Analisis Keamanan Komunikasi Data Berbasis UDP pada Sistem BIPOL Tracker Menggunakan Metode *Defense in Depth* terhadap Serangan *False Data Injection, Replay Attack,* dan *UDP Flood*

SKRIPSI

ABDURRAHMAN AMMAR IHSAN
2207421047

PROGRAM STUDI TEKNIK MULTIMEDIA DAN JARINGAN JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER POLITEKNIK NEGERI JAKARTA
2026

Analisis Keamanan Komunikasi Data Berbasis UDP pada Sistem BIPOL Tracker Menggunakan Metode *Defense in Depth* terhadap Serangan *False Data Injection, Replay Attack,* dan *UDP Flood*

SKRIPSI

Dibuat untuk Melengkapi Syarat-Syarat yang Diperlukan untuk Memperoleh Diploma Empat Politeknik

ABDURRAHMAN AMMAR IHSAN
2207421047

PROGRAM STUDI TEKNIK MULTIMEDIA DAN JARINGAN JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER POLITEKNIK NEGERI JAKARTA

2026

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertanda tangan di bawah ini:

Nama	: Abdurrahman Ammar Ihsan
NIM	: 2207421047
Jurusan/Program Studi	: T.Informatika dan Komputer / Teknik Multimedia dan Jaringan
Judul skripsi	: Analisis Keamanan Komunikasi Data Berbasis UDP pada Sistem BIPOL Tracker Menggunakan Metode *Defense in Depth* terhadap Serangan *False Data Injection, Replay Attack,* dan *UDP Flood*

Menyatakan dengan sebenarnya bahwa skripsi ini benar-benar merupakan hasil karya saya sendiri, bebas dari peniruan terhadap karya dari orang lain. Kutipan pendapat dan tulisan orang lain ditunjuk sesuai dengan cara-cara penulisan karya ilmiah yang berlaku.

Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa dalam skripsi ini terkandung cirri-ciri plagiat dan bentuk-bentuk peniruan lain yang dianggap melanggar peraturan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Depok, 25 Juli 2026

Yang membuat pernyataan

Abdurrahman Ammar Ihsan

NIM 2207421047

LEMBAR PENGESAHAN

Skripsi diajukan oleh :

Nama	: Abdurruhman Ammar Ihsan
NIM	: 2207421047
Program Studi	: Teknik Multimedia dan Jaringan
Judul Skripsi	: ANALISIS KEAMANAN KOMUNIKASI DATA BERBASIS UDP PADA SISTEM BIPOL TRACKER MENGGUNAKAN METODE *DEFENSE IN DEPTH* TERHADAP SERANGAN *FALSE DATA INJECTION, REPLAY ATTACK,* DAN *UDP FLOOD*

Telah diuji oleh tim penguji dalam Sidang Skripsi pada hari Selasa, Tanggal 21, Bulan Juli, Tahun 2026 dan dinyatakan **LULUS.**

Disahkan oleh

Pembimbing I	: Dr. Prihatin Oktivasari, S.Si., M.Si.
Penguji I	: Dr. Indra Hermawan, S.Kom., M.Kom.
Penguji II	: Maria Agustin, S.Kom., M.Kom.
Penguji III	: Arlawan Andi Suhandana, S.Kom., M.T.I.

Mengetahui :
Jurusan Teknik Informatika dan Komputer
Ketua

Dr. Anita Hidayati, S.Kom., M.Kom
NIP 197908032003122003

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT atas rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "Analisis Keamanan Komunikasi Data Berbasis UDP pada Sistem BIPOL Tracker Menggunakan Metode Defense in Depth terhadap Serangan False Data Injection, Replay Attack, dan UDP Flood". Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Multimedia dan Jaringan.

Penulis ingin mengucapkan terima kasih dan penghargaan yang sebesar-besarnya kepada:

1. Kedua orang tua yang selalu memberikan doa, dukungan, dan semangat tanpa henti.
2. Dr. Prihatin Oktivasari, S.Si., M.Si selaku Pembimbing yang telah memberikan arahan, masukan, serta motivasi selama proses bimbingan.
3. Bapak/Ibu dosen di lingkungan Teknik Informatika dan Komputer yang telah memberikan ilmu dan pengetahuan selama masa studi.
4. Rekan perkuliahan saya, Cherry Puspa Scannia dan Shaquille Ariza Hidayat, yang telah membantu dalam pengembangan sistem ini.
5. Teman-teman TMJ 8B yang senantiasa memberikan semangat, bantuan, dan kebersamaan.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna, baik dari segi isi maupun penyajiannya. Oleh karena itu, penulis mengharapkan saran dan kritik yang membangun demi penyempurnaan karya ini di masa yang akan datang. Semoga skripsi ini dapat memberikan manfaat bagi pembaca dan menjadi kontribusi positif di bidang keamanan sistem IoT.

Depok, Juli 2026
Hormat Penulis,

Abdurrahman Ammar Ihsan

SURAT PERNYATAAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Politeknik Negeri Jakarta, saya bertanda tangan dibawah ini:

Nama	: Abdurrahman Ammar Ihsan
NIM	: 2207421047
Jurusan/Program Studi	: Teknik Informatika dan Komputer / Teknik Multimedia dan Jaringan

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Politeknik Negeri Jakarta Hak Bebas Royalti Non-Eksklusif atas karya ilmiah saya yang berjudul:

Analisis Keamanan Komunikasi Data Berbasis UDP pada Sistem BIPOL Tracker Menggunakan Metode *Defense in Depth* terhadap Serangan *False Data Injection*, *Replay Attack*, dan *UDP Flood*

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non Eksklusif ini Politeknik Negeri Jakarta Berhak menyimpan, mengalihmediakan/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan skripsi saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta. Demikian pernyataan ini saya buat dengan sebenarnya.

Depok, 25 Juli 2026

Yang menyatakan

Abdurrahman Ammar Ihsan

NIM. 2207421047

Analisis Keamanan Komunikasi Data Berbasis UDP pada Sistem BIPOL Tracker Menggunakan Metode *Defense in Depth* terhadap Serangan *False Data Injection, Replay Attack,* dan *UDP Flood*

ABSTRAK

Internet of Things (IoT) banyak memanfaatkan protokol User Datagram Protocol (UDP) karena memiliki latensi rendah dan overhead yang kecil sehingga sesuai untuk komunikasi data secara real-time. Namun, UDP tidak menyediakan mekanisme autentikasi, verifikasi integritas, maupun perlindungan terhadap replay secara bawaan sehingga rentan terhadap serangan False Data Injection, Replay Attack, dan UDP Flood. Penelitian ini bertujuan menganalisis keamanan komunikasi data berbasis UDP pada sistem BIPOL Tracker serta mengevaluasi efektivitas penerapan metode Defense in Depth dalam memitigasi ketiga serangan tersebut. Penelitian dilakukan menggunakan metode eksperimen dengan membandingkan kondisi sistem sebelum dan sesudah penerapan Defense in Depth yang terdiri atas rate limiting pada sisi server serta HMAC-SHA256 yang dipadukan dengan timestamp, boot identifier, sequence number, dan duplicate cache. Pengujian dilakukan pada arsitektur BIPOL Tracker yang menggunakan ESP32 dan SIM808 sebagai perangkat IoT serta Virtual Private Server (VPS) sebagai UDP Gateway. Hasil pengujian menunjukkan bahwa sebelum mekanisme keamanan diterapkan, server menerima paket hasil False Data Injection dan Replay Attack serta mengalami peningkatan beban saat terjadi UDP Flood. Setelah Defense in Depth diterapkan, paket yang tidak memiliki HMAC valid maupun paket replay berhasil ditolak, sedangkan mekanisme rate limiting mampu membatasi trafik UDP Flood sehingga komunikasi perangkat yang sah tetap berlangsung dengan baik. Berdasarkan hasil tersebut, dapat disimpulkan bahwa penerapan Defense in Depth mampu meningkatkan aspek autentikasi, integritas, freshness, dan ketersediaan komunikasi data pada sistem BIPOL Tracker sehingga risiko serangan terhadap komunikasi UDP dapat diminimalkan.

Kata kunci: Internet of Things, UDP, Defense in Depth, HMAC, Keamanan Komunikasi Data

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Internet of Things (IoT) telah banyak dimanfaatkan pada berbagai sektor, salah satunya sistem transportasi cerdas (*Intelligent Transportation System*/ITS), untuk mendukung layanan pelacakan kendaraan secara *real-time*. Pada sistem tersebut, perangkat IoT melakukan pertukaran data lokasi dan kondisi operasional kendaraan secara kontinu melalui jaringan komunikasi sehingga informasi dapat diterima pengguna dengan latensi yang rendah. Selain aspek performa, keamanan komunikasi data menjadi faktor penting karena setiap informasi yang dikirimkan melalui jaringan berpotensi menjadi sasaran berbagai ancaman siber apabila tidak dilindungi dengan mekanisme keamanan yang memadai (Khan et al., 2022).

Salah satu implementasi IoT pada bidang transportasi adalah BIPOL Tracker, yaitu sistem berbasis IoT yang dikembangkan untuk memantau posisi dan kondisi operasional bus kampus Politeknik Negeri Jakarta. Sistem ini menggunakan mikrokontroler ESP32 yang terintegrasi dengan modul komunikasi SIM808 untuk memperoleh data lokasi GPS serta mengirimkannya melalui jaringan GPRS menuju server yang berfungsi sebagai UDP *Gateway*, sebelum diteruskan ke aplikasi web maupun mobile secara real-time. Arsitektur tersebut memungkinkan informasi posisi bus dapat diakses secara cepat sehingga mendukung efektivitas layanan transportasi kampus (Tim Proyek BIPOL Tracker, 2025).

Pada implementasi awal, komunikasi data antara perangkat IoT dan server menggunakan *User Datagram Protocol* (UDP). Protokol UDP dipilih karena memiliki *overhead* yang rendah, latensi kecil, serta tidak memerlukan mekanisme pembentukan koneksi (*connectionless*), sehingga sesuai untuk kebutuhan komunikasi real-time pada perangkat IoT yang memiliki keterbatasan sumber daya. Namun demikian, UDP tidak menyediakan mekanisme keamanan bawaan seperti autentikasi, verifikasi integritas data, maupun perlindungan terhadap pengiriman ulang paket (*replay protection*). Oleh karena itu, keamanan

komunikasi bergantung pada mekanisme pengamanan yang diterapkan pada lapisan aplikasi (Khan et al,, 2022).

Ketiadaan mekanisme keamanan tersebut menyebabkan komunikasi UDP pada sistem BIPOL Tracker berpotensi mengalami serangan *false data injection*, *replay attack*, dan UDP *Flood*. *False data injection* dapat menyebabkan data lokasi yang diterima server menjadi tidak sesuai dengan kondisi sebenarnya. *Replay attack* memungkinkan paket yang sebelumnya sah dikirim kembali sehingga server menerima data lama seolah-olah merupakan data baru. Sementara itu, UDP *Flood* dapat membanjiri server dengan paket UDP dalam jumlah besar sehingga layanan menjadi lambat bahkan tidak dapat diakses. (Khan et al., 2022).

