BAB 2
TINJAUAN PUSTAKA

2.1 Landasan Teori

2.1.1 *Keamanan Siber*

Keamanan siber adalah setiap kegiatan kriminal yang dilakukan dengan menggunakan komputer, jaringan komputer, atau internet. (Butarbutar, 2023)

Serangan siber adalah ancaman keamanan yang melibatkan aktivitas jahat melalui jaringan digital yang bertujuan untuk mencuri data, mengganggu layanan, atau merusak infrastruktur digital. (Hider & Shabir, 2024)

Contoh jenis-jenis serangan siber:

1) *Malware*
2) *Phising*
3) *Zero-Day Attack*
4) *Man-in-the-Middle Attack* (MITM)
5) *Distributed Denial of Service* (DDoS)
6) *Social Engineering*

Zero-Day attack adalah kerentanan baru tanpa pertahanan; dengan demikian, kemungkinan serangan akan memiliki probabilitas risiko tinggi, dan dampak kritis (Astudillo dkk., 2021). Contoh: Serangan Stuxnet yang mengeksploitasi beberapa *zero-day vulnerabilities*.

2.1.2 *Denial of Service (DoS)*

DoS adalah jenis serangan dunia maya yang dilakukan oleh pelaku kejahatan siber dengan tujuan membuat komputer atau perangkat lain tidak dapat diakses oleh pengguna yang dituju dengan cara mengganggu fungsi normal perangkat tersebut. (Cloudflare, t.t.)

DDoS adalah jenis serangan DoS yang berasal dari banyak sumber terdistribusi, seperti *botnet (Cloudflare, t.t.)*. Dengan menggunakan banyak perangkat yang terinfeksi *malware*, pelaku dapat mengirimkan jumlah permintaan yang sangat besar secara simultan untuk melumpuhkan target.

Serangan DoS dapat dikategorikan menjadi 2, yaitu:

1) *Buffer overflow attacks*

   Jenis serangan yang menyebabkan luapan *buffer* memori yang dapat menyebabkan mesin menghabiskan semua ruang *hard disk*, memori, atau waktu CPU yang tersedia.

2) *Flood attacks*

   Jenis serangan dengan memenuhi server target dengan paket dalam jumlah yang sangat banyak, pelaku kejahatan dapat memenuhi kapasitas server secara berlebihan, yang mengakibatkan penolakan layanan.

Indikator serangan DoS meliputi:

1) Performa jaringan yang sangat lambat seperti waktu muat yang lama untuk berkas atau situs web.
2) Ketidakmampuan untuk memuat situs web tertentu seperti properti web Anda.
3) Kehilangan konektivitas secara tiba-tiba di seluruh perangkat pada jaringan yang sama.

2.1.3 *Brute Force*

Brute Force adalah metode peretasan yang menggunakan coba-coba untuk memecahkan kata sandi, kredensial *login*, dan kunci enkripsi. (Fortinet, t.t.)

Jenis serangan *brute force*:

1) Serangan *brute force* tradisional

   Penyerang mencoba semua kemungkinan kombinasi kata sandi satu per satu tanpa memanfaatkan pola tertentu.

2) Serangan kamus (*dictionary attack*)

   Menggunakan daftar kata yang sudah ada sebelumnya (*wordlist*) yang berisi kombinasi kata sandi umum atau kata sandi yang pernah bocor.

3) Serangan *brute force* terbalik (*reverse brute force attack*)

   Penyerang memulai dengan kata sandi tertentu yang umum (seperti "123456") dan mencocokkannya dengan berbagai akun hingga menemukan kecocokan.

4) *Credential stuffing*

   Memanfaatkan kredensial (*username* dan *password*) yang bocor dalam serangan sebelumnya untuk mencoba masuk ke akun di layanan lain.

5) *Hybrid brute force attack*

   Kombinasi antara serangan kamus dan *brute force*, di mana penyerang

menggunakan daftar kata sandi umum dan memvariasikannya dengan angka atau simbol.

2.1.4 *Network Scanning*

Network Scanning adalah proses pemecahan masalah untuk menemukan kerentanan pada perangkat aktif di sistem Anda. Pemindaian ini mengidentifikasi dan memeriksa perangkat yang terhubung dengan menerapkan satu atau beberapa fitur dalam protokol jaringan. Fitur-fitur ini menangkap sinyal kerentanan dan memberi Anda umpan balik tentang status keamanan jaringan Anda. (Odogwu, 2021)

Jenis-jenis *network scanning*:

1) *Port Scanning*
2) *Vulnerability Scanning*
3) *Ping Sweep*
4) *Service Scannig*
5) *Network Mapping*

2.1.5 *Intrusion Detection System*

Intrusion Detection System adalah sebuah perangkat lunak yang bertugas untuk memantau sebuah jaringan komputer untuk aktivitas jahat atau serangan yang bertujuan untuk mencuri atau menyensor informasi atau merusak protokol jaringan. (Zamani, 2013)

Berdasarkan cakupan pemantauan, IDS terbagi menjadi:

1) *Host-Based* IDS

   Jenis sistem deteksi intrusi yang dipasang langsung pada komputer atau *host* tertentu. Sistem ini bertugas untuk memantau aktivitas mencurigakan pada perangkat tersebut, termasuk perubahan *file*, *log*, atau proses yang berjalan di sistem operasi. (Rastogi dkk., 2022)

2) *Network-Based* IDS

   Jenis sistem deteksi intrusi yang ditempatkan pada level jaringan, biasanya di titik masuk jaringan internal. Fungsinya untuk memantau aktivitas mencurigakan yang terjadi dalam jaringan, seperti serangan dari luar atau aktivitas tidak normal antar perangkat dalam jaringan. (Rastogi dkk., 2022)

Berdasarkan metode deteksi, IDS dapat menggunakan:

1) *Signature-Based Detection*

   Metode deteksi intrusi yang didasarkan pada gagasan untuk mendefinisikan tanda tangan untuk pola serangan. Tanda tangan ini disimpan dalam basis data tanda tangan dan pola data yang akan diperiksa dicocokkan dengan tanda tangan yang tersimpan untuk deteksi serangan.

