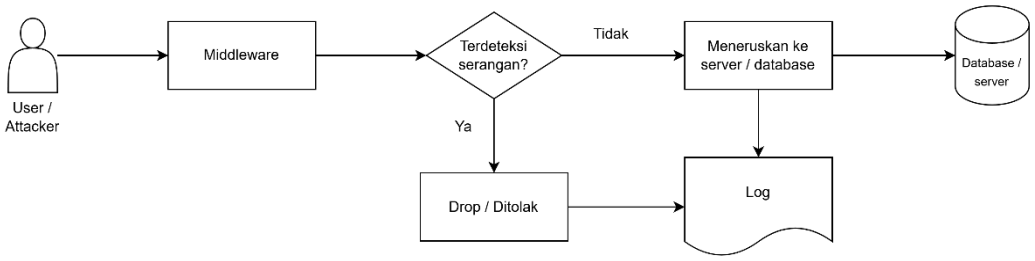
Tabel 3. 5 Artefak yang disimpan

Artefak	Format	Keterangan
model_state.pt	PyTorch state dict	Bobot model yang sudah dilatih
scaler.pkl	Pickle	RobustScaler yang sudah di-fit
transform_meta.json	JSON	Kolom final, heavy cols, medians
config. json	JSON	Hyperparameter terbaik
label_map.json	JSON	Mapping label ke numerik
report.json	JSON	Hasil evaluasi (metrik)

Artefak-artefak ini memungkinkan model untuk digunakan kembali pada data baru tanpa perlu melakukan training ulang.

3.5 Skenario Penerapan dan Pengujian *Middleware* IDS

Untuk memastikan model deteksi intrusi yang dikembangkan dapat berfungsi dalam lingkungan sistem informasi, dilakukan penerapan model ke dalam *middleware* berbasis API. Seperti yang ditunjukan pada gambar 3.4, *Middleware* ini akan bertindak sebagai lapisan keamanan tambahan yang menyaring setiap permintaan (request) jaringan sebelum diteruskan ke endpoint utama layanan. Penerapan *middleware* ini dirancang agar dapat bekerja secara modular, di mana sistem akan mengekstrak fitur yang relevan dari permintaan masuk, lalu melakukan klasifikasi menggunakan model Deep Neural Network (DNN) yang telah dilatih sebelumnya. Berdasarkan hasil klasifikasi, request akan diputuskan apakah akan diteruskan atau ditolak.



Gambar 3. 4 alur pengujian *middleware*

Untuk menguji performa *middleware*, dilakukan simulasi serangan dan request normal menggunakan berbagai tools, seperti hping3, slowloris, curl, dan Postman. Tools ini mewakili berbagai jenis lalu lintas yang mungkin terjadi dalam sistem. *Middleware* akan memproses setiap request dan mencatat keputusan serta waktu responsnya. Pengujian dilakukan dalam beberapa skenario, termasuk jenis serangan yang terdapat pada dataset, serta request normal. Tujuannya adalah untuk mengevaluasi ketepatan klasifikasi, kecepatan inferensi model, serta kestabilan *middleware* dalam menerima lalu lintas berkelanjutan.

DAFTAR PUSTAKA

[1] Saranya, T., Sridevi, S., Deisy, C., Chung, T. D., & Khan, M. K. A. Ahamed. (2020). Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review. Procedia Computer Science, 171, 1251–1260. https://doi.org/10.1016/j.procs.2020.04.133

[2] Badan Siber dan Sandi Negara. (2024). Laporan Tahunan Honeynet 2024. Jakarta: Direktorat Deteksi Ancaman Siber, BSSN.

[3] Togay, C., Kasif, A., Catal, C., & Tekinerdogan, B. (2021). A Firewall Policy Anomaly Detection Framework for Reliable Network Security. IEEE Transactions on Reliability, 71(1), 1–9. https://doi.org/10.1109/tr.2021.3089511

[4] Delplace, A., Hermoso, S., Au, H. N., & Anandita, K. (2019). Cyber Attack Detection thanks to Machine Learning Algorithms COMS7507: Advanced Security. https://github.com/antoinedelplace/Cyberattack-Detection

[5] Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A *Deep learning* Approach for Intrusion Detection Using Recurrent Neural Networks. IEEE Access, 5, 21954–21961. https://doi.org/10.1109/access.2017.2762418

[6] Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A *Deep learning* Approach to Network Intrusion Detection. IEEE Transactions on Emerging Topics in Computational Intelligence, 2(1), 41–50. https://doi.org/10.1109/tetci.2017.2772792

[7] Bhandari, G., Lyth, A., Shalaginov, A., & Grønli, T.-M. (2023). Distributed

Deep Neural-Network-Based *Middleware* for Cyber-Attacks Detection in Smart IoT Ecosystem: A Novel Framework and Performance Evaluation Approach. Electronics, 12(2), 298. https://doi.org/10.3390/electronics12020298

[8] Prijuna Lubis, A. (2024) Intrusion Detection System Berbasis *Deep Learning* Untuk Peningkatan Mitigasi Sql Injection Dan Syn Flood Attack, Journal of Science and Social Research. Available at: http://jurnal.goretanpena.com/index.php/JSSR.