Salah satu pendekatan yang dapat diterapkan adalah *Defense in Depth*, yaitu penerapan beberapa mekanisme keamanan yang saling melengkapi. Pada penelitian ini, pendekatan tersebut diimplementasikan melalui *rate limiting* untuk mengurangi dampak serangan UDP Flood, serta *Hash-based Message Authentication Code* (HMAC) yang dikombinasikan dengan *timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server untuk memverifikasi keaslian perangkat, menjaga integritas data, serta mengurangi risiko *replay attack*.

Meskipun probabilitas *replay attack* relatif lebih rendah karena komunikasi antara perangkat IoT dan server dilakukan melalui jaringan seluler tanpa mengasumsikan adanya penyerang pada jalur komunikasi, *replay attack* tetap dijadikan skenario pengujian untuk mengevaluasi efektivitas mekanisme HMAC dan validasi *timestamp* yang dikombinasikan dengan *boot_id, sequence number*, serta *duplicate cache* pada sisi server.

Berdasarkan uraian tersebut, penelitian ini bertujuan untuk menganalisis keamanan komunikasi data berbasis UDP pada sistem BIPOL Tracker terhadap serangan False Data Injection, Replay Attack, dan UDP Flood, serta mengevaluasi efektivitas penerapan pendekatan Defense in Depth yang terdiri atas rate limiting dan HMAC yang dikombinasikan dengan timestamp, *boot_id*, *sequence number*, serta *duplicate cache* pada sisi server dalam meningkatkan autentisitas, integritas, *freshness*, dan ketersediaan komunikasi data pada sistem IoT.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian Analisis Keamanan Komunikasi Data Berbasis UDP pada Sistem BIPOL Tracker Menggunakan Metode *Defense in Depth* terhadap Serangan *False Data Injection*, *Replay Attack*, dan UDP *Flood* adalah sebagai berikut.:

1. Bagaimana potensi ancaman serta dampak serangan *False Data Injection*, *Replay Attack*, dan *UDP Flood* terhadap keamanan komunikasi data berbasis UDP pada sistem BIPOL Tracker?
2. Bagaimana penerapan metode *Defense in Depth* melalui mekanisme Rate Limiting, HMAC yang dikombinasikan dengan *timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server dalam memitigasi ancaman *False Data Injection, Replay Attack,* dan *UDP Flood* pada sistem BIPOL Tracker?

1.3 Batasan Masalah

Agar penelitian ini lebih terarah dan fokus sesuai dengan tujuan yang telah ditetapkan, maka batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Penelitian difokuskan pada analisis keamanan komunikasi data berbasis *User Datagram Protocol* (UDP) pada sistem BIPOL Tracker, khususnya komunikasi antara perangkat IoT (ESP32 dengan modul SIM808) dan VPS yang berperan sebagai UDP Gateway.
2. Analisis keamanan dibatasi pada tiga jenis ancaman, yaitu *False Data Injection*, *Replay Attack*, dan UDP *Flood*, yang dievaluasi terhadap aspek integritas (*integrity*), keaslian (*authenticity*), kebaruan data (*freshness*), dan ketersediaan layanan (*availability*).
3. Mekanisme pertahanan yang dianalisis menerapkan pendekatan *Defense in Depth*, yang terdiri atas *Rate Limiting* sebagai lapisan perlindungan terhadap UDP *Flood* serta *Hash-based Message Authentication Code* (HMAC) yang dikombinasikan dengan *timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server untuk menjaga autentikasi, integritas, dan mencegah *Replay Attack*. HMAC tidak digunakan sebagai mekanisme kerahasiaan data.

4. Pengujian dilakukan pada lingkungan yang terkontrol dan tidak dilakukan terhadap sistem BIPOL Tracker yang beroperasi pada lingkungan produksi.
5. Penelitian tidak membahas aspek keamanan selain komunikasi data UDP, seperti keamanan aplikasi web dan mobile, keamanan *database*, keamanan sistem operasi server, keamanan fisik perangkat, maupun serangan terhadap lapisan jaringan selain yang termasuk dalam ruang lingkup penelitian.
6. Penelitian tidak membahas mekanisme kerahasiaan data (*confidentiality*), seperti implementasi enkripsi payload, manajemen distribusi kunci, maupun optimasi performa kriptografi pada perangkat IoT.

1.4 Tujuan dan Manfaat Penelitian

Tujuan penelitian Analisis Keamanan Komunikasi Data pada Sistem BIPOL Tracker adalah sebagai berikut:

1. Menganalisis potensi ancaman dan dampak serangan *False Data Injection*, *Replay Attack*, dan *UDP Flood* terhadap keamanan komunikasi data berbasis *User Datagram Protocol* (UDP) pada sistem BIPOL Tracker.
2. Menganalisis penerapan metode Defense in Depth yang terdiri atas mekanisme *Rate Limiting*, serta autentikasi *Hash-based Message Authentication Code* (HMAC) yang dipadukan dengan *timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server dalam memitigasi serangan *False Data Injection*, *Replay Attack*, dan *UDP Flood* pada sistem BIPOL Tracker.

Adapun manfaat dari penelitian Analisis Keamanan Komunikasi Data pada Sistem BIPOL Tracker yaitu:

1. Menghasilkan informasi mengenai potensi ancaman, tingkat kerentanan, serta dampak serangan *False Data Injection, Replay Attack,* dan *UDP Flood* terhadap keamanan komunikasi data berbasis UDP pada sistem BIPOL Tracker sehingga dapat menjadi dasar dalam mengidentifikasi risiko keamanan pada sistem *Internet of Things* (IoT).

2. Memberikan hasil evaluasi mengenai kontribusi penerapan metode *Defense in Depth* yang terdiri atas mekanisme *Rate Limiting* serta *Hash-based Message Authentication Code* (HMAC) yang dipadukan dengan *timestamp, boot_id, sequence number,* dan *duplicate cache* pada sisi server dalam meningkatkan autentisitas, integritas, *freshness*, dan ketersediaan komunikasi data pada sistem BIPOL Tracker. HMAC berperan dalam menjamin autentikasi dan integritas pesan, sedangkan mekanisme *timestamp* mendukung pencegahan *replay attack*.
3. Menjadi referensi akademik bagi penelitian selanjutnya yang membahas analisis keamanan komunikasi data berbasis UDP dan penerapan mekanisme *Defense in Depth* pada sistem (IoT).

1.5 Sistematika Penulisan

Untuk memfokuskan pembahasan permasalahan serta memberikan gambaran yang jelas mengenai isi proposal skripsi secara menyeluruh, maka disusun sistematika penulisan sebagai pedoman dalam penyusunan skripsi. Adapun sistematika penulisan dalam penelitian ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi latar belakang masalah, perumusan masalah, batasan masalah, tujuan dan manfaat penelitian, serta sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini memuat kajian pustaka yang terdiri atas teori-teori yang relevan dan penelitian-penelitian terdahulu yang mendukung serta berkaitan dengan topik penelitian.

BAB III METODE PENELITIAN

Bab ini membahas rancangan penelitian, tahapan penelitian, objek penelitian, model atau teknik yang digunakan, teknik pengumpulan data serta teknik analisis data.

BAB IV HASIL DAN PEMBAHASAN

Bab ini menyajikan hasil penelitian yang meliputi analisis kebutuhan sistem,

implementasi mekanisme komunikasi dan keamanan, pengujian sistem, hasil pengujian, serta pembahasan dan analisis terhadap hasil yang diperoleh.

BAB V PENUTUP

Bab ini berisi kesimpulan yang diperoleh dari hasil penelitian serta saran yang dapat digunakan sebagai rekomendasi untuk pengembangan penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1 Studi Literatur

Tabel 2.1 Studi Literatur

No	Peneliti	Judul	Hasil	Perbedaan
1	Bostami, Ahmed, Choudhury (2019)	*False Data Injection Attacks in Internet of Things*	Penelitian mengkaji berbagai metode mitigasi FDIA pada IoT, seperti *en-route filtering*, MAC *authentication*, dan *machine learning* untuk menjaga integritas data.	Penelitian tersebut membahas mitigasi FDIA menggunakan *en-route filtering* dan MAC *authentication*, sedangkan penelitian ini menggunakan HMAC untuk menjaga autentikasi dan integritas komunikasi UDP sehingga mencegah *False Data Injection*.
2	Sara Lazzaro, Vincenzo De Angelis, Anna Maria Mandalari, Francesco Buccafurri (2024)	*Is Your Kettle Smarter Than a Hacker? A Scalable Tool for Assessing Replay Attack Vulnerabilities on Consumer IoT Devices*	Mengembangkan REPLIoT untuk mengidentifikasi kerentanan *Replay Attack* pada perangkat IoT. Hasil penelitian menunjukkan bahwa 75% perangkat IoT yang mendukung komunikasi lokal masih rentan terhadap *Replay Attack*.	Penelitian tersebut merekomendasikan mutual authentication dan protokol keamanan standar (TLS) sebagai mitigasi *Replay Attack*, sedangkan penelitian ini menggunakan *Defense in Depth* berupa HMAC yang dikombinasikan dengan *Timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server dan *Rate Limiting* untuk mengamankan komunikasi UDP terhadap *False Data Injection*, *Replay Attack*, dan *UDP Flood*.

No	Peneliti	Judul	Hasil	Perbedaan
3	Saeed Javanmardi, Meysam Ghahramani, Mohammad Shojafar, Mamoun Alazab, dan Antonio M. Caruso (2024)	*M-RL: A Mobility and Impersonation-Aware IDS for DDoS UDP Flooding Attacks in IoT-Fog Networks*	Penelitian ini mengusulkan IDS ringan berbasis *Rate Limiting* (RL) dan *Received Signal Strength* (RSS) untuk mendeteksi serangan DDoS *UDP Flooding* pada IoT-Fog. Hasil pengujian menunjukkan metode M-RL mampu mencapai akurasi deteksi lebih dari 99%.	Penelitian tersebut berfokus pada deteksi serangan *UDP Flooding* menggunakan IDS, sedangkan penelitian ini berfokus pada analisis keamanan komunikasi UDP menggunakan *Defense in Depth* berupa HMAC yang dikombinasikan dengan *Timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server dan *Rate Limiting* terhadap serangan *False Data Injection, Replay Attack,* dan *UDP Flood*.
4	El Hanjri, Sekkaki, Erroutbi (2019)	*Secure and Lightweight HMAC Mutual Authentication Protocol for Communication between IoT Devices and Fog Nodes*	Penelitian mengusulkan protokol *mutual authentication* berbasis HMAC dan *timestamp* yang mampu menjaga autentikasi, integritas, serta melindungi komunikasi IoT dari *Replay Attack*, *Man-in-the-Middle*, dan DoS dengan komputasi ringan.	Penelitian tersebut mengembangkan protokol *mutual authentication* pada lingkungan IoT-Fog, sedangkan penelitian ini menganalisis keamanan komunikasi UDP pada BIPOL Tracker menggunakan *Defense in Depth* berupa HMAC yang dikombinasikan dengan *Timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server dan *Rate Limiting* terhadap *False Data Injection*, *Replay Attack*, dan *UDP Flood*.