   Keunggulan dari metode ini adalah sangat efisien untuk mengenali serangan yang diketahui, karena tersedianya tanda tangan untuk serangan tersebut. Kelemahannya adalah tidak memiliki kemampuan untuk mendeteksi serangan baru karena tidak adanya pola tanda tangan. (Ahmad dkk., 2021)

2) *Anomaly-Based Detection*

   Metode deteksi intrusi yang didasarkan pada gagasan untuk mendefinisikan profil aktivitas normal secara jelas. Setiap penyimpangan dari profil normal akan dianggap sebagai anomali atau perilaku abnormal.

   Keunggulan dari metode ini adalah kemampuannya untuk mendeteksi serangan yang tidak diketahui dan baru serta sifat profil aktivitasnya, membuatnya dapat digunakan untuk berbagai jenis jaringan dan aplikasi. Kelemahan utamanya adalah tingkat alarm palsu yang tinggi, karena sulit untuk menemukan batas antara profil normal dan abnormal untuk deteksi intrusi. (Ahmad dkk., 2021)

2.1.6 *Machine Learning*

Machine learning adalah suatu teknik yang membuat komputer dilatih agar memiliki kemampuan untuk secara otomatis mempelajari dan meningkatkan atau mengoptimalkan kriteria kinerja tanpa diprogram secara eksplisit, dengan menggunakan pengalaman masa lalu atau contoh data. (Musa dkk., 2022)

Salah satu jenis *machine learning* yang digunakan dalam penelitian ini adalah *supervised learning*. *Supervised learning* adalah metode pembelajaran mesin di mana model dilatih menggunakan *dataset* yang berlabel. Setiap sampel data memiliki pasangan *input* dan *output* yang sesuai, sehingga model belajar memetakan *input* ke *output*. (Géron, 2019)

2.1.7 *Siklus Hidup Machine Learning*

Siklus hidup *machine learning* adalah proses yang memandu pengembangan

dan penyebaran model *machine learning* dalam jalur yang terstruktur. Siklus hidup *machine learning* terdiri dari berbagai langkah. Setiap langkah memainkan peran penting dalam memastikan keberhasilan dan efektivitas model pembelajaran mesin. (sagar99, 2025)

Langkah-langkah siklus hidup *machine learning*:

1) Definisi masalah

   Pada tahap ini, kita butuh mengidentifikasi masalah bisnis dan menyusunnya. Elemen penting yang harus didefinisikan seperti tujuan proyek, hasil yang diharapkan, dan ruang lingkup tugas.

2) Pengumpulan data

   Pada tahap ini melibatkan pengumpulan data secara sistematis yang dapat digunakan sebagai data mentah untuk melatih model. Kualitas dan keragaman data yang dikumpulkan secara langsung memengaruhi ketahanan dan generalisasi model. Beberapa fitur dasar data yang dikumpulkan: relevansi, kuantitas, kualitas, dan keragaman.

3) Pembersihan dan pra-pemrosesan data

   Setelah data dikumpulkan, data mentah sering kali berantakan dan tidak terstruktur dan data ini tidak bisa digunakan secara langsung untuk pelatihan model, karena bisa menyebabkan akurasi yang buruk dan menangkap hubungan yang tidak perlu dalam data. Pada tahap pembersihan, masalah yang ditangani seperti nilai yang hilang, *outlier* (nilai ekstrem), dan inkonsistensi dalam data.

   Pra-pemrosesan dilakukan dengan melakukan standardisasi format, penskalaan nilai dan mengodekan variabel kategoris yang menciptakan kumpulan data yang konsisten dan terorganisir dengan baik.

4) Analisis eksplorasi data (EDA)

   Pada tahap ini, kita mencari pola dan karakteristik yang tersembunyi dalam data melalui visualisasi data yang sudah diproses. Wawasan yang ditemukan membantu dalam membuat pilihan dalam rekayasa fitur, pemilihan model, dan aspek penting lainnya.

5) Rekayasa dan pemilihan fitur

   Tahap ini adalah proses transformatif yang melibatkan pemilihan hanya fitur yang relevan untuk prediksi model. Pemilihan fitur menyempurnakan

kumpulan variabel yang mengidentifikasi fitur yang paling relevan untuk meningkatkan efisiensi dan efektivitas model.

Rekayasa fitur melibatkan pemilihan fitur yang relevan atau pembuatan fitur baru dengan mengubah fitur yang sudah ada untuk prediksi. Proses kreatif ini memerlukan keahlian domain dan pemahaman mendalam tentang masalah yang memastikan bahwa fitur yang direkayasa memberikan kontribusi yang berarti untuk prediksi model. Ini membantu akurasi sambil meminimalkan kompleksitas komputasi.

6) Pemilihan model

   Pada tahap ini dilakukan pemilihan model untuk menemukan model yang selaras dengan masalah yang sudah ditentukan dan karakteristik dari data yang dikumpulkan.

7) Pelatihan model

   Proses ini melibatkan pemaparan model terhadap data historis yang memungkinkannya mempelajari pola, hubungan, dan dependensi dalam kumpulan data.

8) Pelatihan model adalah proses berulang di mana algoritma menyesuaikan parameternya untuk meminimalkan kesalahan dan meningkatkan akurasi prediktif. Selama fase ini, model menyempurnakan dirinya sendiri untuk pemahaman data yang lebih baik dan mengoptimalkan kemampuannya untuk membuat prediksi. Proses pelatihan yang ketat memastikan bahwa model yang dilatih bekerja dengan baik dengan data baru yang tak terlihat untuk prediksi yang andal dalam skenario dunia nyata.

9) Evaluasi dan penyetelan model

   Evaluasi model melibatkan pengujian ketat terhadap *dataset* validasi atau pengujian untuk menguji keakuratan model pada data baru yang tidak terlihat. Kita dapat menggunakan teknik seperti keakuratan, presisi, recall, dan F1-score untuk memeriksa efektivitas model.

