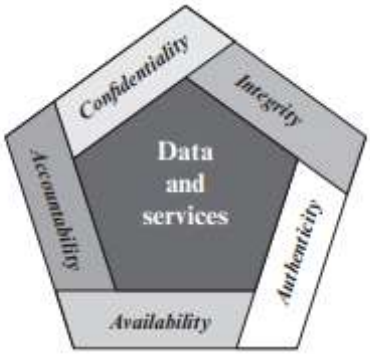
BAB II

TINJAUAN PUSTAKA

2.1 Keamanan Jaringan

Menurut (Stallings & S, 2021) dengan judul buku *"Network Security Essentials sixth edition"*. Keamanan jaringan mempunyai peran sebagai perlindungan yang diberikan pada sistem informasi otomatis untuk mencapai tujuan yang tepat untuk menjaga integritas, ketersediaan, dan kerahasiaan sumber daya sistem informasi (termasuk perangkat lunak, *firmware*, informasi atau data, dan telekomunikasi). Prinsip dasar keamanan informasi yang dikenal sebagai *CIA Triad* harus diterapkan dalam upaya menjaga keamanan, keandalan, dan integritas jaringan. Prinsip ini bertujuan untuk memberikan panduan bagi orang-orang dalam mengembangkan aplikasi, prosedur, atau kebijakan yang berkaitan dengan keamanan informasi untuk membantu mereka menghindari berbagai serangan yang ada.



Gambar 2. 1 Data and Service

a. Kerahasiaan: Menjaga pembatasan resmi pada akses dan pengungkapan informasi, termasuk cara untuk melindungi privasi dan informasi kepemilikan. Hilangnya kerahasiaan adalah pengungkapan yang tidak sah.
b. Integritas: Menjaga terhadap modifikasi atau perusakan informasi yang tidak semestinya, termasuk memastikan bahwa informasi tidak palsu atau palsu. Hilangnya integritas adalah modifikasi atau perusakan yang tidak sah.
c. Ketersediaan: Memastikan bahwa informasi dapat diakses dan digunakan dengan cepat dan andal. Kerugian ketersediaan adalah ketika tidak dapat mengakses atau menggunakan sistem informasi atau informasi.

Ketiga konsep ini membentuk apa yang sering disebut sebagai *triad CIA*. Ketiga konsep tersebut mewujudkan tujuan keamanan mendasar untuk data dan layanan informasi serta komputasi. Misalnya, *Standar NIST* untuk Kategorisasi Keamanan Informasi Federal dan Sistem Informasi (*FIPS* 199) mencantumkan kerahasiaan, integritas, dan ketersediaan sebagai tiga tujuan keamanan untuk informasi dan sistem informasi. *FIPS* 199 memberikan karakterisasi yang berguna dari ketiga tujuan ini dalam hal persyaratan dan definisi hilangnya keamanan dalam setiap kategori.

a. Kerahasiaan: Mempertahankan pembatasan yang sah atas akses dan pengungkapan informasi, termasuk cara untuk melindungi privasi pribadi dan informasi hak milik. Hilangnya kerahasiaan adalah pengungkapan informasi yang tidak sah.
b. Integritas: Melindungi dari modifikasi atau penghancuran informasi yang tidak semestinya, termasuk memastikan informasi tidak dapat disangkal dan

keasliannya. Hilangnya integritas adalah modifikasi atau penghancuran informasi yang tidak sah.

c. Ketersediaan: Memastikan akses dan penggunaan informasi yang tepat waktu dan andal. Hilangnya ketersediaan adalah gangguan akses atau penggunaan informasi atau sistem informasi.

Meskipun *triad CIA* sering digunakan untuk menentukan tujuan keamanan, ada beberapa orang di bidang keamanan yang merasa bahwa konsep tambahan diperlukan untuk memberikan informasi yang lebih baik. Dua yang paling umum adalah:

a. *Authenticity*: kualitas yang asli dan dapat diverifikasi dan dipercaya; kepercayaan terhadap transmisi, pesan, atau pembuat pesan; ini berarti memverifikasi bahwa pengguna benar-benar sesuai dengan apa yang mereka katakan dan bahwa setiap masukan yang masuk ke sistem berasal dari sumber yang dapat dipercaya.

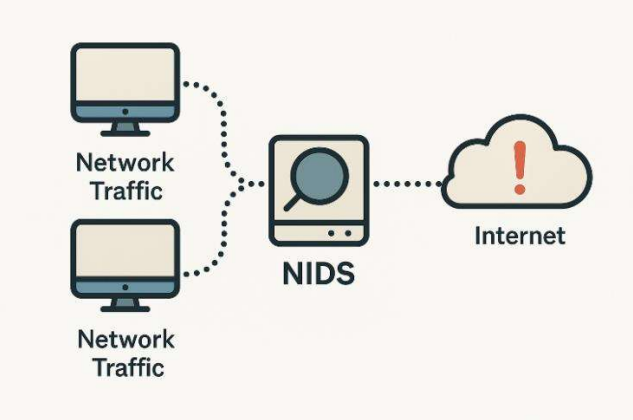
b. *Accountability*: sasaran keamanan yang berarti bahwa tindakan suatu entitas hanya dapat dilacak oleh entitas tersebut. Ini mendukung non-penyangkalan, pencegahan dan isolasi kesalahan, deteksi dan pencegahan intrusi, dan pemulihan setelah tindakan hukum. Karena sistem yang sepenuhnya aman belum menjadi tujuan yang dapat dicapai, kita harus memiliki kemampuan untuk mengidentifikasi pihak yang bertanggung jawab atas pelanggaran keamanan. Sehingga analisis forensik dapat melacak keamanan di masa mendatang, sistem harus menyimpan catatan aktivitasnya.

2.2 *Intrusion Detection System*

Intrusion detection system adalah sebuah mekanisme keamanan jaringan yang dirancang untuk mengidentifikasi dan memantau aktivitas tidak biasa, mencurigakan, atau berpotensi berbahaya dalam sistem jaringan komputer. Sistem ini bekerja dengan cara menganalisis lalu lintas data dan memberikan peringatan secara otomatis kepada administrator jaringan jika terdeteksi adanya ancaman atau penyimpangan dari pola yang dianggap normal (Abdulganiyu et al., 2023).