No	Peneliti	Judul	Hasil	Perbedaan
5	Mosteiro-Sanchez, Barcelo, Astroga, Urbieta (2020)	*Securing IIoT using Defence-in-Depth: Towards an End-to-End Secure Industry 4.0*	Mengusulkan penerapan *Defense in Depth* dengan OSCORE dan Attribute-Based Encryption (ABE) untuk meningkatkan keamanan end-to-end pada sistem IIoT.	Penelitian tersebut berfokus pada keamanan *end-to-end IIoT* menggunakan OSCORE dan ABE, sedangkan penelitian ini menganalisis keamanan komunikasi UDP pada BIPOL Tracker menggunakan HMAC, *Timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server dan *Rate Limiting* terhadap *False Data Injection, Replay Attack, dan UDP Flood.*

2.2 Keamanan Komunikasi Data pada IoT

Keamanan komunikasi data merupakan aspek penting dalam sistem *Internet of Things* (IoT) karena sebagian besar proses operasional bergantung pada pertukaran data antar perangkat melalui jaringan untuk mendukung fungsi pemantauan (*monitoring*) dan pengambilan keputusan. Dalam banyak implementasi, perangkat IoT menggunakan protokol *User Datagram Protocol* (UDP) karena memiliki *overhead* yang rendah dan latensi yang kecil sehingga sesuai untuk aplikasi *real-time*. Namun, UDP merupakan protokol *connectionless* yang tidak menyediakan mekanisme bawaan untuk autentikasi sumber, verifikasi integritas pesan, maupun perlindungan terhadap serangan *replay*, sehingga server tidak dapat memastikan bahwa paket yang diterima benar-benar berasal dari perangkat yang sah, tidak mengalami perubahan selama proses pengiriman, serta bukan merupakan paket yang dikirim ulang oleh pihak yang tidak berwenang. Kondisi tersebut menyebabkan komunikasi data rentan terhadap berbagai ancaman, seperti *False Data Injection* (FDI) yang memungkinkan penyerang mengirimkan data palsu dengan format menyerupai perangkat IoT yang sah sehingga mengganggu integritas informasi, *Replay Attack* yang memanfaatkan pengiriman ulang paket yang sebelumnya valid untuk memperoleh respons yang tidak semestinya, serta *Denial of Service* (DoS) melalui teknik *UDP Flood* yang

dapat mengganggu ketersediaan layanan akibat pengiriman paket UDP dalam jumlah besar. Oleh karena itu, perlindungan terhadap aspek autentikasi, integritas, *freshness*, dan ketersediaan komunikasi data menjadi kebutuhan penting untuk meningkatkan keandalan dan keamanan sistem IoT berbasis UDP.(Khan et al., 2022).

2.3 *User Datagram Protocol* (UDP) dalam Sistem IoT

User Datagram Protocol (UDP) merupakan protokol pada lapisan transport yang banyak digunakan dalam sistem *Internet of Things* (IoT) karena bersifat *connectionless*, memiliki struktur *header* yang sederhana, serta *overhead* yang rendah sehingga mampu mendukung komunikasi dengan latensi rendah dan penggunaan bandwidth yang lebih efisien. Karakteristik tersebut menjadikan UDP sesuai untuk perangkat IoT yang memiliki keterbatasan sumber daya komputasi, memori, dan konsumsi daya. Namun, berbeda dengan *Transmission Control Protocol* (TCP), UDP tidak menyediakan mekanisme bawaan untuk membangun koneksi, menjamin kehandalan pengiriman, autentikasi sumber, verifikasi integritas pesan, maupun perlindungan terhadap serangan *replay*. Akibatnya, server tidak dapat memverifikasi bahwa paket yang diterima benar-benar berasal dari perangkat yang sah, tidak mengalami perubahan selama proses transmisi, maupun bukan merupakan paket yang dikirim ulang oleh pihak yang tidak berwenang. Keterbatasan tersebut menyebabkan komunikasi berbasis UDP lebih rentan terhadap berbagai ancaman, seperti *False Data Injection*, *Replay Attack*, dan *UDP Flood*, sehingga diperlukan mekanisme keamanan tambahan untuk menjaga autentisitas, integritas, *freshness*, dan ketersediaan komunikasi data pada sistem IoT (Williams, 2022).

2.4 *Database*

Database merupakan kumpulan data yang terorganisasi sehingga dapat disimpan, dikelola, dan diakses secara efisien oleh aplikasi. Pada sistem *Internet of Things* (IoT), *database* berfungsi sebagai media penyimpanan data sensor yang dikirimkan oleh perangkat sehingga informasi dapat diolah, ditampilkan, dan dianalisis lebih lanjut (Silberschatz et al., 2019).

Pada sistem BIPOL Tracker, *database* digunakan untuk menyimpan data lokasi GPS, informasi kualitas udara, waktu pengiriman data, identitas perangkat, serta informasi lain yang diterima oleh server. Setelah paket UDP diterima dan divalidasi pada VPS sebagai UDP Gateway, data diteruskan ke *database Supabase* yang menggunakan *PostgreSQL* sebagai *database management system* (DBMS). Selanjutnya data tersebut dapat diakses oleh aplikasi web maupun aplikasi mobile untuk menampilkan informasi lokasi bus secara *real-time*.

2.5 Ancaman *False Data Injection* pada Komunikasi Data IoT

False Data Injection (FDI), merupakan bentuk serangan terhadap komunikasi data di mana entitas yang tidak sah mengirimkan paket data palsu ke sistem dengan meniru format atau identitas perangkat yang sah. Serangan ini tidak memerlukan penyadapan jalur komunikasi, melainkan murni memanfaatkan ketiadaan mekanisme autentikasi dan verifikasi integritas pada protokol yang digunakan. Pada sistem *Internet of Things* (IoT), khususnya yang menggunakan protokol ringan seperti *User Datagram Protocol* (UDP), server umumnya menerima dan memproses setiap paket yang sesuai dengan format yang diharapkan tanpa mampu memvalidasi keaslian sumber pengirim. Kondisi tersebut membuka peluang bagi penyerang untuk melakukan pemalsuan data (*packet forging*) atau penyisipan pesan (*message injection*) yang dapat memanipulasi informasi sistem, seperti perubahan koordinat lokasi atau nilai sensor. Dampak dari serangan FDI ini terutama berkaitan dengan pelanggaran integritas dan keaslian data, yang berpotensi menghasilkan informasi tidak valid dan memicu keputusan sistem yang keliru. Oleh karena itu, pada sistem IoT berbasis UDP tanpa mekanisme keamanan tambahan, *False Data Injection* ini menjadi salah satu ancaman utama terhadap keandalan komunikasi data. (A.A. Habib et al., 2023).

2.6 Ancaman *UDP Flood* pada Sistem IoT

UDP Flood merupakan salah satu teknik serangan *Denial of Service* (DoS) maupun *Distributed Denial of Service* (DDoS) yang bertujuan mengganggu

ketersediaan (*availability*) layanan dengan mengirimkan paket *User Datagram Protocol* (UDP) dalam jumlah besar secara terus-menerus ke server target. Serangan ini memanfaatkan karakteristik UDP yang bersifat *connectionless* sehingga server akan menerima dan memproses setiap paket UDP yang masuk tanpa mekanisme pembentukan koneksi terlebih dahulu. Akibatnya, kapasitas bandwidth, memori, maupun sumber daya pemrosesan server dapat terkuras sehingga layanan mengalami penurunan performa atau bahkan tidak dapat melayani permintaan dari perangkat yang sah. Pada sistem *Internet of Things* (IoT), ancaman *UDP Flood* menjadi semakin signifikan karena banyak perangkat menggunakan UDP untuk memenuhi kebutuhan komunikasi berlatensi rendah dan *overhead* yang kecil, sehingga serangan ini secara langsung mengancam aspek *availability* dalam *CIA Triad* (Kingsley et al., 2023).

2.7 *Replay Attack pada Sistem IoT*

Replay Attack merupakan salah satu bentuk serangan pada komunikasi *Internet of Things* (IoT) yang dilakukan dengan cara menangkap paket data yang sah, kemudian mengirimkan kembali paket tersebut pada waktu yang berbeda tanpa mengubah isi pesan. Karena paket yang diputar ulang berasal dari komunikasi yang valid, sistem yang tidak memiliki mekanisme untuk memverifikasi *freshness* data dapat menganggap paket tersebut sebagai data baru dan tetap memprosesnya. Akibatnya, sistem berpotensi menggunakan informasi yang sudah tidak relevan atau mengeksekusi kembali suatu proses berdasarkan data lama. Dalam konteks keamanan informasi, *Replay Attack* terutama mengancam aspek freshness karena sistem tidak mampu membedakan apakah paket benar-benar dikirim pada waktu saat ini atau hanya merupakan hasil pengulangan dari komunikasi sebelumnya. Oleh karena itu, komunikasi IoT memerlukan mekanisme yang mampu memverifikasi keaslian sekaligus kebaruan setiap paket data untuk mencegah pemrosesan ulang paket yang telah direkam oleh penyerang (Lazzaro et al., 2024).

2.8 *Defense in Depth* sebagai Pendekatan Keamanan

Defense in Depth merupakan pendekatan keamanan yang menerapkan beberapa lapisan pengamanan yang saling melengkapi untuk mengurangi risiko keberhasilan serangan terhadap suatu sistem. Pendekatan ini didasarkan pada prinsip bahwa satu mekanisme keamanan saja tidak cukup untuk menghadapi berbagai ancaman siber, sehingga apabila satu lapisan pertahanan berhasil ditembus, lapisan lainnya tetap dapat memberikan perlindungan terhadap sistem. Dalam lingkungan *Internet of Things* (IoT), penerapan *Defense in Depth* menjadi semakin penting karena keterbatasan sumber daya perangkat serta tingginya eksposur komunikasi jaringan. *Defense in Depth* mencakup berbagai mekanisme keamanan pada setiap lapisan sistem, mulai dari perangkat, jaringan, *middleware*, hingga aplikasi, sehingga mampu meningkatkan redundansi, ketahanan, serta mengurangi peluang keberhasilan serangan. Pada penelitian ini, konsep *Defense in Depth* diterapkan pada sisi komunikasi data berbasis UDP dengan mengkombinasikan beberapa mekanisme keamanan yang saling melengkapi, yaitu *Rate Limiting* untuk menjaga aspek ketersediaan (*availability*) dari serangan *UDP Flood*, serta HMAC yang dikombinasikan dengan *Timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server untuk menjamin autentikasi, integritas, dan *freshness* data sehingga dapat memitigasi serangan *False Data Injection* dan *Replay Attack*. Pendekatan berlapis tersebut diharapkan mampu meningkatkan keamanan komunikasi tanpa menghilangkan karakteristik ringan yang dibutuhkan oleh sistem IoT (Mosteiro-Sanchez et al., 2020).

2.9 HMAC dalam *Defense in Depth*

Hash-based Message Authentication Code (HMAC) merupakan mekanisme autentikasi pesan yang memanfaatkan fungsi hash kriptografi dan kunci rahasia (*secret key*) untuk memverifikasi keaslian (*authenticity*) dan integritas (*integrity*) data yang ditransmisikan melalui jaringan. HMAC bekerja dengan mengkombinasikan pesan dengan kunci rahasia yang hanya diketahui oleh pengirim dan penerima untuk menghasilkan nilai autentikasi (*message authentication code*). Pada sisi penerima, nilai HMAC dihitung kembali menggunakan kunci yang sama dan dibandingkan dengan nilai yang diterima.