   Evaluasi sangat penting untuk memberikan wawasan tentang kekuatan dan kelemahan model. Jika model gagal mencapai tingkat kinerja yang diinginkan, kita mungkin perlu menyetel model lagi dan menyesuaikan *hyperparameter*-nya untuk meningkatkan keakuratan prediktif. Siklus evaluasi dan penyetelan

berulang ini sangat penting untuk mencapai tingkat ketahanan dan keandalan model yang diinginkan.

10) Penerapan model

Setelah evaluasi berhasil, model pembelajaran mesin siap digunakan untuk aplikasi di dunia nyata. Penggunaan model melibatkan pengintegrasian model prediktif dengan sistem yang ada yang memungkinkan bisnis menggunakannya untuk pengambilan keputusan yang tepat.

2.1.8 *Feature Selection for Network Intrusion Detection (FSNID)*

FSNID dibangun di atas prinsip teori informasi , terutama konsep *Mutual Information* (MI) dan *Transfer Entropy* (TE), untuk mengukur relevansi fitur terhadap label serangan dalam deteksi intrusi jaringan (*Network Intrusion Detection*/NID). (Westphal dkk., 2024)

1) *Mutual Information* (MI)

MI mengukur seberapa besar ketidakpastian tentang label serangan (Y) berkurang dengan mengetahui fitur X. Rumusnya:

$$I(X;Y) = H(X) - H(X|Y)$$

Semakin tinggi nilai MI, semakin informatif fitur X terhadap label Y.

2) *Transfer Entropy* (TE):

TE mengukur aliran informasi dari fitur Xi ke label serangan Y. Rumusnya:

$$\Phi X\_i; X \rightarrow Y = H(Y \mid X \setminus X\_i\ ) - H(Y \mid X)$$

Jika ΦX_i;X→Y=0, fitur Xi dianggap tidak informatif dan dihapus.

FSNID secara eksplisit mengatasi masalah *redundant* (fitur yang saling berkorelasi tinggi) dengan pendekatan *sequential*:

1) Jika fitur Xi dihapus karena tidak ΦX_i;X→Y=0, fitur *redundant* lainnya (Xj) akan dinilai ulang berdasarkan *subset* fitur yang tersisa.
2) Hal ini memastikan hanya satu fitur dari kelompok *redundant* yang dipertahankan, mengurangi kompleksitas tanpa kehilangan informasi kritis.

Algoritma FSNID bekerja sebagai berikut:

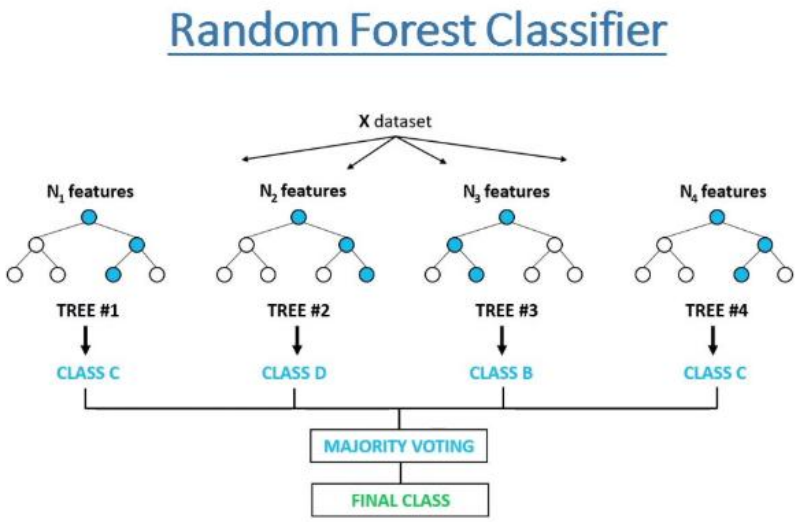
1) Inisialisasi *subset* fitur kosong (X^*=∅).
2) Untuk setiap fitur Xi:
    1) Hitung ΦX_i;X→Y menggunakan estimasi *neural network*.
    2) Jika ΦX_i;X→Y>0, tambahkan Xi ke X*.

3) Jika ΦX_i;X→Y=0, hapus Xi dari *dataset*.

Subset fitur X* menjadi *input* untuk model klasifikasi.

2.1.9 *Random Forest*

Algoritma *Random Forest* merupakan salah satu algoritma klasifikasi terbimbing. Sesuai dengan namanya, algoritma ini bekerja dengan cara membuat hutan secara acak dengan cara yang tepat. *Random forest* merupakan algoritma berbasis pohon keputusan yang digunakan untuk membangun beberapa pohon kemudian menggabungkan *output*-nya untuk meningkatkan kemampuan generalisasi model. Algoritma *Random Forest* merupakan gabungan beberapa pohon individu untuk menghasilkan pohon pembelajar yang kuat. (Hadi, 2018)



Gambar 2. 1 Gambar Algoritma *Random Forest*

Untuk meningkatkan kinerja dari algoritma *Random Forest*, kita bisa menyesuaikan beberapa *hyperparameter* dalam *Random Forest*. *Hyperparameter* adalah parameter yang ditentukan sebelum pelatihan model dan tidak dioptimalkan selama pelatihan.