2.2.1 *Network Intrusion Detection System*

Network-Based Intrusion Detection System (*NIDS*) diimplementasikan pada keseluruhan jaringan komputer dengan tujuan untuk memantau, merekam, dan menganalisis lalu lintas data. Melalui proses ini, *NIDS* berfungsi untuk mendeteksi berbagai aktivitas mencurigakan atau berpotensi membahayakan yang mungkin terjadi di dalam jaringan, sehingga dapat membantu administrator sistem dalam mengidentifikasi serangan sebelum menimbulkan dampak yang lebih luas (Ahmad et al., 2021).



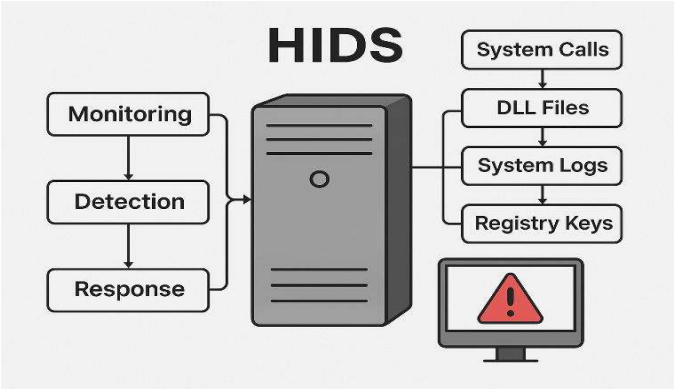
Gambar 2. 2 *Network Intrusion Detection System*

Pada gambar 2.2 *NIDS* tersebut menggambarkan alur kerja sistem deteksi intrusi berbasis jaringan di mana data dari beberapa komputer klien mengalir melalui jaringan dan diawasi oleh perangkat *NIDS* yang ditempatkan di titik strategis antara jaringan internal dan koneksi ke internet. Perangkat *NIDS* ini berfungsi untuk memonitor semua lalu lintas jaringan secara *real-time*, dengan tujuan mengidentifikasi adanya aktivitas yang mencurigakan atau pola serangan yang berpotensi membahayakan sistem. Dalam pengembangannya, pendekatan terbaru berbasis model Transformer yang memanfaatkan mekanisme *attention* terbukti mampu meningkatkan akurasi deteksi hingga lebih dari 93%, sehingga efektif dalam memperkuat keamanan jaringan, khususnya di lingkungan komputasi awan (Long et al., 2024). Setiap paket data yang melewati jalur ini dianalisis oleh *NIDS* untuk dibandingkan dengan tanda-tanda atau perilaku serangan yang telah dikenal sebelumnya. Bila terdeteksi aktivitas yang tidak sesuai dengan pola lalu lintas normal, *NIDS* akan menghasilkan peringatan yang dikirimkan ke administrator jaringan. Dengan begitu, sistem ini dapat membantu dalam mendeteksi serangan sejak dini sebelum mencapai tujuan utamanya, seperti eksploitasi sistem atau pencurian data.

2.2.2 *Host-based Intrusion Detection System*

Host-Based Intrusion Detection System (*HIDS*) bekerja dengan memeriksa *file log* dari lalu lintas *server* berdasarkan *database* yang ada di *server*, dan akan mengirimkan laporan kepada administrator sistem terkait ketika terdeteksi serangan yang tidak terduga. Sistem ini dipasang pada *server* khusus dan bertugas mendeteksi atau mencegah serangan dengan memantau *file* konfigurasi, mencatat

aktivitas sistem, memeriksa perubahan pada integritas sistem, serta mencegah penyalahgunaan. Kompatibilitas dengan sistem operasi menjadi hal penting, karena sifat sistem operasi yang berbeda dapat memengaruhi efektivitas *HIDS* (Efe & Abaci, 2022).



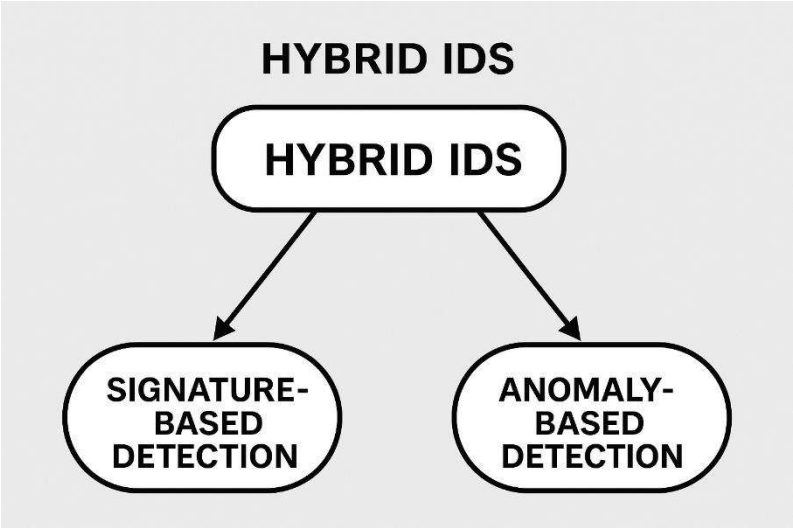
Gambar 2. 3 *Host-based Intrusion Detection System*

Pada gambar 2.3 menggambarkan alur kerja *Host-Based Intrusion Detection System* (*HIDS*) secara sederhana dan sistematis. Sistem ini berfokus pada pemantauan aktivitas yang terjadi langsung pada sebuah perangkat komputer atau *host*. Dimulai dari proses *monitoring*, *HIDS* mengawasi berbagai aktivitas yang berkaitan dengan sistem operasi, seperti panggilan sistem (*system calls*), *file DLL*, *registry keys*, dan *log* aktivitas lainnya yang ada di *host* tersebut (Sworna et al., 2023). Data yang terkumpul kemudian dianalisis dalam tahap *detection* menggunakan metode tertentu, yakni *signature-based detection* untuk mencocokkan aktivitas dengan pola serangan yang telah dikenal, dan *anomaly-based detection* yang menggunakan pendekatan statistik atau teknik *machine learning atau deep learning* untuk mengenali aktivitas yang menyimpang dari profil normal sistem. Pendekatan *anomaly-based* ini menjadikan *HIDS* lebih tangguh dalam menghadapi serangan baru atau *zero-day attacks* dibandingkan

signature-based HIDS (Satilmis et al., 2024). Jika ditemukan aktivitas mencurigakan atau serangan, sistem akan masuk ke tahap *response*, di mana *HIDS* memberikan peringatan kepada administrator sistem atau bahkan mengambil tindakan otomatis seperti memblokir proses yang berbahaya untuk mencegah kerusakan lebih lanjut (Timmons et al., 2025).