[9] Ali, S., Nasim, F., & Haider, K. (2025). Secure *Middleware* Model For Public Restful Apis. Al-Aasar, 2(1), 50-62.

[10] Silva, Giuntini, F. T., Ranieri, C. M., Meneguette, R. I., Garcia, R. D., Ramachandran, G. S., Bhaskar Krishnamachari, & Jó Ueyama. (2023). MADCS: A *Middleware* for Anomaly Detection and Content Sharing for Blockchain-Based Systems. Journal of Network and Systems Management, 31(3). https://doi.org/10.1007/s10922-023-09736-1

[11] Ozkan-Okay, M., Samet, R., Aslan, O., & Gupta, D. (2021). A Comprehensive Systematic Literature Review on Intrusion Detection Systems. IEEE Access, 1–1. https://doi.org/10.1109/access.2021.3129336

[11] Abdulganiyu, O.H., Ait Tchakoucht, T. and Saheed, Y.K. (2023) "A systematic literature review for network intrusion detection system (IDS)," International Journal of Information Security, 22(5), pp. 1125–1162. Available at: https://doi.org/10.1007/s10207-023-00682-2.

[12] Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. Artificial Intelligence Review, 53. https://doi.org/10.1007/s10462-020-09825-6

[13] Ullah, F., Edwards, M., Ramdhany, R., Chitchyan, R., Babar, M. A., & Rashid, A. (2018). Data exfiltration: A review of external attack vectors and countermeasures. *Journal of Network and Computer Applications*, *101*, 18–54. https://doi.org/10.1016/j.jnca.2017.10.016

[14] Yilmaz, Sevcan & ÇETER, Muhammet. (2018). Feature Selection and

Comparison of Classification Algorithms for Intrusion Detection. Anadolu University Journal Of Science And Technology A - Applied Sciences and Engineering. 19. 206-218. 10.18038/aubtda.356705.

[15] Wang, Xianlin & Liu, Yuqing & Xin, Haohui. (2021). Bond strength prediction of concrete-encased steel structures using hybrid machine learning method. Structures. 32. 2279-2292. 10.1016/j.istruc.2021.04.018.

[16] Sheehan, Sara & Song, Yun. (2016). Deep Learning for Population Genetic Inference. PLOS Computational Biology. 12. e1004845. 10.1371/journal.pcbi.1004845.

[17] Cichy, R. M., & Kaiser, D. (2019). Deep Neural Networks as Scientific Models. *Trends in Cognitive Sciences*, *23*(4), 305–317. https://doi.org/10.1016/j.tics.2019.01.009

[18] Wu P, Guo H. (2019). LuNet: A Deep Neural Network for Network Intrusion Detection. http://arxiv.org/abs/1909.10031

[19] Munawarah S, Arip Winanto E. (2024). Jurnal Informatika Dan Rekayasa Komputer (JAKAKOM) Deteksi Serangan DDoS SYN Flood Pada Jaringan Internet of Things (IoT) Menggunakan Metode Deep Neural Network (DNN), 4(1). doi:10.33998/jakakom.v4i1

[20] Martha, G., & Bororing, G. (2024). Pengembangan Algoritma Machine Learning Untuk Mendeteksi Anomali Dalam Jaringan Komputer [Review Of *Pengembangan Algoritma Machine Learning Untuk Mendeteksi Anomali Dalam Jaringan Komputer*]. *Jurnal Review Pendidikan Dan Pengajaran*, *7*(1).

[21] Prijuna Lubis A. (2024). Intrusion Detection System Berbasis Deep Learning Untuk Peningkatan Mitigasi Sql Injection Dan Syn Flood Attack. http://jurnal.goretanpena.com/index.php/JSSR

[22] Sirajuddin Asjad. (2023). Intrusion Detection and Cyber Attack Classification for Encrypted DDS Communication Middleware in OT Networks Using Machine Learning. http://www.usn.no

[23] Kumar Tambi V, Singh N. (2022). A New Framework and Performance Assessment Method for Distributed Deep Neural Network-Based Middleware for Cyberattack Detection in the Smart IoT Ecosystem. Impact Factor: 8.18. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 11(5). doi:10.15662/IJAREEIE.2022.1105035

[24] Liu H, Lang B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. Applied Sciences (Switzerland), 9(20). doi:10.3390/app9204396

LAMPIRAN

Data-data pendukung, *curriculum vitae*(CV) untuk pembimbing dari luar Universitas Telkom, dsb.