Beberapa *hyperparameter* yang digunakan dalam *Random Forest*:

1) *n_estimators* (jumlah pohon)
2) *max_depth* (kedalaman maksimum pohon)
3) *min_sample_split* (jumlah minimum sampel untuk membagi simpul)
4) *min_sample_leaf* (jumlah minimum sampel per daun)

2.1.10 *Python*

Python adalah bahasa pemrograman berorientasi objek tingkat tinggi yang canggih yang diciptakan oleh Guido van Rossum. Python dikenal karena sintaknya yang mudah dipahami dan digunakan untuk berbagai aplikasi, mulai dari pengembangan web hingga analisis data dan pembelajaran mesin (Srinath, 2017). Salah satu keunggulan utama Python adalah kemudahan dalam penggunaannya, menjadikannya pilihan yang ideal bagi pemula maupun profesional di bidang pengembangan perangkat lunak (Lutz & Safari, 2013). Selain itu, Python memiliki pustaka standar yang sangat luas, yang mencakup berbagai kebutuhan pemrograman, seperti manipulasi data, pengolahan teks, pengembangan web, kecerdasan buatan, hingga keamanan siber (McKinney, 2017).

Meskipun memiliki banyak keunggulan, Python juga memiliki beberapa keterbatasan. Salah satunya adalah kecepatan eksekusi yang lebih lambat dibandingkan bahasa pemrograman seperti C atau Java karena sifatnya yang interpretatif. Selain itu, konsumsi memori yang lebih tinggi membuat Python kurang optimal untuk aplikasi yang membutuhkan kinerja tinggi dan pemrosesan waktu nyata (Lutz & Safari, 2013).

Gambar 2. 2 Python (Sumber: www.python.org)

2.1.11 *Scikit-Learn*

Scikit-learn adalah modul python yang mengintegrasikan berbagai algoritma pembelajaran mesin canggih untuk masalah pembelajaran mesin skala menengah yang diawasi (*supervised*) dan tidak diawasi (*unsupervised*). Paket ini berfokus untuk menyediakan pembelajaran mesin bagi non-spesialis menggunakan bahasa tingkat tinggi untuk keperluan umum dalam analisis data (Pedregosa dkk., 2011).

Salah satu keunggulan utama Scikit-learn adalah kelengkapan algoritma pembelajaran mesin yang mencakup metode *supervised* seperti *Support Vector Machine* (SVM), *Decision Tree*, *Random Forest*, dan *Naïve Bayes*, serta metode *unsupervised* seperti K-Means, DBSCAN, dan *Principal Component Analysis* (PCA). Selain itu, Scikit-learn juga memiliki fitur untuk validasi model dengan

teknik seperti *cross-validation*, serta mendukung penyetelan parameter model (*hyperparameter tuning*) menggunakan GridSearchCV dan RandomizedSearchCV. (Pedregosa dkk., 2011)

Gambar 2. 3 Scikit-Learn (Sumber: scikit-learn.org)

2.1.12 *Flask*

Flask adalah kerangka kerja web (web *framework*) berbasis Python yang dikenal sebagai *microframework* karena desainnya yang minimalis dan ringan. Flask memberikan fleksibilitas tinggi kepada pengembang dengan menyediakan inti yang sederhana namun dapat diperluas sesuai kebutuhan melalui berbagai ekstensi. Hal ini memungkinkan pengembang untuk membangun aplikasi web dengan cepat tanpa memaksakan struktur atau dependensi tertentu. (Grinberg, 2018)

Keunggulan flask:

1) Sederhana.
2) Fleksibel.
3) Komunitas yang aktif dan ketersediaan berbagai ekstensi.

Komponen utama aplikasi flask:

1) Objek flask yang merupakan inti dari aplikasi dan digunakan untuk mengonfigurasi serta mendaftarkan rute
2) Fungsi yang menangani permintaan ke URL tertentu dan mengembalikan respon ke klien.
3) *Template* atau file HTML yang digunakan untuk merender tampilan dinamis bagi pengguna.
4) *Blueprints* atau komponen modular yang memungkinkan pengorganisasian aplikasi besar menjadi bagian-bagian yang lebih kecil dan terpisah.

Gambar 2. 4 Flask (Sumber: developer.aliyun.com)

2.1.13 *Black Box Testing*

Black box testing adalah metode pengujian perangkat lunak yang berfokus pada pengujian fungsionalitas sistem tanpa memperhatikan struktur internal atau kode sumbernya. Tujuan utama dari teknik ini adalah untuk memastikan bahwa sistem berfungsi sesuai dengan spesifikasi yang telah ditentukan. Pengujian dilakukan dengan memberikan berbagai *input* ke dalam sistem dan mengamati *output* yang dihasilkan. (Verma dkk., 2017)

Keunggulan *black box testing* adalah:

1) Tidak memerlukan pemahaman tentang kode sumber, sehingga dapat dilakukan oleh penguji non-teknis.
2) Menguji perangkat lunak dari perspektif pengguna akhir, memastikan bahwa sistem memenuhi kebutuhan pelanggan.
3) Dapat diterapkan dalam tahap *Integration Testing*, *System Testing*, dan *Acceptance Testing* dalam siklus pengembangan perangkat lunak.

Kelemahan *black box testing* adalah:

1) Dapat menghasilkan pengujian yang berlebihan jika pengembang telah melakukan pengujian serupa.
2) Sulit mendesain kasus uji jika spesifikasi perangkat lunak tidak jelas.
3) Kurang efektif dalam mengidentifikasi kesalahan yang berkaitan dengan struktur internal perangkat lunak.

2.1.14 *Cross-Validation*

Cross-validation merupakan metode *resampling* data yang digunakan untuk mengevaluasi kemampuan generalisasi model prediktif dan mencegah *overfitting* atau *underfitting*. Dalam konteks *supervised learning*, tujuan utama *cross-validation* adalah mengestimasi kesalahan prediksi sebenarnya (*true prediction error*) yang akan terjadi pada data baru yang berasal dari populasi yang sama dengan data pelatihan. Metode ini termasuk dalam keluarga metode *Monte Carlo*, bersama dengan *bootstrap* , dan sangat umum digunakan ketika validasi independen dengan data eksternal tidak memungkinkan. Prinsip dasarnya adalah membagi *dataset* (*learning set*) menjadi subset pelatihan (*training set*) dan *subset* uji (*test*/*validation set*) secara berulang untuk menghasilkan estimasi performa yang

lebih akurat. (Berrar, 2019)

Jenis metode *cross-validation*:

1) *Single Hold-Out Subsampling*

   Metode paling sederhana di mana *dataset* dibagi menjadi satu pasangan *training set* dan *test set* . Misalnya, 70% data digunakan untuk pelatihan dan 30% untuk validasi.