2.2.3 *Hybrid Intrusion Detection System*

Hybrid Intrusion Detection Sistem merupakan pendekatan canggih yang mengintegrasikan dua metode utama deteksi ancaman, yaitu pendekatan berbasis aturan (*signature-based*) dan pendekatan berbasis perilaku (*anomaly-based*). Dengan memadukan keunggulan kedua metode ini, sistem *Hybrid* mampu mengatasi keterbatasan yang dimiliki oleh sistem deteksi intrusi konvensional. Berbagai penelitian telah membuktikan bahwa pendekatan *Hybrid* memberikan performa yang lebih unggul dibandingkan sistem deteksi tunggal (Maseno et al., 2022).



Gambar 2. 4 *Hybrid Intrusion Detection System*

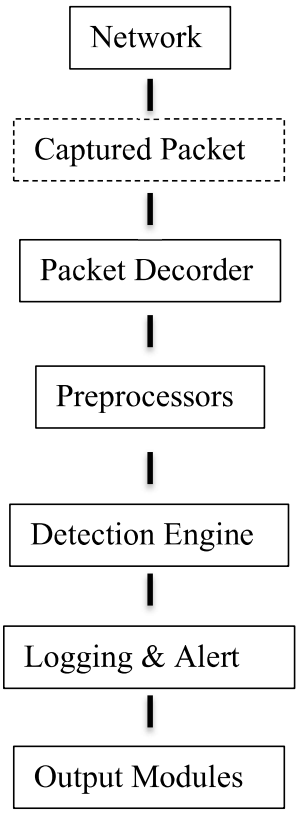
Pada gambar 2.4 menggambarkan *Hybrid IDS* yang memiliki dua pola deteksi yaitu *signaturebased detection* dan *anomly based detection*. Pada pendekatan *signature-based IDS*, sistem bergantung pada pola tertentu yang telah ditentukan sebelumnya untuk mendeteksi aktivitas berbahaya. *IDS* berbasis *signature* bergantung pada pola atau *signature* yang telah ditentukan sebelumnya untuk mengidentifikasi aktivitas berbahaya. Sistem ini membandingkan perilaku atau lalu lintas jaringan yang diamati dengan basis data *signature*, seperti urutan *byte* tertentu, *hash malware* yang telah dikenal, atau perintah mencurigakan. Deteksi berbasis *signature* sangat efektif untuk mengidentifikasi ancaman yang sudah terdokumentasi dengan baik dan menawarkan keunggulan berupa tingkat *false positive* yang rendah (Lorenzo et al., 2025). Di sisi lain, *anomaly-based detection* menggunakan pendekatan yang berbeda, di mana sistem mempelajari pola perilaku normal dari suatu lingkungan jaringan untuk kemudian mendeteksi adanya penyimpangan sebagai indikasi potensi ancaman. Deteksi anomali berbasis *normal-only* menggunakan model *machine learning* atau statistik yang dilatih secara eksklusif pada data yang merepresentasikan kondisi normal. Dalam banyak skenario dunia nyata, khususnya pada sistem siber-fisik (*CPS*), seringkali sulit untuk memperoleh dataset komprehensif yang mencakup seluruh jenis anomali, karena anomali cenderung jarang terjadi dan sangat beragam. Oleh karena itu, model mempelajari apa yang menjadi perilaku normal dan menandai setiap penyimpangan dari pola tersebut sebagai potensi *anomaly* (Acquaah & Kaushik, 2024).

2.3 *Snort*

Snort adalah perangkat lunak *open source* yang digunakan untuk mendeteksi dan mencegah intrusi pada jaringan komputer. *Snort* memiliki kemampuan untuk menganalisis lalu lintas jaringan secara *real-time* dan melakukan *logging* paket-paket data yang mencurigakan, serta mampu mendeteksi berbagai jenis serangan seperti *buffer overflow*, *stealth port scans*, *CGI attacks*, *SMB probes*, *OS fingerprinting attempts* (Barends et al., 2022).

2.3.1 *Snort* Arsitektur

Menurut peneliti (Shuai & Li, 2021) dalam *Snort* secara umum terdiri dari lima modul utama, yaitu modul penangkap paket (*Snort DAQ*), modul parsing, modul praproses, mesin deteksi (*detection engine*), dan modul keluaran alarm. Modul praproses bertugas untuk memproses lebih lanjut struktur paket yang telah diuraikan, sehingga siap untuk dideteksi oleh mesin deteksi. Mesin deteksi pada *Snort* melakukan pendeteksian lalu lintas intrusi berdasarkan serangkaian aturan deteksi, dan akan memicu mekanisme alarm *Snort* jika ditemukan kecocokan dengan aturan tersebut. Berikut gambaran aristektur snort secara keseluruhan:



Gambar 2. 5 *Snort* arsitektur

Pada gambar 2.5 *Snort* arsitektur merupakan struktur deteksi bekerja secara bertahap dengan memproses paket data dari jaringan melalui serangkaian komponen utama. Proses dimulai saat paket ditangkap oleh sistem, lalu diteruskan ke komponen *packet decorder* yang bertugas membaca dan menguraikan struktur dasar dari paket tersebut. Setelah itu, data yang telah didekodekan dikirim ke *preprocessor*. Selanjutnya, paket yang telah diproses akan masuk ke *detection engine*, yaitu inti dari sistem *Snort*. Disinilah proses pencocokan terhadap aturan (*rules*) dilakukan. Jika sebuah paket memenuhi kondisi yang ditentukan dalam salah satu *rule*, maka *Snort* akan menandainya sebagai potensi ancaman. Setelah itu, informasi hasil deteksi akan diteruskan ke bagian *logging* dan *alerting*. Di sini, *Snort* menyimpan detail kejadian ke dalam *log* atau menghasilkan peringatan sesuai

konfigurasi. Terakhir, data dari hasil deteksi tersebut bisa dikirim ke berbagai *output module*, baik dalam bentuk *file*, *console*, dan sistem *monitoring* eksternal (S & P, 2023).