Apabila kedua nilai tersebut berbeda, paket dianggap telah dimodifikasi atau berasal dari pihak yang tidak sah sehingga dapat ditolak. Oleh karena itu, HMAC mampu mencegah serangan *False Data Injection*, karena penyerang yang tidak memiliki kunci rahasia tidak dapat menghasilkan nilai HMAC yang valid. Perlu diperhatikan bahwa HMAC hanya menyediakan autentikasi dan integritas data, serta tidak memberikan kerahasiaan (*confidentiality*) terhadap isi pesan. Dalam penelitian ini, HMAC digunakan sebagai salah satu lapisan pada pendekatan *Defense in Depth* untuk memastikan bahwa hanya data yang berasal dari perangkat IoT yang sah dan tidak mengalami perubahan selama proses transmisi yang diterima oleh server (El Hanjri et al., 2019).

2.10 *Rate Limiting* dalam *Defense in Depth*

Rate Limiting merupakan mekanisme pengendalian lalu lintas jaringan yang membatasi jumlah paket atau permintaan yang dapat diproses dalam periode waktu tertentu untuk menjaga ketersediaan (*availability*) layanan. Dalam pendekatan *Defense in Depth*, *Rate Limiting* berfungsi sebagai salah satu lapisan pertahanan yang mengurangi dampak serangan *UDP Flood* dengan membatasi laju paket yang diterima server sehingga penggunaan sumber daya seperti CPU, memori, dan bandwidth tetap terkendali. Pada sistem berbasis Linux, mekanisme ini dapat diterapkan melalui iptables yang menyediakan kemampuan *packet filtering* dan *rate limiting* untuk menyaring lalu lintas jaringan berdasarkan aturan yang telah ditentukan. Meskipun demikian, *Rate Limiting* tidak melakukan autentikasi maupun verifikasi integritas data sehingga tidak mampu mencegah *False Data Injection* dan *Replay Attack*. Oleh karena itu, dalam pendekatan *Defense in Depth*, mekanisme ini perlu dikombinasikan dengan HMAC, *Timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server agar setiap lapisan keamanan saling melengkapi dalam menjaga aspek *availability, integrity, authenticity*, dan *freshness* pada komunikasi data IoT (Kingsley et al., 2023).

2.11 *Timestamp* dalam *Defense in Depth*

Timestamp merupakan mekanisme yang menyertakan informasi waktu pada setiap paket data untuk menunjukkan kapan paket tersebut dibuat atau dikirim. Dalam pendekatan *Defense in Depth*, *Timestamp* berfungsi menjaga *freshness* data dengan memungkinkan server memverifikasi apakah paket masih berada dalam batas waktu yang telah ditentukan, sehingga paket yang telah melewati batas waktu dapat ditolak. Meskipun demikian, *Timestamp* tidak dapat menjamin keaslian pengirim maupun integritas data apabila digunakan secara mandiri, serta belum sepenuhnya mampu mendeteksi paket yang dikirim ulang selama masih berada dalam rentang waktu yang diizinkan. Oleh karena itu, pada penelitian ini Timestamp dikombinasikan dengan *Hash-based Message Authentication Code* (HMAC) untuk memverifikasi autentikasi dan integritas paket, serta dilengkapi dengan *boot_id* sebagai identitas sesi perangkat, *sequence number* sebagai nomor urut paket dalam satu sesi, dan *duplicate cache* pada sisi server untuk menyimpan identitas paket yang telah diproses. Dengan mekanisme tersebut, server dapat memverifikasi HMAC, memvalidasi *Timestamp*, kemudian membandingkan kombinasi *boot_id* dan *sequence number* terhadap data pada *duplicate cache* sehingga paket yang telah diterima sebelumnya dapat diidentifikasi sebagai *Replay Attack* dan ditolak meskipun nilai HMAC masih valid. Dengan demikian, mekanisme *anti-replay* pada penelitian ini menerapkan beberapa lapisan keamanan yang saling melengkapi, yaitu HMAC untuk menjamin *authenticity* dan *integrity*, *Timestamp* untuk memastikan *freshness*, serta *boot_id, sequence number,* dan *duplicate cache* untuk mendeteksi serta mencegah pemrosesan ulang paket pada komunikasi UDP sistem BIPOL Tracker.

BAB III
METODE PENELITIAN

3.1 Rancangan Penelitian

Penelitian ini merupakan penelitian eksperimental dengan pendekatan kuantitatif yang bertujuan untuk menganalisis kerentanan komunikasi data berbasis *User Datagram Protocol* (UDP) pada sistem BIPOL Tracker serta mengevaluasi efektivitas penerapan metode *Defense in Depth* dalam meningkatkan keamanan komunikasi data. Pendekatan eksperimen dipilih karena penelitian ini melibatkan simulasi skenario serangan terhadap sistem pada lingkungan yang terkontrol, sehingga memungkinkan dilakukannya pengukuran dan perbandingan secara objektif antara kondisi sebelum dan sesudah penerapan mekanisme keamanan. Skenario serangan yang digunakan dalam penelitian ini meliputi *False Data Injection, Replay Attack,* dan *UDP Flood*.

Eksperimen dilakukan dengan memanipulasi variabel bebas berupa skenario serangan yang disimulasikan, kemudian mengamati dampaknya terhadap variabel terikat yang meliputi integritas (*integrity*), keaslian data (*authenticity*), data terbaru (*freshness*), dan ketersediaan layanan (*availability*). Pada skenario *False Data Injection*, dilakukan simulasi pengiriman paket UDP palsu yang menyerupai paket dari perangkat sah. Pada skenario *Replay Attack*, dilakukan pengiriman ulang paket yang sebelumnya telah direkam untuk menguji kemampuan sistem dalam mendeteksi paket yang tidak lagi valid. Sementara itu, pada skenario *UDP Flood*, dilakukan pengiriman paket UDP dalam jumlah besar untuk mengevaluasi dampaknya terhadap ketersediaan layanan. Parameter yang diamati meliputi *latency, packet loss, throughput*, jumlah paket yang diterima per satuan waktu, serta validitas data yang diterima server.

Hasil pengujian dari setiap skenario kemudian dianalisis secara komparatif dengan membandingkan kondisi sebelum dan sesudah penerapan metode *Defense in Depth*. Pendekatan tersebut terdiri atas *Rate Limiting* sebagai lapisan pertama untuk mengurangi dampak serangan *UDP Flood*, serta *Hash-based Message Authentication Code* (HMAC) yang dikombinasikan dengan *Timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server sebagai lapisan kedua

untuk memastikan autentisitas, integritas, dan data terbaru sekaligus mencegah *False Data Injection* dan *Replay Attack*. Hasil analisis digunakan untuk mengevaluasi tingkat kerentanan komunikasi berbasis UDP serta efektivitas penerapan Defense in Depth dalam meningkatkan keamanan sistem BIPOL Tracker.

3.2 Tahapan Penelitian

Penelitian ini dilaksanakan melalui beberapa tahapan yang disusun secara sistematis untuk mencapai tujuan penelitian. Tahapan penelitian dimulai dari identifikasi permasalahan hingga analisis hasil pengujian terhadap mekanisme keamanan yang diterapkan pada sistem BIPOL Tracker. Adapun tahapan penelitian meliputi sebagai berikut:

a) Studi Literatur

Tahap awal dilakukan dengan mengkaji berbagai referensi ilmiah yang berkaitan dengan keamanan komunikasi IoT berbasis UDP, karakteristik protokol UDP, konsep *Defense in Depth*, mekanisme autentikasi menggunakan HMAC, *timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server sebagai mekanisme pencegahan *replay attack, rate limiting*, serta penelitian terdahulu mengenai *False Data Injection, Replay Attack,* dan *UDP Flood.* Hasil studi literatur digunakan sebagai dasar dalam penyusunan model ancaman, rancangan pengujian, dan teknik analisis penelitian.

b) Analisis Sistem BIPOL Tracker

Tahap ini bertujuan untuk memahami arsitektur komunikasi pada sistem BIPOL Tracker, meliputi perangkat ESP32 yang terhubung dengan modul SIM808, jaringan GPRS, VPS sebagai *UDP Gateway*, *middleware*, serta aplikasi web dan mobile. Pada tahap ini dilakukan identifikasi aset yang dilindungi, karakteristik komunikasi UDP, serta potensi kerentanan keamanan yang dapat dimanfaatkan oleh penyerang.

c) Penyusunan Model Ancaman

Berdasarkan hasil analisis sistem, disusun model ancaman yang mendefinisikan batasan penelitian, kemampuan penyerang, aset yang

menjadi target, serta jenis serangan yang dianalisis. Model ancaman difokuskan pada tiga skenario serangan, yaitu *False Data Injection, Replay Attack,* dan *UDP Flood.*

d) Perancangan *Defense in Depth*

Pada tahap ini dirancang mekanisme keamanan berlapis menggunakan pendekatan *Defense in Depth*. Lapisan pertama berupa penerapan *rate limiting* pada server untuk mengurangi dampak *UDP Flood* terhadap ketersediaan layanan. Lapisan kedua berupa penerapan HMAC yang dikombinasikan dengan *timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server untuk menjaga autentikasi, integritas, serta data terbaru sekaligus mencegah *replay attack*.

e) Pelaksanaan Pengujian

Tahap pengujian dilakukan dalam lingkungan yang terkontrol dengan dua kondisi, yaitu sebelum dan sesudah penerapan *Defense in Depth*. Pengujian dilakukan menggunakan tiga skenario serangan, yaitu *False Data Injection, Replay Attack*, dan *UDP Flood*. Selama pengujian, seluruh aktivitas komunikasi dicatat melalui mekanisme logging pada server sebagai data penelitian.

f) Pengumpulan Data

Data penelitian dikumpulkan dari hasil pengujian melalui log komunikasi pada *UDP Gateway*. Data yang diperoleh meliputi status penerimaan paket, hasil verifikasi HMAC, validitas *timestamp*, jumlah paket yang diterima maupun ditolak, serta informasi lain yang diperlukan untuk mengevaluasi dampak masing-masing skenario serangan.

g) Analisis Hasil

Tahap terakhir dilakukan dengan membandingkan hasil pengujian sebelum dan sesudah penerapan *Defense in Depth*. Analisis difokuskan pada perubahan tingkat keamanan komunikasi berdasarkan aspek integritas (*integrity*), keaslian (*authenticity*), data terbaru (*freshness*), dan ketersediaan layanan (*availability*). Hasil analisis digunakan untuk mengevaluasi efektivitas setiap lapisan pertahanan dalam mengurangi risiko dari False Data Injection, Replay Attack, dan UDP Flood.

3.3 Objek dan Lingkup Penelitian

3.3.1 Objek Penelitian

Objek pada penelitian ini adalah sistem BIPOL Tracker yang menggunakan komunikasi data berbasis *User Datagram Protocol* (UDP) antara perangkat IoT yang terdiri atas ESP32 dan modul SIM808 dengan server *UDP Gateway* yang berjalan pada *Virtual Private Server* (VPS). Komunikasi tersebut digunakan untuk mengirimkan data operasional, seperti identitas kendaraan (*bus ID*), koordinat lokasi, kecepatan kendaraan, nilai sensor gas, dan *timestamp* dari perangkat menuju server sebelum diteruskan ke middleware dan aplikasi web maupun mobile.