2) *k-Fold Random Subsampling*

   Pengulangan metode *single hold-out* sebanyak k kali, dengan partisi data yang berbeda pada setiap iterasi. *Subset* dapat tumpang tindih antar iterasi.

3) *k-Fold Cross-Validation*

   Dataset dibagi menjadi k *subset* (*folds*) yang saling lepas dan tidak tumpang tindih. Setiap *fold* digunakan sekali sebagai *validation set*, sementara sisanya menjadi *training set*. *Stratified k-Fold* adalah variasi dari *k-Fold cross-validation*. Variasi ini Memastikan proporsi kelas pada setiap *fold* mencerminkan *dataset* asli (penting untuk klasifikasi dengan ketidakseimbangan kelas).

4) *Leave-One-Out Cross-Validation* (LOOCV)

   Kasus khusus dari *k-fold cross-validation* dengan k = n (jumlah data). Setiap iterasi, satu data digunakan sebagai *validation set*, sisanya sebagai *training set*.

5) *Jackknife*

   Metode serupa dengan LOOCV, tetapi fokus pada estimasi bias dan varians suatu statistik ($\hat{\theta}$), bukan performa model prediktif.

2.1.15 *Metrik Evaluasi*

Metrik evaluasi adalah parameter atau ukuran yang digunakan untuk mengkuantifikasi kualitas penjelasan (*explanations*) dalam sistem *machine learning* (ML). (Zhou dkk., 2021)

Metrik evaluasi yang digunakan dalam klasifikasi:

1) Akurasi

   Persentase prediksi yang benar dari total data.

$$akurasi = \frac{TP + TN}{TP + TN + FP + FN}$$

2) Presisi

Kemampuan model untuk hanya memberikan prediksi positif yang benar.

$$presisi = \frac{TP}{TP + FP}$$

3) Recall

Kemampuan model untuk mendeteksi semua sampel positif.

$$recall = \frac{TP}{TP + FN}$$

4) F1-score

Kombinasi *harmonic mean* dari presisi dan recall.

$$F1\ Score = \frac{2 * (Precision * Recall)}{Precision + Recall}$$

5) *Area Under the Curve* (AUC)

Mengukur kemampuan model dalam membedakan antara kelas positif dan negatif pada berbagai ambang batas.

Penjelasan nilai dari AUC:

1) AUC = 1.0: Model sempurna.
2) AUC = 0.5: Model tidak lebih baik dari tebakan acak.
3) AUC < 0.5: Model lebih buruk dari tebakan acak.

2.1.16 *Confusion Matrix*

Confusion matrix adalah tabel 2x2 (untuk klasifikasi biner) atau tabel NxN (untuk klasifikasi multi-kelas) yang merangkum hasil prediksi model terhadap label yang sebenarnya. Matriks ini membantu dalam memahami kesalahan model secara lebih rinci dibandingkan hanya menggunakan akurasi.

Tabel 2. 1 Tabel *Confusion Matrix*

	Prediksi Positif	Prediksi Negatif
Label Positif	*True Positive* (TP)	*False Negative* (FN)
Label Negatif	*False Positive* (FP)	*True Negative* (TN)

Komponen utama dari *confusion matrix* pada Tabel 2. 1:

1) *True positive* (TP) adalah kondisi dimana prediksi model adalah positif, dan label sebenarnya juga positif.
2) *True negative* (TN) adalah kondisi dimana prediksi model adalah negatif,

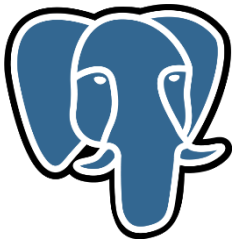
dan label sebenarnya juga negatif.

3) *False positive* (FP) adalah kondisi dimana prediksi model adalah positif, dan label sebenarnya juga negatif.
4) *False negative* (FN) adalah kondisi dimana prediksi model adalah negatif, dan label sebenarnya juga positif.

2.1.17 *PostgreSQL*

PostgreSQL merupakan sistem manajemen basis data relasional objek (Object-Relational Database Management System atau ORDBMS) yang bersifat *open source*. PostgreSQL dikembangkan oleh PostgreSQL Global Development Group dan dirancang untuk mendukung standar SQL serta berbagai fitur lanjutan, seperti transaksi, *foreign key*, *view*, *trigger*, *stored procedure*, dan berbagai tipe data yang dapat dikembangkan sesuai kebutuhan. Selain itu, PostgreSQL mendukung konsep *Multi-Version Concurrency Control* (MVCC) sehingga beberapa pengguna dapat mengakses dan memodifikasi data secara bersamaan tanpa mengurangi konsistensi data. Fitur-fitur tersebut menjadikan PostgreSQL sebagai salah satu sistem basis data yang banyak digunakan dalam pengembangan aplikasi berskala kecil hingga perusahaan. (The PostgreSQL Global Development Group, 2026)

PostgreSQL juga dikenal memiliki tingkat keandalan, keamanan, dan fleksibilitas yang tinggi karena mendukung transaksi yang memenuhi prinsip ACID (*Atomicity*, *Consistency*, *Isolation*, dan *Durability*). Selain itu, PostgreSQL menyediakan berbagai mekanisme untuk menjaga integritas data, mendukung berbagai bahasa pemrograman, serta dapat dijalankan pada berbagai sistem operasi, seperti Linux, Windows, dan macOS. Kemampuan tersebut membuat PostgreSQL banyak dimanfaatkan sebagai sistem basis data pada aplikasi web, sistem informasi, analisis data, hingga layanan berbasis *cloud*. (The PostgreSQL Global Development Group, 2026)



Gambar 2. 5 PostgreSQL (Sumber: www.postgresql.com)