2.3.2 *Snort Rules*

Menurut peneliti (Prakash & Palanivelan, 2024) dengan buku berjudul *"Algorithms in Advanced Artificial Intelligence"*, *Snort* memiliki 8 *rule* sebagai berikut:

a. *#Rule 1: Rule for Classic SQL Injection attack*

Aturan ini membantu mendeteksi Injeksi *SQL* klasik dengan memasukkan kondisi yang selalu bernilai benar. *tcp* memberi peringatan apa pun -> *any any (msg: "Possible Classic SQL Injection Alert"; flow:to_server, established; content:"' OR 1=1"; http_uri; sid:1000001; rev:1;)*

b. *#Rule 2: Rule for Blind SQL Injection attack*

Aturan ini berfokus pada Peringatan Injeksi *SQL* Buta dengan memasukkan perintah tidur, yang menyebabkan keterlambatan respon server. peringatan *tcp* apa pun -> *any any (msg: "Possible Blind SQL Injection Alert"; flow:to_server, established; content:"' AND SLEEP (5) --"; http_uri; sid:1000002; rey:1;)*

c. # Rule 3: *Rule for Time based blind SQL Injection attack*

Aturan ini mendeteksi Injeksi *SQL* buta berbasis waktu dengan memasukkan kondisi yang menyebabkan penundaan jika benar. peringatan *tcp* apa pun -> *any any (msg:"Possible Time-Based Blind SQLInjection Alert"; flow:to server, established; content:"'AND IF (1=1, SLEEP (5),0) --"; http_uri; sid:1000003; rev:1;)*

d. *#Rule 4: Rule for SQL Injection attack* (*Based on error*)

Aturan ini mendeteksi Peringatan Injeksi *SQL* berbasis Kesalahan dengan memasukkan kode yang memicu kesalahan pada *database*. beri tahu *tcp* apa pun -> *any any (msg: "Possible Error type SQL Injection Alert"; flow:to_server, established; content:"'AND 1=CONVERT (int, @@version) -"; http_uri; sid:1000004; rev:1;)*

e. *#Rule 5: Rule for Union SQL Injection attack*

Aturan ini mendeteksi Peringatan Injeksi *SQL* berbasis *Union* dengan memasukkan kata kunci *UNION*, memperingatkan untuk menggabungkan hasil dari kueri *database* yang berbeda. beri tahu *tcp* apa pun -> *any any (msg:"Possible Union SQL Injection Alert"; flow:to_server, established; content:"UNION"; http_uri; sid:1000005; rev:1;)*

f. *#Rule 6: Rule for Second order SQL Injection attack*

Aturan ini mendeteksi Peringatan Injeksi *SQL* Urutan Kedua, tempat muatan berbahaya disimpan untuk dieksekusi nanti. beri tahu *tcp* apa pun -> *any any (msg:"Possible SecondOrder SQL Injection Alert"; flow:to_server, established; content:"'OR '1'='1"; http_uri; sid:1000007; rev:1;)*

g. *#Rule 7: Rule for Out of band SQL Injection attack*

Aturan ini mengidentifikasi Peringatan Injeksi *SQL* Di Luar Pita dengan Peringatan untuk mengambil data melalui saluran alternatif. beri tahu *tcp* apa pun -> *any any (msg:"Possible Out-ofBand SQL Injection Alert"; flow:to_server, established; content:"UNION SELECT NULL, load_file ('/etc/ passwd'), NULL --"; http_uri; sid:1000008; rev:1;)*

h. #*Rule 8: Rule for Boolean SQL Injection attack*

Aturan ini mendeteksi Peringatan Injeksi *SQL boolean* dengan memasukkan kondisi yang selalu bernilai benar. beri tahu tcp apa pun -> *any any (msg:"Possible Boolean SQL Injection Alert"; flow:to_server, established; content:"'AND 1=1 --"; http_uri; sid:1000009; rev:1;).*

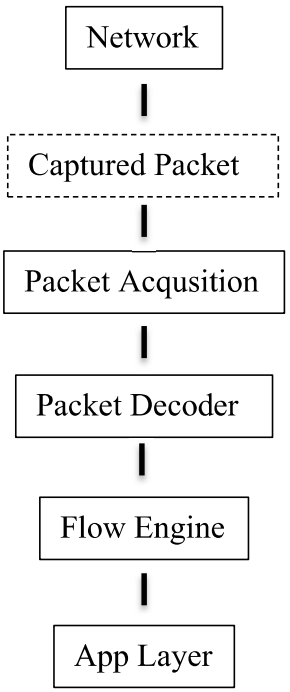
2.4 *Suricata*

Suricata merupakan sistem berbasis aturan yang menggunakan dari berbagai kumpulan aturan yang dikembangkan secara eksternal untuk memantau lalu lintas jaringan dan memperingatkan pada sistem administrator saat terjadi peristiwa yang mencurigakan tidak normal seperti paket yang dikirimkan secara terus menerus seperti indikasi serangan *Denial of Service Attack*. *Suricata* yang dapat mempercepat deteksi dan respon terhadap ancaman siber, sehingga memungkinkan administrator untuk lebih cepat merespon ancaman keamanan dan melindungi jaringan mereka dengan lebih efektif. Dalam tools pendeteksi pada *Suricata* yang sudah mengandung *Intrusion Detection System*, sehingga dapat menghentikan serangan dengan cara melakukan *drop packet* yang dicurigai. *Suricata* berbeda dari *Snort* namun *Suricata* memiliki dukungan untuk *rules* yang luas sehingga memiliki sintaks yang serupa dengan bahasa *rules Snort* (Hoover & Thompson, 2022).