Penelitian ini berfokus pada analisis keamanan komunikasi data berbasis UDP pada batas keamanan VPS sebagai *UDP Gateway*. Analisis dilakukan terhadap karakteristik komunikasi UDP yang secara bawaan tidak menyediakan mekanisme autentikasi, verifikasi integritas, maupun perlindungan terhadap *replay attack*, sehingga berpotensi dimanfaatkan oleh pihak yang tidak berwenang untuk melakukan manipulasi data maupun mengganggu ketersediaan layanan.

3.3.2 Lingkup Penelitian

Lingkup penelitian ini dibatasi pada analisis keamanan komunikasi data berbasis UDP pada sistem BIPOL Tracker dengan asumsi bahwa penyerang mengetahui alamat IP server, port UDP yang digunakan, serta format paket komunikasi, namun tidak memiliki akses fisik terhadap perangkat IoT maupun kunci rahasia yang digunakan dalam mekanisme autentikasi.

Analisis difokuskan pada tiga skenario serangan, yaitu *False Data Injection, Replay Attack, dan UDP Flood.* Ketiga skenario tersebut digunakan untuk mengevaluasi dampak ancaman terhadap aspek integritas (*integrity*), keaslian (*authenticity*), data terbaru (freshness), dan ketersediaan layanan (*availability*) pada komunikasi data berbasis UDP.

Sebagai mekanisme mitigasi, penelitian ini menerapkan pendekatan *Defense in Depth* yang terdiri atas dua lapisan pertahanan, yaitu *rate limiting* untuk mengurangi dampak *UDP Flood* serta HMAC

yang dikombinasikan dengan timestamp, *boot_id, sequence number,* serta *duplicate cache* pada sisi server untuk memberikan autentikasi, menjaga integritas data, dan mencegah *replay attack*. Penelitian ini tidak membahas aspek kerahasiaan (*confidentiality*), serangan pada lapisan fisik, *firmware*, penyadapan jaringan (*sniffing*), maupun keamanan aplikasi pada lapisan web dan mobile. Fokus penelitian hanya terbatas pada analisis keamanan komunikasi data antara perangkat IoT dan VPS sebagai *UDP Gateway.*

3.4 Teknik Pengumpulan Data

Teknik pengumpulan data dalam penelitian ini dilakukan melalui observasi dan pengujian eksperimental terhadap sistem BIPOL Tracker yang menggunakan komunikasi data berbasis *User Datagram Protocol* (UDP). Observasi dilakukan dengan memantau proses komunikasi antara perangkat IoT dan *UDP Gateway* pada VPS melalui mekanisme logging yang tersedia pada aplikasi server serta keluaran terminal *Visual Studio Code*. Data yang dikumpulkan meliputi informasi paket yang diterima, waktu penerimaan paket, hasil verifikasi HMAC, validitas *timestamp*, *boot_id, sequence number,* serta *duplicate cache* pada sisi server status penerimaan atau penolakan paket, serta jumlah paket yang diterima selama proses pengujian.

Selain observasi pada kondisi normal, pengumpulan data juga dilakukan melalui pengujian tiga skenario serangan, yaitu *False Data Injection, Replay Attack,* dan *UDP Flood*. Setiap skenario dijalankan pada kondisi sebelum dan sesudah penerapan mekanisme *Defense in Depth* yang terdiri atas *rate limiting* serta HMAC yang dikombinasikan dengan *timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server. Seluruh hasil pengujian didokumentasikan melalui logging pada server dan digunakan sebagai dasar untuk menganalisis efektivitas mekanisme pertahanan terhadap aspek integritas (*integrity*), keaslian (*authenticity*), data terbaru (*freshness*), dan ketersediaan layanan (*availability*).

3.5 Teknik Analisis Data

Teknik analisis data pada penelitian ini menggunakan analisis komparatif dengan membandingkan hasil pengujian pada kondisi sebelum dan sesudah

penerapan mekanisme *Defense in Depth*. Analisis dilakukan terhadap empat kondisi pengujian, yaitu kondisi normal (*baseline*), *False Data Injection, Replay Attack,* dan *UDP Flood*. Data yang diperoleh dari mekanisme logging pada *UDP Gateway*, keluaran terminal *Visual Studio Code*, serta hasil simulasi serangan dianalisis berdasarkan parameter yang telah ditentukan, meliputi *latency, packet loss, throughput*, jumlah paket yang diterima maupun ditolak, hasil verifikasi HMAC, serta validitas *timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server.

Analisis dilakukan dengan mengevaluasi dampak setiap skenario serangan terhadap aspek keamanan komunikasi yang meliputi integritas (*integrity*), keaslian (*authenticity*), data terbaru (*freshness*), dan ketersediaan layanan (*availability*). Pada skenario *False Data Injection*, analisis difokuskan pada kemampuan sistem dalam mendeteksi dan menolak paket yang tidak memiliki autentikasi yang valid. Pada skenario *Replay Attack*, analisis dilakukan terhadap kemampuan mekanisme *timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server dalam mengidentifikasi dan menolak paket yang dikirim ulang. Sementara itu, pada skenario *UDP Flood*, analisis difokuskan pada perubahan performa server melalui pengukuran *latency, packet loss, throughput*, serta jumlah paket yang berhasil diproses setelah penerapan *rate limiting*.

Seluruh hasil pengukuran disajikan dalam bentuk tabel dan grafik untuk memudahkan perbandingan antara kondisi sebelum dan sesudah penerapan *Defense in Depth*. Selanjutnya, hasil pengujian dianalisis secara deskriptif untuk mengevaluasi efektivitas setiap lapisan pertahanan dalam mengurangi dampak *False Data Injection, Replay Attack,* dan *UDP Flood* terhadap komunikasi data berbasis UDP pada sistem BIPOL Tracker.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk mengidentifikasi kebutuhan penelitian yang diperlukan dalam menganalisis keamanan komunikasi data berbasis User Datagram Protocol (UDP) pada sistem BIPOL Tracker. Kebutuhan tersebut meliputi perangkat keras, perangkat lunak, serta kebutuhan fungsional yang mendukung proses implementasi mekanisme Defense in Depth dan pelaksanaan pengujian terhadap serangan False Data Injection, Replay Attack, dan UDP Flood. Hasil analisis kebutuhan menjadi acuan dalam proses perancangan, implementasi, dan pengujian sistem pada penelitian ini. Tabel 4.1 menunjukkan kebutuhan fungsional penelitian yang digunakan sebagai dasar pelaksanaan implementasi dan pengujian.

Tabel 4.1 Analisis Kebutuhan Keamanan Sistem

No	Kebutuhan Fungsional
1	Sistem mampu menerima paket UDP dari perangkat ESP32 yang dikirim melalui jaringan GPRS.
2	Sistem mampu menerapkan mekanisme *Rate Limiting* pada *UDP Gateway* untuk membatasi laju paket yang diterima.
3	Sistem mampu memverifikasi autentikasi dan integritas paket menggunakan HMAC SHA-256.
4	Sistem mampu memvalidasi timestamp, *boot_id, sequence number,* serta *duplicate cache* pada sisi server untuk memastikan kesegaran data dan mencegah *Replay Attack*.
5	Sistem mampu meneruskan paket yang lolos proses validasi ke *middleware*.
6	Sistem mampu menolak paket yang tidak memenuhi proses autentikasi, integritas, atau validasi timestamp, *boot_id, sequence number,* serta *duplicate cache* pada sisi server.
7	Sistem mampu mencatat hasil pengujian sebelum dan sesudah penerapan *Defense in Depth* sebagai bahan analisis penelitian.

Selanjutnya, kebutuhan perangkat yang digunakan selama penelitian ditunjukkan pada Tabel 4.2 dan 4.3.

Tabel 4.2 Kebutuhan Perangkat Keras

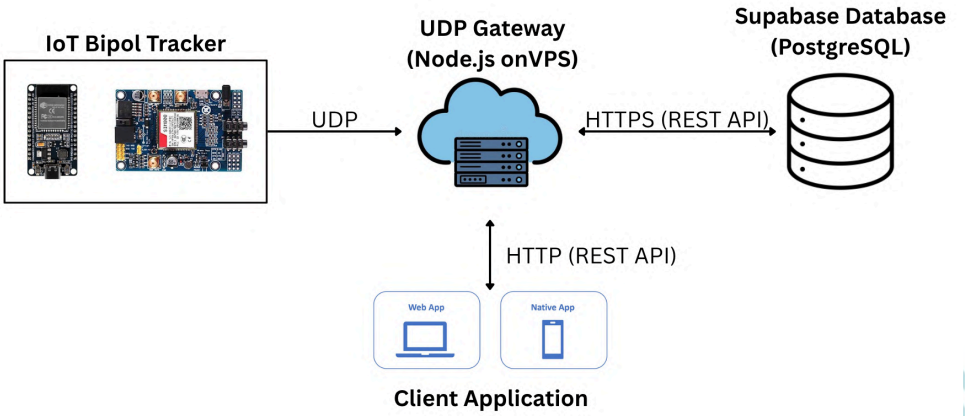
No	Perangkat	Fungsi
1	ESP32	Perangkat IoT pengirim data
2	SIM808	Modul GPS dan komunikasi GPRS
3	VPS Ubuntu	UDP Gateway dan implementasi *Defense in Depth*
4	Laptop	Implementasi, konfigurasi, dan analisis hasil pengujian
5	Jaringan Internet	Media komunikasi antara perangkat IoT dan VPS

Tabel 4.3 Kebutuhan Perangkat Lunak

No	Perangkat Lunak	Fungsi
1	Ubuntu Server	Sistem operasi VPS
2	Node.js	Implementasi *UDP Gateway*
3	Visual Studio Code	Membuat program
4	iptables	Implementasi *Rate Limiting*
5	VirtualBox	Lingkungan virtualisasi

4.2 Perancangan Sistem Keamanan

4.2.1 Arsitektur Sistem BIPOL Tracker



Gambar 4.1 Diagram Arsitektur Bipol Tracker

Gambar 4.1 menunjukkan arsitektur sistem BIPOL Tracker yang digunakan pada penelitian ini. Sistem terdiri atas empat komponen utama, yaitu perangkat IoT BIPOL Tracker, *UDP Gateway* yang berjalan pada *Virtual Private Machine* (VPS), database *Supabase PostgreSQL*, serta aplikasi klien yang terdiri atas aplikasi web dan aplikasi mobile. Arsitektur ini dirancang agar proses penerimaan, validasi, penyimpanan, dan penyajian data lokasi kendaraan dapat dilakukan secara terpusat.

Perangkat IoT BIPOL Tracker yang menggunakan mikrokontroler ESP32 dan modul SIM808 berfungsi memperoleh data lokasi *Global Positioning System* (GPS), kecepatan, dan gas kemudian mengirimkannya ke *UDP Gateway* menggunakan protokol *User Datagram Protocol* (UDP). Pemilihan UDP dilakukan karena memiliki *overhead* yang rendah sehingga sesuai untuk komunikasi data IoT yang membutuhkan pengiriman data secara periodik dengan latensi yang rendah.