2.1.18 *VirtualBox*

Oracle VirtualBox merupakan perangkat lunak virtualisasi lintas platform (*cross-platform virtualization application*) yang memungkinkan pengguna menjalankan beberapa sistem operasi secara bersamaan dalam satu komputer fisik melalui mesin virtual (*Virtual Machine* atau VM). VirtualBox bekerja sebagai *hosted hypervisor* (*hypervisor* tipe 2), yaitu berjalan di atas sistem operasi utama (*host operating system*) dan menyediakan lingkungan virtual yang dapat digunakan untuk menginstal sistem operasi lain tanpa memengaruhi sistem utama. Selain mendukung berbagai sistem operasi tamu (*guest operating system*), VirtualBox juga memungkinkan pengguna membuat, menyimpan, menyalin, mengekspor, dan memindahkan mesin virtual ke perangkat lain. (Oracle, 2026)

VirtualBox menyediakan berbagai fitur yang mendukung proses pengembangan dan pengujian sistem, seperti *snapshot*, *shared folder*, *virtual networking*, serta dukungan terhadap format *Open Virtualization Format* (OVF). Fitur-fitur tersebut memudahkan pengguna dalam melakukan simulasi server, pengujian aplikasi, maupun pembelajaran sistem operasi tanpa harus menyediakan perangkat keras tambahan. Oleh karena itu, VirtualBox banyak dimanfaatkan pada lingkungan pendidikan, penelitian, maupun industri sebagai platform virtualisasi yang efisien dan fleksibel. (Oracle, 2026)

2.1.19 *Windows Subsystem for Linux* (WSL)

Windows Subsystem for Linux (WSL) merupakan fitur pada sistem operasi Microsoft Windows yang memungkinkan pengguna menjalankan lingkungan Linux secara langsung di dalam Windows tanpa memerlukan mekanisme *dual boot* maupun mesin virtual (*virtual machine*) konvensional. Melalui WSL, pengguna dapat menginstal berbagai distribusi Linux, seperti Ubuntu, Debian, Kali Linux, dan OpenSUSE, kemudian menjalankan utilitas Linux, shell Bash, serta aplikasi berbasis Linux secara berdampingan dengan aplikasi Windows. Kehadiran WSL bertujuan untuk memberikan lingkungan kerja yang lebih efisien bagi pengembang maupun administrator sistem yang membutuhkan kompatibilitas antara ekosistem Windows dan Linux. (Microsoft, 2025)

WSL versi 2 menggunakan teknologi virtualisasi ringan (*lightweight utility virtual machine*) dengan *kernel* Linux asli yang dikembangkan dan dipelihara oleh Microsoft. Arsitektur tersebut memberikan kompatibilitas penuh terhadap *system call* Linux, meningkatkan performa sistem berkas dibandingkan WSL generasi pertama, serta mendukung berbagai layanan dan aplikasi Linux tanpa memerlukan modifikasi. Selain itu, WSL memungkinkan integrasi antara sistem operasi Windows dan Linux melalui akses berkas, jaringan, maupun pemanggilan aplikasi lintas platform sehingga banyak dimanfaatkan dalam pengembangan perangkat lunak, administrasi server, komputasi awan, dan pengujian aplikasi. (Microsoft, 2025)

2.1.20 *Suricata*

Suricata merupakan perangkat lunak *open source* yang dikembangkan oleh *Open Information Security Foundation* (OISF) sebagai mesin *Intrusion Detection System* (IDS), *Intrusion Prevention System* (IPS), dan *Network Security Monitoring* (NSM). Suricata dirancang untuk melakukan inspeksi lalu lintas jaringan secara *real time* dengan mendeteksi aktivitas yang mencurigakan berdasarkan aturan (*rule-based detection*), analisis protokol jaringan, serta pemeriksaan isi paket (*deep packet inspection*). Kemampuan tersebut memungkinkan Suricata mengidentifikasi berbagai ancaman keamanan, seperti serangan jaringan, eksploitasi kerentanan, dan aktivitas berbahaya lainnya secara cepat dan akurat. (Open Information Security Foundation, 2026)

Selain berfungsi sebagai IDS dan IPS, Suricata menyediakan fitur pencatatan (*logging*) dalam format EVE JSON yang memudahkan integrasi dengan berbagai platform analisis keamanan, seperti Elasticsearch, Logstash, Kibana (ELK), maupun *Security Information* and *Event Management* (SIEM). Suricata juga mendukung pemrosesan *multi-threading*, sehingga mampu memanfaatkan prosesor *multicore* untuk meningkatkan kinerja analisis lalu lintas jaringan berkecepatan tinggi. Berbagai kemampuan tersebut menjadikan Suricata sebagai salah satu solusi keamanan jaringan yang banyak digunakan pada lingkungan penelitian, institusi pendidikan, maupun perusahaan. (Open Information Security Foundation, 2026)

2.2 Penelitian Terdahulu

Tinjauan penelitian terdahulu ini membantu peneliti mengidentifikasi kesenjangan penelitian yang belum terjawab, sehingga penelitian yang dilakukan memiliki kontribusi yang unik. Selain itu, tinjauan pusaka membangun dasar teoritis yang kuat dengan memahami konsep, teori, atau model yang relevan sebagai landasan penelitian. Proses ini juga memvalidasi pentingnya masalah penelitian dengan menunjukkan bahwa isu yang diangkat sudah menjadi perhatian dalam penelitian sebelumnya.