2.4.1 *Suricata* Arsitektur

Dalam arsitektur *Snort*, fungsi *preprocessor* bertanggung jawab secara terpusat untuk pra-pemrosesan paket jaringan, sementara *Suricata* mengadopsi pendekatan modular dengan cara memisahkan proses ini menjadi dua bagian komponen utama: *decoding* dan *detection* (Dhuha Sabri Ghazi et al., 2024). Studi

terhadap kode sumber mengungkap bahwa modul *decoding* dalam *Suricata* berperan memperkaya representasi internal paket dengan *metadata* tambahan, sedangkan modul *detection* memanfaatkan data ini untuk menyediakan *keyword* yang diperlukan dalam pembuatan aturan (Boukebous et al., 2023). Desain terpisah ini tidak hanya meningkatkan fleksibilitas dalam menangani lalu lintas kompleks, tetapi juga mengoptimalkan kinerja pada lingkungan berkecepatan tinggi khususnya dalam skenario serangan terenkripsi atau *zero-day* (Asad et al., 2024). Berikut gambaran aristektur *Suricata* secara keseluruhan:



Gambar 2. 6 *Suricata* arsitektur

Pada gambar 2.6 Struktur *Suricata* bekerja dengan menangkap lalu lintas jaringan melalui modul *packet acquisition*, dimana sistem secara efisien mengumpulkan paket data dari antarmuka jaringan (Guo et al., 2022). Setelah paket ditangkap, proses dilanjutkan ke *packet decoder* untuk mengidentifikasi jenis

protokol dan struktur dasarnya, sebuah tahap yang kritis dalam memastikan interpretasi yang akurat terhadap data jaringan (Chandler & Wick, 2023). Informasi ini kemudian diteruskan ke *flow engine*, yang bertugas mengelola dan melacak aliran data untuk memahami sesi komunikasi antar *host*, termasuk pembangunan *flow table* untuk analisis *stateful* (Asad et al., 2024). Setelah aliran dikenali, *Suricata* melanjutkan ke level yang lebih dalam dengan melakukan inspeksi protokol aplikasi melalui *application layer*, memungkinkan deteksi serangan yang tersembunyi dalam protokol seperti *HTTP* atau *DNS* (Sudhanshu Sekhar & Bichitrananda, 2024). Selanjutnya *detection engine* menjalankan proses pencocokan terhadap *rule* yang telah didefinisikan, baik dalam format *Suricata* maupun *Snort style*, dengan dukungan mesin deteksi berbasis *signature* yang dapat diperluas (El Aeraj & Leghris, 2023). Jika paket sesuai dengan *rule* tertentu, hasilnya akan dikirim ke *output module* untuk menghasilkan *alert*, *log*, atau tindakan pencegahan sesuai konfigurasi seperti yang diterapkan dalam *IDS* berbasis *cloud* untuk respon otomatis (Rajesh, 2022).

Menurut peneliti (Mohanta & Saldanha, 2020) dari buku berjudul *"Malware Analysis and Detection Engineering: A Comprehensive Approach to Detect and Analyze Modern Malware, Suricata ",* mendukung 7 *action* yang dapat digunakan dalam aturannya dan artinya sebagai berikut:

Tabel 2. 1 *Action Suricata*

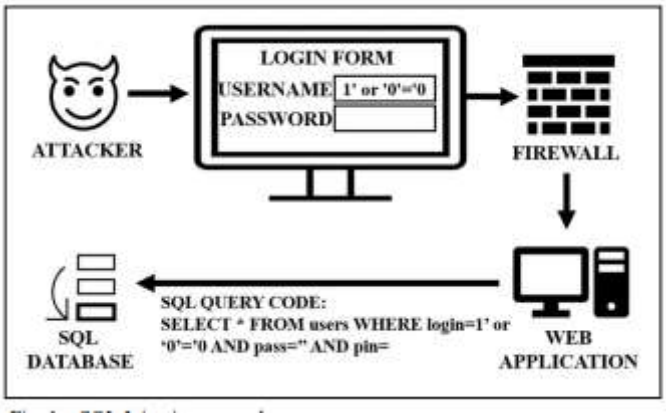
Action	deksripsi
alert	mencatat pemberitahuan untuk aturan jika cocok.
Pass	Jika aturan dengan tindakan ini cocok, *Suricata* tidak memperingatkan pada paket untuk apa pun aturan yang cocok di dalamnya sejauh ini dan juga melewatkan pencocokan aturan lain yang dimuat untuk paket itu.
Drop	Menjatuhkan paket. digunakan saat *Suricata* dijalankan sebagai *IpS*. juga mencatat pemberitahuan untuk Aturan
Reject	Semua tindakan penolakan adalah fitur *IpS*, di mana ketika aturan dengan tindakan ini cocok, *Suricata* mengirimkan penolakan aktif dari paket. Jika paket menyala yang aturan cocok adalah paket *tcp*, *Suricata* mengirimkan paket *tcp rst* ke pengirim paket yang cocok. Untuk semua jenis paket lainnya, itu mengirim paket kesalahan *ICMp*.
Rejectsrc	Sama seperti tindakan tolak.
Rejectdst	Bekerja sama dengan menolak, kecuali bahwa paket penolakan aktif dikirim ke tujuan paket tempat aturan cocok.
Rejectboth	Bekerja sama dengan menolak, kecuali bahwa paket penolakan aktif dikirim ke baik sumber maupun tujuan paket tempat aturan cocok.

2.5 *SQL Injection*

Menurut peneliti (Chen et al., 2021), *Structured Query Language* adalah bahasa pemrograman yang digunakan untuk *query database*. *SQL* didasarkan pada aljabar

relasional dan dirancang untuk mengelola data dalam Sistem Manajemen Basis Data Relasional (*RDBMS*). Beberapa perintah *SQL* yang paling penting adalah *CREATE*, *INSERT*, *UPDATE*, *DELETE*, *SELECT* dan *DROP*.

Dalam buku berjudul *"Malware Analysis and Detection Engineering: A Comprehensive Approach to Detect and Analyze Modern Malware SQL Injection"*. Mengacu pada kategori serangan injeksi di mana data yang disediakan oleh pengguna disertakan dalam kueri *SQL*. Tujuan dari kueri ini adalah untuk mengakui input pengguna sebagai kode *SQL*. Dengan memanfaatkan kerentanan ini, penyerang dapat mengirimkan perintah *SQL* langsung ke *database*. Serangan-serangan ini adalah beberapa ancaman paling serius terhadap aplikasi *Web* apa pun yang mengambil *input* data dari pengguna dan mengintegrasikannya ke dalam kueri *SQL*. Sebagian besar aplikasi *web* berfungsi seperti yang disebutkan di atas dan oleh karena itu rentan terhadap serangan injeksi *SQL*.