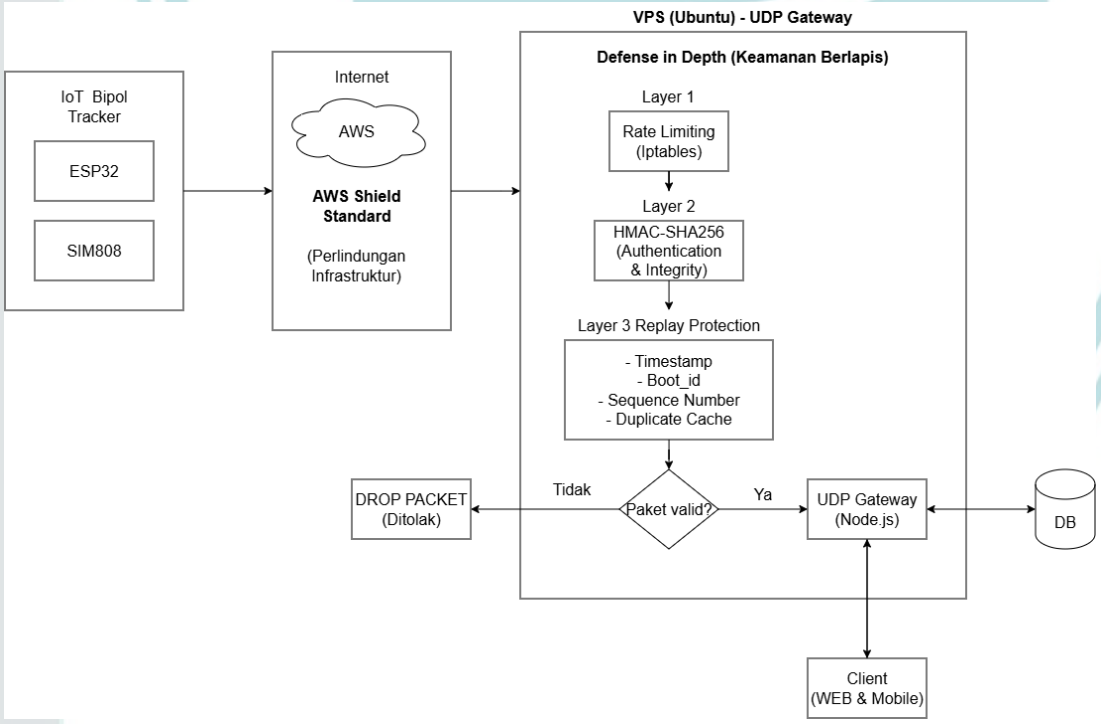
UDP Gateway yang diimplementasikan menggunakan *Node.js* pada VPS berfungsi sebagai batas keamanan (*security boundary*) sekaligus pusat pemrosesan komunikasi. *Gateway* menerima paket UDP dari perangkat IoT, melakukan proses validasi data sesuai mekanisme keamanan yang diterapkan, kemudian meneruskan data yang valid ke database *Supabase* melalui *REST API* menggunakan protokol HTTPS. Dengan demikian,

database tidak menerima koneksi secara langsung dari perangkat IoT, melainkan hanya melalui gateway.

Data yang telah tersimpan pada database kemudian diakses oleh aplikasi web dan aplikasi mobile melalui *REST API* yang disediakan oleh *UDP Gateway*. Mekanisme tersebut menjadikan seluruh komunikasi antara aplikasi klien dan server terpusat pada *gateway* sehingga proses autentikasi, validasi, serta pengelolaan akses data dapat dilakukan pada satu titik pengendalian.

Arsitektur ini juga mendukung penerapan mekanisme keamanan berlapis (*Defense in Depth*), di mana komunikasi UDP antara perangkat IoT dan UDP *Gateway* menjadi fokus penerapan pengamanan menggunakan *Rate Limiting* serta HMAC yang dikombinasikan dengan *timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server. Pendekatan tersebut bertujuan untuk meningkatkan aspek *integrity, authenticity*, *freshness*, dan *availability* pada komunikasi data perangkat IoT sebelum data disimpan ke dalam *database*.

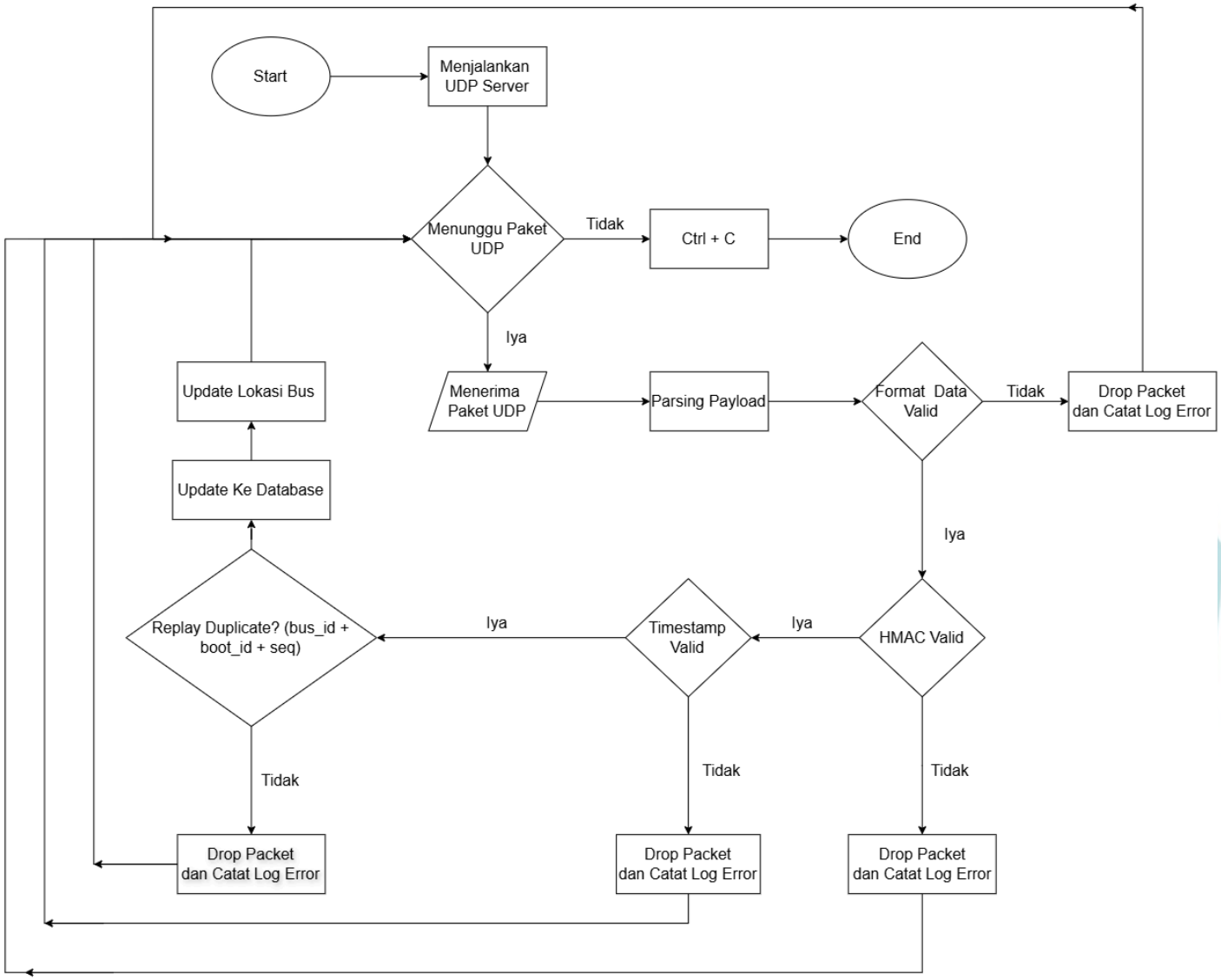
4.2.2 Perancangan Defense in Depth



Gambar 4.2 Diagram Blok Defense in Depth pada Sistem BIPOL Tracker

Gambar 4.2 menunjukkan rancangan mekanisme *Defense in Depth* yang diterapkan pada *UDP Gateway* sistem BIPOL Tracker. Paket data yang dikirimkan oleh perangkat IoT berbasis ESP32 dan SIM808 melalui jaringan seluler terlebih dahulu melewati infrastruktur AWS sebelum diterima oleh VPS yang berfungsi sebagai *UDP Gateway*. Pada sisi VPS diterapkan mekanisme keamanan berlapis yang terdiri atas tiga lapisan, yaitu *Rate Limiting* menggunakan *iptables* sebagai lapisan pertama untuk membatasi laju paket yang masuk sehingga dapat mengurangi dampak serangan *UDP Flood*, HMAC sebagai lapisan kedua untuk memverifikasi keaslian perangkat pengirim sekaligus menjaga integritas data, serta *Timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server sebagai lapisan ketiga untuk memastikan data terbaru (*freshness*) dan mencegah terjadinya *Replay Attack*. Setelah seluruh proses validasi selesai dilakukan, paket yang memenuhi seluruh persyaratan keamanan diteruskan ke *backend* untuk diproses dan disimpan ke *database*. Sebaliknya, paket yang tidak memenuhi salah satu mekanisme validasi akan ditolak (*drop packet*) sehingga tidak dapat memengaruhi proses pengolahan data pada sistem. Penerapan beberapa lapisan keamanan tersebut menunjukkan konsep *Defense in Depth*, yaitu setiap lapisan saling melengkapi untuk meningkatkan keamanan komunikasi data berbasis UDP terhadap ancaman *False Data Injection*, *Replay Attack*, dan *UDP Flood*. Konsep pertahanan berlapis ini sejalan dengan prinsip *Defense in Depth*, yaitu mengkombinasikan beberapa mekanisme keamanan independen sehingga kegagalan pada satu lapisan tidak secara langsung menyebabkan kegagalan keseluruhan sistem.

4.2.3 Perancangan Alur Validasi Paket



Gambar 4.3 *Flowchart* Validasi Paket UDP pada Server BIPOL Tracker

Gambar 4.3 menunjukkan alur validasi paket UDP yang diterapkan pada server BIPOL Tracker sebelum data diproses lebih lanjut. Proses dimulai ketika UDP Server dijalankan dan menunggu paket UDP dari perangkat IoT. Setelah paket diterima, server melakukan parsing payload untuk memisahkan setiap atribut data yang dikirimkan. Selanjutnya dilakukan validasi format data guna memastikan struktur payload sesuai dengan format yang telah ditentukan. Apabila format data tidak valid, paket akan ditolak dan sistem mencatat informasi kesalahan ke dalam log.

Apabila format data valid, server melanjutkan proses autentikasi menggunakan HMAC untuk memverifikasi keaslian perangkat pengirim sekaligus memastikan integritas data yang diterima. Paket yang gagal melewati proses autentikasi akan ditolak dan dicatat ke dalam log sebagai bagian dari mekanisme audit keamanan.

Setelah autentikasi berhasil, server melakukan validasi terhadap timestamp, *boot_id, sequence number,* serta *duplicate cache* pada sisi server untuk memastikan data masih berada dalam rentang waktu yang diizinkan sehingga mampu mencegah penerimaan kembali paket lama (*Replay Attack*). Paket yang memiliki *timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server tidak valid akan ditolak dan dicatat pada log kesalahan. Sebaliknya, apabila seluruh tahapan validasi berhasil dilewati, data dinyatakan valid sehingga server memperbarui data pada *database* dan informasi lokasi bus sebelum kembali ke kondisi menunggu paket UDP berikutnya. Alur ini menunjukkan penerapan mekanisme keamanan berlapis (*Defense in Depth*) melalui validasi format data, autentikasi HMAC, serta pemeriksaan *timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server sebelum data diproses lebih lanjut.