Penelitian pertama yang ditinjau, yaitu penelitian dari Sari dkk. (2024) dengan judul "*Implementasi Machine Learning untuk Deteksi Intrusi pada Jaringan Komputer*". Penelitian ini mengembangkan dan mengevaluasi model *machine learning* deteksi intrusi pada jaringan komputer, dengan fokus pada tiga algoritma utama: *Decision Tree*, *Random Forest*, dan *Support Vector Machine* (SVM). *Dataset* yang digunakan dalam penelitian ini adalah KDD CUP 99, NSL KDD, dan CIC-IDS 2017 (Sharafaldin dkk., 2018). Berdasarkan hasil penelitian, ketiga model *machine learning* yang diuji menunjukkan kinerja yang baik dalam mendeteksi intrusi, dengan *Random Forest* secara konsisten memberikan hasil terbaik. Model ini mencapai akurasi 96.8%, presisi 95.5%, recall 94.7%, dan F1-score 91.5%, yang mengungguli *Decision Tree* dan SVM. Penelitian ini mengindikasikan bahwa *Random Forest* lebih efektif dalam mendeteksi intrusi pada jaringan, mengingat kemampuannya untuk mengatasi kompleksitas data dan mengurangi risiko *overfitting*, serta lebih andal dalam membedakan antara lalu lintas jaringan normal dan anomali.

Penelitian kedua adalah penelitian dari Mohammed & Talib (2024) dengan judul "*Using Machine Learning Algorithms in Intrusion Detection Systems: A Review*" membahas penerapan algoritma pembelajaran mesin (*machine learning*) untuk sistem deteksi intrusi (IDS) dalam jaringan *Internet of Things* (IoT). Penelitian ini mengulas tantangan khusus yang dihadapi lingkungan IoT, seperti heterogenitas, keterbatasan sumber daya, deteksi secara *real-time*, serta ancaman terhadap privasi dan serangan *adversarial*. Selain itu, penelitian ini mengkaji berbagai algoritma pembelajaran mesin, termasuk *supervised*, *unsupervised*, *semi supervised learning*, serta arsitektur pembelajaran mendalam seperti *Decision Tree*,

Random Forest, *Support Vector Machines*, dan *Deep Learning*. Penelitian ini bersifat tinjauan pustaka (*review*) sehingga lebih banyak membahas studi literatur yang ada daripada melakukan eksperimen langsung, temuan dari tinjauan ini menunjukkan bahwa meskipun ada tantangan yang signifikan dalam penerapan IDS pada IoT, algoritma *machine learning* menawarkan solusi yang menjanjikan untuk meningkatkan keamanan dan deteksi intrusi dalam sistem IoT. Penelitian ini menyarankan untuk menerapkan metode *ensemble* dan teknik *feature selection* untuk meningkatkan akurasi dan efektivitas sistem deteksi intrusi. Namun, tanpa eksperimen praktis atau pengujian terhadap *dataset* spesifik, efektivitas algoritma yang dibahas mungkin belum sepenuhnya diuji dalam berbagai skenario dunia nyata.

Penelitian ketiga adalah penelitian dari Sabha (2020) dengan judul "*Anomaly based Intrusion Detection using Machine Learning Algorithms - A Review Paper*" yang membahas tinjauan literatur tentang sistem deteksi intrusi (IDS) berbasis anomali menggunakan algoritma pembelajaran mesin. Fokus utama penelitian ini adalah memberikan gambaran tentang berbagai teknik IDS, algoritma pembelajaran mesin yang digunakan, dan *dataset* yang relevan, seperti KDD-Cup99 dan NSL KDD. Penelitian ini menunjukkan bahwa setiap teknik deteksi memiliki kelebihan dan kekurangan, dan pemilihan algoritma yang tepat sangat penting untuk mencapai hasil yang optimal. Beberapa algoritma pembelajaran mesin yang dianalisis, termasuk *Decision Tree* (DT), *Naïve Bayes*, *Support Vector Machine* (SVM), dan *Artificial Neural Networks* (ANN), menunjukkan potensi yang signifikan dalam mendeteksi intrusi. Penelitian ini juga menyoroti keunggulan *anomaly-based* IDS, yang lebih efektif dalam mendeteksi serangan baru dibandingkan dengan pendekatan berbasis tanda tangan. Selain itu, analisis terhadap *dataset* KDD-Cup99 dan versi terbarunya, NSL-KDD, menunjukkan bahwa NSL-KDD mengatasi beberapa masalah yang ada pada *dataset* KDD-Cup99, seperti ketidakseimbangan data dan *redundansi*. Penelitian ini menekankan bahwa meskipun NSL-KDD sering digunakan oleh peneliti untuk membandingkan berbagai model deteksi intrusi, *dataset* ini masih dianggap kurang representatif untuk ancaman siber modern. Jurnal ini bersifat tinjauan pustaka tanpa eksperimen praktis atau implementasi dari algoritma yang dibahas, sehingga penelitian ini lebih fokus pada analisis teoritis

dan kurang memberikan kontribusi langsung pada pengembangan sistem IDS praktis.

Penelitian keempat adalah penelitian dari Sarthak Rastogi dkk. (2022) dengan judul "*An Analysis of Intrusion Detection Classification using Supervised Machine Learning Algorithms on NSL-KDD Dataset*" membahas analisis sistem deteksi intrusi (IDS) berbasis algoritma pembelajaran mesin terawasi (*supervised machine learning*) dengan menggunakan *dataset* NSL-KDD. Fokus utama penelitian adalah mengevaluasi dan membandingkan kinerja beberapa algoritma pembelajaran mesin, termasuk *k-Nearest Neighbour* (KNN), *Support Vector Machine* (SVM), *Random Forest* (RF), *Logistic Regression* (LR), *Naive Bayes* (NB), dan *Decision Tree* (DT), dalam mendeteksi lalu lintas jaringan yang abnormal. Hasil penelitian ini menunjukkan bahwa algoritma *Random Forest* memberikan akurasi tertinggi pada kedua jenis klasifikasi, sedangkan KNN menunjukkan waktu pengujian yang lebih lama tetapi tetap memberikan akurasi yang baik. *Decision Tree* menunjukkan kinerja paling buruk dalam penelitian ini. Penelitian hanya menggunakan *dataset* NSL-KDD, yang meskipun populer tetapi sudah dianggap kurang representatif untuk ancaman siber modern.