Gambar 2. 7 Contoh *SQL Injection*

Seperti terlihat pada Gambar 2.7 pengguna pada *form login* aplikasi *web* memasukkan kode "1' atau '0'='0". Serangan *SQL* ini didasarkan pada tautologi data dan pengenalannya menghasilkan kueri berikut:

"*SELECT * FROM users WHERE login=1' atau '0'='0 AND pass='' AND pin= "*

Dengan kueri di atas, kode dimasukkan dalam kondisi *OR* 0 = 0, mengubah seluruh istilah *WHERE* menjadi tautologi. Basis data menggunakan kondisi di atas sebagai dasar untuk mengevaluasi setiap baris dan memutuskan mana yang akan dikembalikan ke aplikasi. Karena kondisinya didasarkan pada tautologi, kueri dievaluasi sebagai benar untuk setiap baris dalam larik, sehingga menghasilkan pengembalian semua baris dalam tabel *database*.

2.5.1 Teknik *SQL Injection*

Pada peneliti (Gupta & Sharma, 2022) jurnal berjudul *"A Novel Approach for Detecting SQL Injection Attacks Using Snort"*. Serangan *SQL injection* (*SQLIA*) merupakan serangan paling umum pada basis data yang menyebabkan kerugian besar bagi organisasi. Beberapa teknik injeksi *SQL* yang paling umum sebagai berikut:

a. *AND/OR Attacks*

Untuk mengakses *database*, *programmer web* sering mengambil masukan dari formulir *web* dan sematkan nilai yang dimasukkan langsung ke kueri *SQL* untuk mendapatkan informasi yang diperlukan. Nilai *username_form* dan *password_form string* yang diperoleh dari pengguna melalui formulir *web* disematkan ke dalam kueri *SQL* untuk mengakses *database* berikut ini:

*SELECT * FROM tablename WHERE username = 'username_ form' AND password =' password_ form'*

Penyerang dapat mengubah kueri ini dan menambahkan " 'atau '1' = '1" ke kata sandi agar selalu benar dan memberikan akses ke *database* seperti yang ditunjukkan,

*SELECT * FROM tablename WHERE username = 'username_ form' AND password = 'xyz' OR '1' = '1';*

b. *Comments Attack*

Penyerang mengeksploitasi komentar sebaris dalam kueri *SQL* dengan dua cara; Untuk mengabaikan sebagian kueri dengan menetapkan nama pengguna yang valid diikuti dengan karakter komentar,

SELECT username, password FROM table_name WHERE username = 'xyz' –' AND password = 'none'

Di sini, segala sesuatu yang mengikuti tanda komentar ‘‘–’’ masuk klausa *WHERE* tidak akan diperiksa, dan penyerang akan berhasil masuk sebagai "xyz". Untuk mengaburkan *signature SQLIA* apa pun, mis. Tanda tangan "UNION ALL" bias diubah menjadi *'UNION /**/ALL'* atau *'UN/**/ION A/**/ LL'* dalam kueri.

c. *UNION Injection Attack*

UNION juga dimodifikasi oleh penyerang untuk mengakses rekaman dari lebih dari satu tabel dalam *database*. Misalnya, dalam kueri berikut penyerang menggunakan *UNION* untuk mengakses *data* dari tabel lain bernama Akun bersama dengan kueri asli.

SELECT username, password FROM table_name UNION ALL SELECT Account_Num, Name, balance FROM Accounts

d. *Stacked query*

Kueri bertumpuk dieksploitasi oleh penyerang dengan mengeksekusi beberapa pernyataan dalam satu panggilan ke *database*, dengan maksud untuk menghancurkan data pengguna dari *database* atau untuk mendapatkan hak istimewa admin seperti yang ditunjukkan,

*SELECT * FROM product_list WHERE category_id = 1; UPDATE members SET password = 'pswd' WHERE username = 'admin*

e. *Keywords*

Perintah kontrol administratif istimewa seperti Hapus, Jatuhkan, Versi, *UUID*, dll. pada *database* bertindak seperti *honeypot* untuk penyerang dan dieksploitasi untuk membuat perubahan yang tidak terkendali dalam *database*.

2.6 Tool Pendukung Uji Coba

2.6.1 *VirtualBox*

VirtualBox adalah perangkat lunak virtualisasi *native* yang memungkinkan pengguna untuk menjalankan beberapa sistem *guest OS* dalam satu komputer fisik (*host*). Solusi ini cocok untuk simulasi jaringan, seperti laboratorium pelatihan telepon *IP*, karena mendukung berbagai sistem operasi, menghubungkan hingga 36 adaptor jaringan *virtual*, dan bersifat *open-source*. Keunggulan *VirtualBox* dibandingkan solusi sejenis terletak pada kemudahan distribusi, fleksibilitas, serta kemampuannya menciptakan lingkungan *virtual* yang efisien untuk keperluan pendidikan atau pengembangan TI (Semerikov & Striuk, 2024).

2.6.2 *Ubuntu*

Ubuntu merupakan varian khusus dari sistem operasi *Ubuntu* yang dirancang khusus untuk keperluan *server*. Sebagai bagian dari keluarga *Linux*, sistem operasi ini menawarkan berbagai keunggulan dibandingkan sistem operasi *server* lainnya, termasuk kemudahan pengoperasian, stabilitas tinggi, serta dukungan komunitas pengembang yang luas yang turut berkontribusi dalam pengembangan platform berbasis *Linux* ini (Prathama et al., 2021).

2.6.3 *Kali Linux*

Kali Linux adalah sistem operasi berbasis *Linux* yang dikembangkan dari distribusi *Debian*, khusus dirancang untuk keperluan pengujian keamanan siber dan forensik digital. Sistem operasi ini dikembangkan oleh *Offensive Security* dan dilengkapi lebih dari 600 perangkat lunak keamanan bawaan seperti *Nmap*, *Metasploit*, dan *Wireshark* yang biasanya harus diinstal secara terpisah pada sistem operasi lain. *Kali Linux* mendukung berbagai arsitektur perangkat keras termasuk x86, x64, *ARM*, serta platform virtualisasi, dengan antarmuka yang fleksibel baik berbasis grafis maupun *command line*. Sebagai proyek *open-source*, *Kali Linux* memiliki komunitas pengembang yang aktif dan menggunakan model repositori aman untuk menjamin integritas alat-alat keamanannya. Karakteristik ini menjadikannya sebagai platform utama bagi profesional keamanan siber dalam melakukan pengujian penetrasi, sekaligus sarana pembelajaran bagi mereka yang ingin mempelajari teknik keamanan jaringan (walidin prana et al., 2025).