4.2.4 Perancangan Skenario Pengujian

Pengujian dilakukan dalam lingkungan terkontrol dengan mensimulasikan tiga jenis serangan, yaitu *False Data Injection, Replay Attack,* dan *UDP Flood*. Setiap skenario dilaksanakan pada dua kondisi, yaitu sebelum penerapan mekanisme *Defense in Depth* dan setelah penerapan mekanisme *Defense in Depth* yang terdiri atas *Rate Limiting,* HMAC yang dikombinasikan dengan *Timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server. Hasil pengujian digunakan untuk membandingkan tingkat keamanan sistem terhadap aspek integritas, autentikasi, *freshness*, dan ketersediaan layanan. Pendekatan ini sejalan dengan konsep *Defense in Depth* yang mengkombinasikan beberapa lapisan pengamanan agar mampu meningkatkan ketahanan sistem dibandingkan penggunaan satu mekanisme keamanan saja.

Tabel 4.4 Perancangan Skenario Pengujian

No	Jenis Pengujian	Tujuan	Kondisi Sebelum Defense in Depth	Kondisi Sesudah Defense in Depth	Parameter Evaluasi
1	*False Data Injection*	Mengetahui Kemampuan sistem mendeteksi paket palsu	Paket palsu dikirim tanpa autentikasi	Paket diverifikasi menggunakan HMAC	Integritas, Autentikasi
2	*Replay Attack*	Mengetahui kemampuan sistem menolak paket yang dikirim ulang	Paket lama diterima kembali	*Timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server digunakan untuk mendeteksi paket kedaluwarsa	*Freshness*, Autentikasi
3	*UDP Flood*	Mengetahui ketahanan server terhadap trafik UDP berlebihan	Server menerima seluruh paket	*Rate Limiting* membatasi jumlah paket per IP	*Availability*

a) Skenario Pengujian *False Data Injection*

Pada skenario ini dilakukan pengiriman paket UDP yang memiliki format sesuai dengan protokol komunikasi BIPOL Tracker, tetapi berisi data yang telah dimodifikasi sehingga tidak berasal dari perangkat IoT yang sah. Pengujian dilakukan untuk mengetahui apakah server dapat membedakan paket asli dan paket palsu sebelum dan sesudah penerapan HMAC. Indikator keberhasilan pengujian adalah kemampuan server menolak paket yang tidak memiliki nilai autentikasi yang valid sehingga integritas dan autentikasi data tetap terjaga. Konsep HMAC digunakan untuk menjamin autentikasi dan integritas pesan, bukan untuk menyediakan kerahasiaan data.

b) Skenario Pengujian *Replay Attack*

Pada skenario ini digunakan kembali paket UDP yang sebelumnya telah diterima server. Tujuan pengujian adalah mengevaluasi kemampuan sistem dalam mendeteksi paket lama yang dikirim ulang oleh penyerang. Sebelum penerapan *Defense in Depth*, server masih berpotensi menerima paket tersebut sebagai paket baru. Setelah penerapan Timestamp, *boot_id, sequence number,* serta *duplicate cache* pada sisi server yang diverifikasi bersama HMAC, paket yang berada di luar batas waktu valid akan ditolak sehingga aspek *freshness* dan autentikasi komunikasi dapat dipertahankan.

c) Skenario Pengujian *UDP Flood*

Pada skenario ini dilakukan pengiriman paket UDP dalam jumlah besar menuju *UDP Gateway* untuk mengukur kemampuan server dalam mempertahankan ketersediaan layanan. Sebelum penerapan *Defense in Depth* seluruh paket akan diproses oleh server sehingga berpotensi meningkatkan penggunaan sumber daya. Setelah penerapan *Rate Limiting* menggunakan *iptables*, jumlah paket yang diproses dibatasi sehingga dampak serangan terhadap ketersediaan layanan dapat dikurangi. Parameter yang diamati meliputi jumlah paket yang diterima, jumlah paket yang diblokir, penggunaan CPU, penggunaan memori, serta respons server selama pengujian. Fokus utama skenario ini adalah evaluasi terhadap aspek *Availability*.

4.3 Implementasi Sistem

Tahap implementasi sistem merupakan proses merealisasikan rancangan yang telah disusun pada sub bab sebelumnya menjadi sistem yang dapat digunakan dalam pengujian. Pada penelitian ini, implementasi dilakukan pada server *UDP Gateway* BIPOL Tracker dengan menerapkan mekanisme *Defense in Depth* yang terdiri atas *Rate Limiting*, HMAC, yang dikombinasikan dengan *Timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server. Selain itu, pada tahap ini juga dijelaskan implementasi lingkungan pengujian, alur pemrosesan paket UDP, serta mekanisme pencatatan (*logging*) yang digunakan sebagai dasar dalam proses evaluasi keamanan sistem.

4.3.1 Implementasi Lingkungan Sistem

Tabel 4.5 Spesifikasi Lingkungan Pengujian

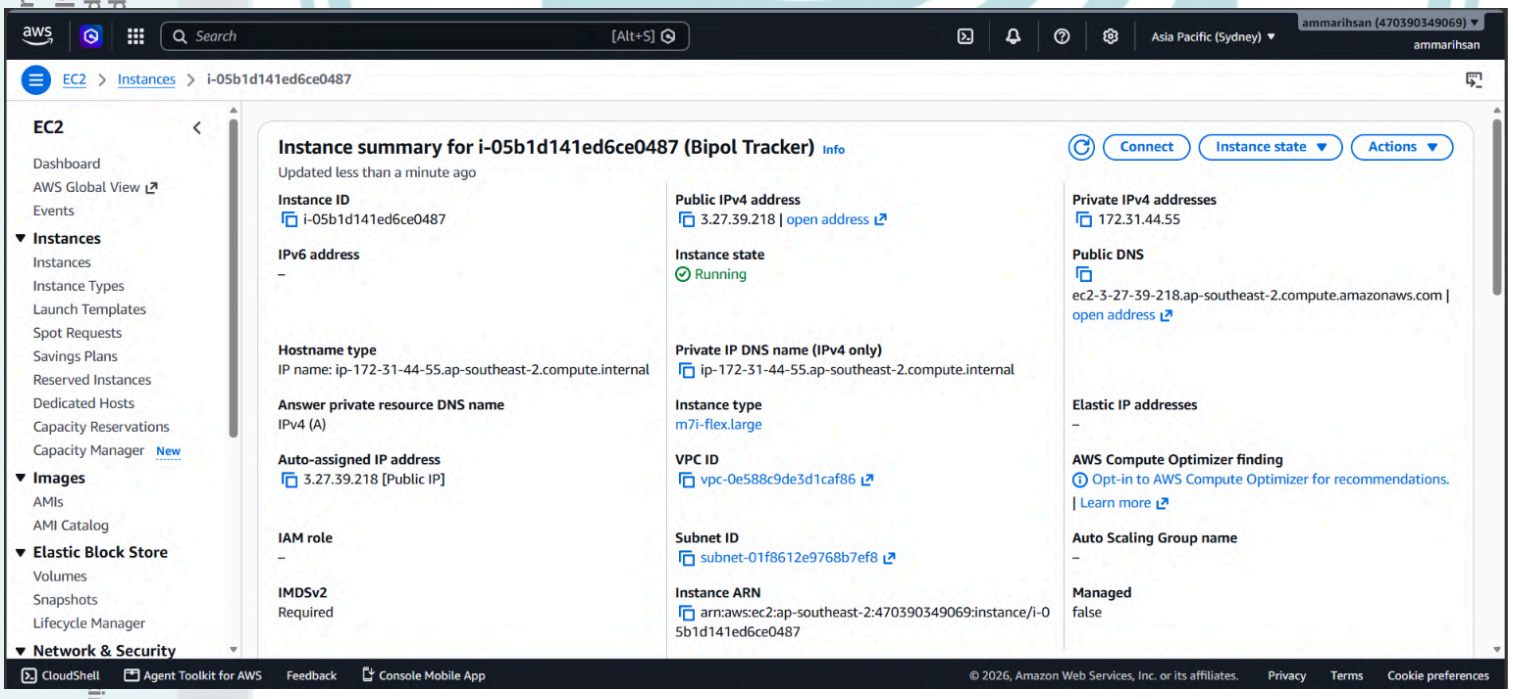
Komponen	Spesifikasi
Cloud Server	AWS VPS
Instance VPS	S m7i-flex.large
vCPU	2 vCPU
RAM Server	8 GB
Sistem Operasi	Ubuntu Server 22.04 LTS
Runtime Server	Node.js + Express
Port UDP Server	UDP 3333
Runtime Attacker	Node.js v22.18.0
Modul UDP	Dgram (Node.js built-in)
Database	Supabase
Real-time Dashboard	Socket.IO
Jaringan Pengujian	Internet Publik
Perangkat *Attacker*	Acer Nitro V15
Processor *Attacker*	Intel Core i5
RAM *Attacker*	16 GB
GPU *Attacker*	NVIDIA RTX 2050
Sistem Operasi *Attacker*	Windows 11

Implementasi lingkungan sistem ini dilakukan menggunakan layanan *cloud server* berbasis *Virtual Private Server* (VPS) pada platform AWS dengan spesifikasi instance *Standard_D2s_v3* dan *m7i-flex.large*. VPS digunakan sebagai server utama sistem BIPOL Tracker untuk menerima komunikasi UDP dari perangkat IoT berbasis ESP32 melalui jaringan GPRS/2G. Server menjalankan backend Node.js dan Express sebagai penerima paket UDP pada port 3333, kemudian memproses data GPS dan

sensor sebelum meneruskannya ke *dashboard real-time* menggunakan *Socket.IO* serta menyimpannya ke database Supabase. Penggunaan server cloud dipilih agar lingkungan pengujian lebih merepresentasikan implementasi sistem IoT nyata yang diakses melalui jaringan Internet publik.

Simulasi serangan dilakukan menggunakan perangkat terpisah yang berfungsi sebagai attacker machine dengan *runtime* Node.js v22.18.0 dan modul UDP bawaan Node.js (dgram). Pengujian difokuskan pada dua skenario ancaman, yaitu *unauthorized data injection* melalui pengiriman paket UDP palsu dan *Denial of Service* (DoS) melalui *flooding* paket UDP secara berulang ke server AWS. Seluruh komunikasi pengujian dilakukan melalui jaringan Internet publik sehingga pengujian dapat menggambarkan kondisi operasional nyata, termasuk pengaruh latency jaringan dan beban trafik terhadap ketersediaan layanan server.