Penelitian kelima adalah penelitian oleh Morshedi dkk. (2023) dengan judul "*Intrusion Detection for IoT Network Security with Deep Neural Network*" membahas pengembangan model deteksi intrusi berbasis *Deep Neural Networks* (DNN) untuk keamanan jaringan *Internet of Things* (IoT) dengan menggunakan *dataset* CIC-IDS 2017. Penelitian ini fokus pada deteksi serangan *Distributed Denial of Service* (DDoS) serta serangan siber lainnya yang umum terjadi pada jaringan IoT. Beberapa arsitektur jaringan yang digunakan mencakup *Convolutional Neural Networks* (CNN), *DenseNet*, dan model *hybrid* CNN-LSTM. Fokus utama penelitian adalah pada akurasi, dengan hasil yang menunjukkan bahwa model yang diusulkan memberikan kinerja terbaik dibandingkan dengan model *deep learning* lainnya dan algoritma *machine learning*, dengan mencapai akurasi 99,7%. Analisis juga menunjukkan bahwa model yang diusulkan dapat mendeteksi berbagai jenis serangan siber yang menargetkan perangkat IoT. Namun, penelitian ini tidak membahas secara mendalam tentang efisiensi waktu pelatihan, waktu inferensi, atau penggunaan memori, yang penting untuk implementasi di

perangkat IoT.

Tabel 2. 2 Perbandingan Penelitian Terdahulu

No.	Judul Penelitian	Fokus Penelitian	*Dataset* yang digunakan	Hasil Penelitian
1	*Implementasi Machine Learning untuk Deteksi Intrusi pada Jaringan Komputer*	Pengembangan dan evaluasi model *machine learning* untuk deteksi intrusi dengan algoritma *Decision Tree*, *Random Forest*, dan SVM.	KDD CUP 99, NSL KDD, CIC-IDS 2017	*Random Forest* memberikan akurasi tertinggi 96.8%, presisi 95.5%, recall 94.7%, F1-score 91.5%. *Random Forest* lebih efektif dalam mendeteksi intrusi dan mengurangi risiko *overfitting*.
2	*Using Machine Learning Algorithms in Intrusion Detection Systems: A Review*	Tinjauan literatur penerapan algoritma *machine learning* dalam IDS untuk IoT, termasuk tantangan seperti heterogenitas, keterbatasan sumber daya, dan privasi.		Menyimpulkan bahwa algoritma *machine learning* menawarkan solusi yang menjanjikan untuk IDS pada IoT, dengan rekomendasi untuk menggunakan metode *ensemble* dan teknik *feature selection* untuk meningkatkan akurasi dan efektivitas, meskipun tidak ada eksperimen praktis.
3	*Anomaly based Intrusion Detection using Machine Learning Algorithms - A Review Paper*	Tinjauan literatur tentang IDS berbasis anomali, dengan analisis berbagai algoritma *machine learning* dan *dataset* terkait, serta keunggulan *anomaly-*	KDD-Cup99, NSL-KDD	Menyimpulkan bahwa *anomaly-based* IDS lebih efektif dalam mendeteksi serangan baru. NSL-KDD

No.	Judul Penelitian	Fokus Penelitian	*Dataset* yang digunakan	Hasil Penelitian
		based IDS dalam mendeteksi serangan baru		mengatasi beberapa masalah KDD-Cup99, tetapi tetap dianggap kurang representatif untuk ancaman siber modern. Tanpa eksperimen praktis atau implementasi algoritma, lebih fokus pada analisis teoritis.
4	*An Analysis of Intrusion Detection Classification using Supervised Machine Learning Algorithms on NSL-KDD Dataset*	Menganalisis dan membandingkan kinerja beberapa algoritma *supervised machine learning* dalam mendeteksi intrusi menggunakan *dataset* NSL-KDD	NSL-KDD	*Random Forest* memberikan akurasi tertinggi, sedangkan KNN menunjukkan waktu pengujian yang lebih lama namun akurasi baik. *Decision Tree* menunjukkan kinerja paling buruk. NSL-KDD dianggap kurang representatif untuk ancaman siber modern.
5	*Intrusion Detection for IoT Network Security with Deep Neural Network*	Mengembangkan model deteksi intrusi berbasis *Deep Neural Networks* (DNN) untuk keamanan IoT, fokus pada deteksi DDoS dan serangan siber lainnya. Menggunakan beberapa arsitektur jaringan seperti CNN, *DenseNet*, dan CNN-LSTM.	CIC-IDS 2017	Model yang diusulkan memberikan akurasi terbaik 99.7% dibandingkan dengan model *deep learning* lainnya dan algoritma *machine learning*. Dapat mendeteksi

No.	Judul Penelitian	Fokus Penelitian	*Dataset* yang digunakan	Hasil Penelitian
				berbagai serangan siber di perangkat IoT, namun tidak membahas efisiensi waktu pelatihan dan penggunaan memori untuk implementasi di perangkat IoT.

Penelitian ini bertujuan mengimplementasikan sistem deteksi intrusi berbasis *machine learning* untuk mengidentifikasi serangan *cyber* pada jaringan menggunakan algoritma *Random Forest* dengan memanfaatkan *dataset* CIC-IDS 2017. Dalam penelitian ini, peneliti melakukan pra-pemrosesan data yang meliputi pembersihan (menangani nilai hilang dan duplikat), normalisasi fitur numerik, serta seleksi fitur kritis dari *dataset* CIC-IDS 2017 yang mencakup serangan modern seperti *Brute Force*, *Heartbleed*, dan *Botnet*. *Dataset* ini dipilih karena merepresentasikan lalu lintas jaringan nyata (*real-world PCAP traces*) dengan 80+ fitur statistik aliran (*flow-based*), seperti ukuran paket, interval waktu, dan pola komunikasi, yang memungkinkan analisis perilaku serangan secara komprehensif. Untuk mengatasi ketidakseimbangan kelas (*class imbalance*), data dibagi menggunakan *stratified sampling* dan dilakukan teknik *resampling* selektif pada *subset* data latih.