2.7 Peneliti terdahulu

Penelitian ini didasarkan pada penelitian sebelumnya dan analisis peneliti selama pengumpulan data. Penelitian sebelumnya juga terus menambah penelitian para peneliti. Berikut penelitian sebelumnya yang berkaitan dengan penelitian ini:

No	Author	Judul	Masalah	Metode	Teknik	Hasil
1.	Lukman, Melati Suci (Lukman & Suci, 2020)	Analisis Perbandingan Kinerja *Snort* Dan *Suricata* Sebagai *Intrusion Detection System* Dalam Mendeteksi Serangan *Syn Flood* Pada Web *Server Apache*	Bagaimana menganalisis dan membandingkan kinerja dua sistem deteksi intrusi (*IDS*), yaitu *Snort* dan *Suricata*, dalam mendeteksi serangan *SYN Flood* pada *web server Apache.*	Metode *SPDLC* (*Security Policy Development Life Cycle*).	-	*Suricata* berhasil mendeteksi serangan *SYN Flood* dengan baik, dan analisis dilakukan berdasarkan beberapa parameter, seperti jumlah serangan yang terdeteksi, efektivitas serangan, dan penggunaan sumber daya (*CPU* dan *RAM*) oleh masing-masing *IDS*.
2.	Dewi Yuliandari, Walim, Bangkit Kharisma Raja, Rahayu	Simulasi Penerapan Sistem *Monitoring* Jaringan *Snort NIDS* Pada *Web*	Bagaimana sistem keamanan jaringan terutama pada sebuah *server* yang selalu	Metode *SPDLC* (*Security Policy Development Life Cycle*).	-	Penerapan *Snort Base* pada jaringan dapat membantu administrator jaringan

No	Author	Judul	Masalah	Metode	Teknik	Hasil
	Ningsih, Ahmad Jurnaidi Wahidin (Yuliandari et al., 2023)	*Server* Menggunakan Metode *SPDLC*	berinteraksi dengan pengguna.			untuk meningkatkan keamanan jaringan serta membantu meningkatkan kewaspadaan terhadap ancaman siber.
3.	Faula Tanang Anugrah, Syariful Ikhwan, Jafaruddin Gusti A.G (Anugrah et al., n.d.)	Implementasi *Intrusion Prevention Sistem* (*IPS*) Menggunakan *Suricata* Untuk *Serangan SQL Injection*	Efektif aturan yang diterapkan dalam *Suricata* untuk mendeteksi serangan *SQL Injenction*, seperti *Blind SQL Injection* dan *Error SQL Injection*.	-	Metode eksperimen	*IPS* yang bekerja setiap terjadinya serangan *SQL Injection* akan dideteksi oleh *Suricata* dengan melakukan pengecekan kecocokan paket terhadap *signature rules*. *Rules* yang dinilai efektif untuk menghadapi serangan *SQL Injection* adalah

No	Author	Judul	Masalah	Metode	Teknik	Hasil
						rules yang mengguna kan beberapa kode asli sebagai kata kuncinya.
4.	Sahren, Ruri Ashari Dalimun the, Herman Saputra, Dian Yudha Kurnia Sirni (Dalimu nthe et al., 2023)	*IDPS Performan ce analysis for mitigating SQL Injections and SYN Flood attacks*	Tantangan dalam menilai efektivitas sistem *intrusion detection* dan *prevention system* (*IDPS*) dalam melindungi sistem informasi	-	Metode eksperi men	performa *IDPS* dapat membantu organisasi atau perusahaan dalam memilih dan mengimple mentasika n *IDPS* yang tepat sesuai
5.	Benny Wijaya, Arie Pratama (Wijaya & Pratama, 2020)	Deteksi Penyusupa n Pada *Server* Mengguna kan Metode *Intrusion Detection System* (*IDS*) Berbasis *Snort*	Permasalah an yang ada di warnet *cyber* terletak pada belum terdapatnya sistem keamanan server warnet, oleh karena itu beberapa kali server warnet *cyber* mengalami permasalaha	-	Metode ekperim en	Dengan mengguna kan Metode *Intrusion Detection System* (*IDS*) berbasis *Snort* administrat or jaringan bisa mengetahu i aliran paket-paket yang keluar

No	Author	Judul	Masalah	Metode	Teknik	Hasil
			n karena adanya penyeranga n yang dilakukan oleh pihak lain seperti *ping flood, smurf attack* dan lain - lain.			masuk sistem, *IDS* akan merekam semua aktifitas tersebut dan memberika n laporan paket-paket yang mencuriga kan sehingga administrat or jaringan dapat mengetahu i jika ada serangan yang masuk kedalam jaringan
6.	Akhil Krishna, Ashik Lal M A, Athul Joe Mathew kutty, Dhanya Sarah Jacob, Hari M (Krishna et al., 2020)	*Intrusion Detection and Preventio n System Using Deep Learning*	untuk menerapkan sistem deteksi dan pencegahan intrusi menggunak an *Deep Learning* yang dapat segera mendeteksi serangan seperti *DOS, Probe, R2L* dan *U2R* dan mencegahn	Metode *deep learning*	-	Sistem Deteksi Intrusi yang terpisah maupun sistem Pencegaha n Intrusi digabungk an sebagai satu sistem untuk mencapai tujuan tugas deteksi dan pencegaha

No	Author	Judul	Masalah	Metode	Teknik	Hasil
			ya. Intrusi yang muncul dideteksi menggunakan model *Deep Learning* yang disebut *Multi-Layer Perceptron* yang dilatih oleh kumpulan data *kddcup99* dengan akurasi tinggi.			n intrusi dengan cara yang lebih cepat dan efisien.
7.	Rachmawati Yulia Andarini, Purwono Hendradi, Setiya Nugroho (Yulia Andarini et al., 2023)	Meningkatkan keamanan terhadap *sql injection* studi kasus sistem kepegewaian BNN	Rentannya *website* simpan bank Magelang terhadap serangan *SQL Injection* dapat menimbulkan risiko kebocoran data pribadi dan berkas pelaporan bulanan pegawai.	-	Metode ekperimen	peretasan tidak dapat dilakukan, *SQL Injection* dapat juga dicegah dengan cara menerapkan keamanan website melalui penggunaan *captcha, OTP*, anti injeksi dibagian *sintax query database* pada

No	Author	Judul	Masalah	Metode	Teknik	Hasil
						source code dan menerapkan validasi tertentu seperti membatasi jumlah penginputan karakter serta mengubah jenis penginputan pada *form login*.
8.	Alka gupta, lalit sen sharma (Gupta & Sharma, 2020)	*Performance analysis and comparison of snort on various platforms*	Bagaimana performa *snort* dapat berbeda tergantung pada platform operasinya, yang dapat mempengaruhi kecepetan deteksi, akurasi, dan efesiensi sistem dalam menangani ancaman keamanan jaringan.	-	Metode eksperimen	Performa *snort* bervariasi tergantung pada platform yang digunakan. Pada beberapa kasus, *snort* menunjukkan performa lebih baik di linux dibandingkan dengan *windows.*
9.	Didit Hari Kuncoro Raharjo, Muhammad Salman	*Analyzing Suricata alert detection performance issue based on*	Penurunan akurasi deteksi *Suricata IDS* saat jumlah aturan aktif	-	Eksperimen	Hasil pengujian menunjukkan bahwa sistem deteksi intrusi

No	Author	Judul	Masalah	Metode	Teknik	Hasil
	(Raharjo & Muhammad Salman, 2023)	*active indicator of compromise rules*	(*IoC rules*) meningkat, terutama ketika diuji secara bersamaan. Studi sebelumnya fokus pada *packet drop*, bukan akurasi deteksi berbasis aturan.			(*IDS*) mampu mendeteksi serangan dengan akurasi 100% ketika jumlah aturan yang digunakan masih di bawah 10.000. Namun, terjadi penurunan akurasi yang signifikan ketika jumlah aturan melebihi 100.000. Pada skenario kelima, di mana seluruh aturan diaktifkan hingga mencapai total sekitar 1 juta aturan, sistem gagal mendeteksi serangan sama sekali,

No	Author	Judul	Masalah	Metode	Teknik	Hasil
						dengan tingkat deteksi 0%. Temuan ini mengindikasikan bahwa terlalu banyak aturan dapat membebani sistem dan menurunkan kinerjanya secara drastis. Meskipun demikian, proses tuning atau penyesuaian konfigurasi terbukti mampu meningkatkan performa deteksi secara keseluruhan.
10.	Rudi Hermawan (Hermawan, 2021)	Teknik uji penetrasi web server menggunakan *SQL Injection* dengan	Meningkatnya serangan siber pada keamanan website, khususnya melalui	-	Metode Eksperimen	Penyerang berhasil mengambil alih server target dengan mendapatk

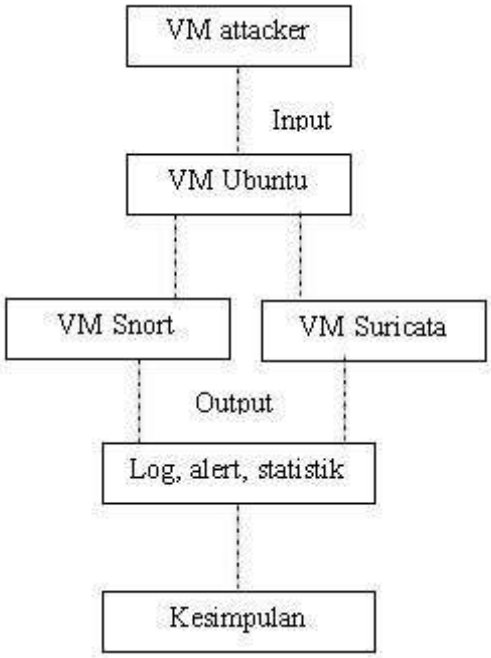
No	Author	Judul	Masalah	Metode	Teknik	Hasil
		SQLMap di *KaliLinux*	*teknik SQL injection*, yang memanfaatkan celah keamanan pada aplikasi *web* untuk mencuri atau memanipulasi data.			an informasi *user* dan *password* melalui *SQL injection*. Simulasi menunjukkan betapa rentannya sistem yang tidak menerapkan *filter input* yang memadai.

Tabel 2. 2 Penelitian Terdahulu

2.8 Kerangka pemikiran

Kerangka pemikiran berupa gambar alur logis yang menjelaskan hubungan antara masalah, tujuan dan pendekatan yang digunakan untuk menyelesaikan masalah dalam sebuah penelitian. Bisa dilihat pada gambar 2.8 kerangka pemikiran pada penelitian ini, proses penelitian ini dilatarbelakangi oleh meningkatnya ancaman serangan *SQL Injection* pada aplikasi *web* yang dapat menyebabkan pencurian atau kerusakan data. Salah satu metode mitigasi yang dapat digunakan adalah penerapan sistem deteksi intruksi (Intrusion Detection System / IDS). Dua *IDS* yang umum digunakan dalam lingkungan *open-source* adalah *Snort* dan *Suricata*. Untuk mengevaluasi performa keduanya, dilakukan penelitian eksperimental menggunakan lingkungan *virtual*. Pertama-tama, peneliti merancang *testbed virtual* yang terdiri dari dua mesin *virtual*: satu sebagai *attacker* (penyerang), satu sebagai *victim* (korban) yang menjalankan aplikasi *web* rentan

terhadap *SQL Injection* bersamaan dengan *IDS* (Snort atau Suricata). Setelah lingkungan siap, serangan *SQL Injection* disimulasikan menggunakan alat seperti *SQL Injection manual*. *IDS* akan memantau lalu lintas jaringan dan mendeteksi upaya serangan tersebut. Data berupa *log alert*, jumlah deteksi, dan performa sistem dikumpulkan dari masing-masing *IDS*. Data tersebut kemudian dianalisis untuk menilai tingkat akurasi deteksi. Dari hasil ini, dapat ditarik kesimpulan mengenai performa relatif antara *Snort* dan *Suricata* dalam mendeteksi serangan *SQL Injection*.



Gambar 2. 8 Kerangka Pemikiran