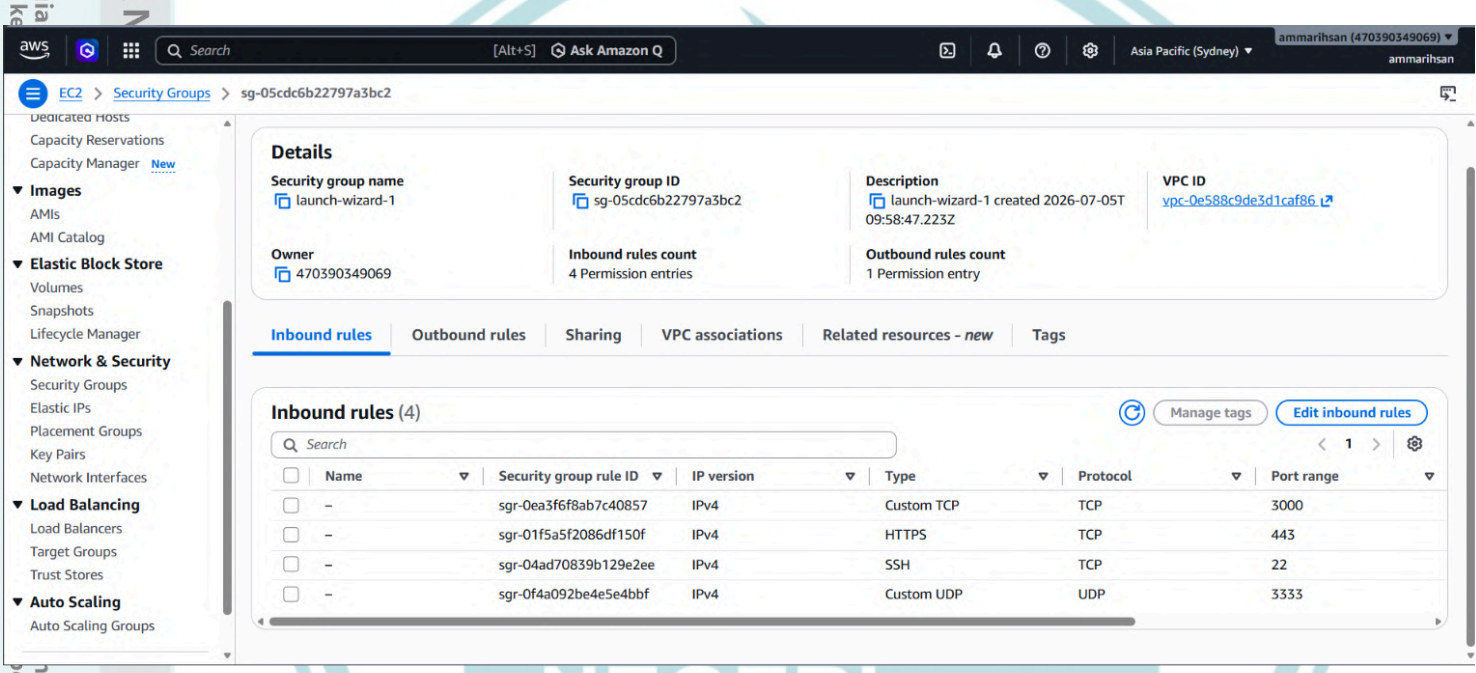
4.3.2 Implementasi Infrastruktur AWS



Gambar 4.4 *Instance Amazon EC2*

Gambar 4.4 menunjukkan instance Amazon EC2 yang digunakan sebagai server pada penelitian ini. Instance EC2 menjalankan sistem operasi

Ubuntu Server yang berfungsi sebagai UDP Gateway untuk menerima paket UDP dari perangkat ESP32 melalui jaringan GPRS. Server memiliki alamat Public IPv4 yang dapat diakses oleh perangkat IoT sehingga komunikasi data dapat berlangsung melalui jaringan internet. Setelah paket diterima oleh server, data diproses menggunakan aplikasi *UDP Gateway* berbasis Node.js sebelum dilakukan proses autentikasi HMAC, validasi *timestamp*, dan mekanisme *rate limiting*.



Gambar 4.5 *Security Group* AWS

Gambar 4.5 menunjukkan konfigurasi *Security Group* pada *instance Amazon EC2*. *Security Group* berfungsi sebagai *firewall virtual* yang mengatur lalu lintas jaringan menuju server. Pada penelitian ini, port UDP 3333 dibuka agar perangkat IoT dapat mengirimkan paket komunikasi ke UDP Gateway. Selain itu, port SSH (22) digunakan untuk administrasi server dari jarak jauh. Konfigurasi *Security Group* menjadi lapisan awal yang mengatur akses jaringan sebelum paket diproses lebih lanjut oleh *firewall Linux (iptables)* dan aplikasi *UDP Gateway*.

4.3.3 Implementasi *Rate Limiting*

Rate Limiting diimplementasikan pada UDP Gateway yang berjalan pada *Virtual Private Server* (VPS) menggunakan utilitas iptables pada sistem operasi Linux. Konfigurasi diterapkan pada port UDP 3333 dengan memanfaatkan modul hashlimit untuk membatasi jumlah paket UDP yang diterima dari setiap alamat IP sumber. Konfigurasi Rate Limiting yang digunakan pada penelitian ini ditunjukkan pada Gambar 4.6.

```
ubuntu@ip-172-31-44-55:~/hmac/bipol_v2$ sudo iptables -A INPUT -p udp --dport 3333 \
-m hashlimit \
--hashlimit-upto 10/sec \
--hashlimit-burst 20 \
--hashlimit-mode srcip \
--hashlimit-name perlindungan_bipol \
-j ACCEPT
ubuntu@ip-172-31-44-55:~/hmac/bipol_v2$ sudo iptables -A INPUT -p udp --dport 3333 -j DROP
ubuntu@ip-172-31-44-55:~/hmac/bipol_v2$ sudo iptables -L INPUT -v -n
Chain INPUT (policy ACCEPT 747 packets, 70677 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     17   --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:3333 limit: up to 10/sec burst 20 mode srcip
    0     0 DROP       17   --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:3333
ubuntu@ip-172-31-44-55:~/hmac/bipol_v2$
```

Gambar 4.6 *Konfigurasi Rate Limiting pada UDP Gateway*

Gambar 4.6 menunjukkan konfigurasi *Rate Limiting* yang diterapkan pada *UDP Gateway* menggunakan *iptables*. Aturan pertama membatasi penerimaan paket UDP pada port 3333 hingga maksimum 10 paket per detik dengan toleransi (*burst*) sebanyak 20 paket berdasarkan alamat IP sumber (*srcip*). Paket yang masih berada dalam batas tersebut akan diterima (*ACCEPT*), sedangkan paket yang melebihi batas akan ditolak (*DROP*). Hasil verifikasi menggunakan perintah *iptables* -L INPUT -v -n juga menunjukkan bahwa kedua aturan telah berhasil diterapkan pada *chain INPUT*.

Nilai 10 paket per detik pada mekanisme iptables hashlimit ditetapkan berdasarkan karakteristik komunikasi normal Bipol Tracker. Perangkat IoT mengirimkan paket dengan frekuensi yang jauh lebih rendah dari batas tersebut, sehingga seluruh trafik normal tetap dapat diterima tanpa mengalami pembatasan. Pemilihan nilai ini merupakan proses penyesuaian (*parameter tuning*) terhadap kebutuhan sistem, sesuai dengan prinsip bahwa konfigurasi *rate limiting* harus disesuaikan dengan

karakteristik layanan yang dilindungi, bukan menggunakan nilai baku yang berlaku untuk seluruh sistem.

4.3.4 Implementasi HMAC

Implementasi *Hash-based Message Authentication Code* (HMAC) dilakukan pada komunikasi data antara perangkat IoT BIPOL Tracker dan *UDP Gateway* sebagai mekanisme autentikasi serta verifikasi integritas data. Sebelum paket UDP dikirimkan oleh perangkat ESP32, sistem menghasilkan nilai HMAC menggunakan algoritma *SHA-256* berdasarkan isi pesan (*payload*) dan kunci rahasia (*secret key*) yang telah dibagikan sebelumnya antara perangkat dan server. Nilai HMAC tersebut kemudian disisipkan ke dalam paket UDP sebagai bagian dari data yang dikirimkan.

Pada sisi *UDP Gateway*, server melakukan perhitungan ulang nilai HMAC menggunakan payload yang diterima serta secret key yang sesuai dengan perangkat pengirim. Selanjutnya, hasil perhitungan tersebut dibandingkan dengan nilai HMAC yang terdapat pada paket UDP. Apabila kedua nilai identik, maka paket dianggap berasal dari perangkat yang sah serta tidak mengalami perubahan selama proses pengiriman. Sebaliknya, apabila nilai HMAC tidak sesuai, paket akan ditolak dan tidak diteruskan ke *middleware* maupun aplikasi.

Implementasi ini bertujuan untuk meningkatkan aspek autentikasi (*authentication*) dan integritas (*integrity*) pada komunikasi UDP yang secara bawaan tidak menyediakan mekanisme verifikasi identitas pengirim maupun pemeriksaan perubahan data. Dengan demikian, upaya penyisipan data palsu (*False Data Injection*) oleh pihak yang tidak memiliki *secret key* dapat diminimalkan karena server hanya menerima paket yang memiliki nilai HMAC yang valid. Peran HMAC sebagai mekanisme autentikasi dan integritas pada komunikasi IoT yang ringan juga banyak digunakan karena memiliki kebutuhan komputasi yang relatif rendah dibandingkan mekanisme kriptografi yang lebih kompleks.

```
function computeHmacHex(message, secret) {
  return crypto.createHmac('sha256', secret).update(message).digest('hex');
}
```

Gambar 4.7 Implementasi Pembentukan Nilai HMAC pada *UDP Gateway*

Gambar 4.7 menunjukkan implementasi pembentukan nilai HMAC menggunakan algoritma SHA-256. Fungsi computeHmacHex() menghasilkan nilai HMAC berdasarkan payload dan *secret key* yang telah ditentukan. Nilai HMAC tersebut kemudian digunakan sebagai kode autentikasi yang akan diverifikasi oleh UDP Gateway untuk memastikan bahwa data berasal dari perangkat yang sah dan tidak mengalami perubahan selama proses pengiriman.

```
function verifyPacketAuth(parsed, recvTs) {
  if (!parsed.signature) {
    return {
      signatureValid: null,
      invalidReason: REQUIRE_UDP_HMAC ? 'missing_signature' : null,
      accepted: !REQUIRE_UDP_HMAC
    };
  }

  const secret = getUdpSecretForBus(parsed.busId);
  if (!secret) {
    return {
      signatureValid: false,
      invalidReason: 'unknown_bus',
      accepted: false
    };
  }

  const expected = computeHmacHex(parsed.canonicalPayload, secret);
  const signatureValid = constantTimeHexEqual(expected, parsed.signature);

  return {
    signatureValid,
    invalidReason: signatureValid ? null : 'bad_hmac',
    accepted: signatureValid
  };
}
```

Gambar 4.8 Implementasi Verifikasi HMAC pada UDP Gateway

Gambar 4.8 menunjukkan proses verifikasi HMAC pada *UDP Gateway* setelah paket UDP diterima dari perangkat IoT. Server terlebih dahulu memeriksa keberadaan nilai HMAC (*signature*) pada paket yang diterima. Selanjutnya, server mengambil *secret key* berdasarkan identitas perangkat (*bus ID*) dan menghitung kembali nilai HMAC menggunakan fungsi computeHmacHex(). Nilai HMAC hasil perhitungan kemudian dibandingkan dengan nilai HMAC yang diterima menggunakan fungsi constantTimeHexEqual(). Apabila kedua nilai identik, paket dinyatakan valid dan diterima untuk diproses lebih lanjut. Sebaliknya, apabila nilai HMAC tidak sesuai, paket ditolak sehingga dapat mencegah penerimaan data palsu (*False Data Injection*) maupun perubahan data selama proses komunikasi.

Implementasi HMAC merupakan lapisan kedua pada pendekatan *Defense in Depth* yang diterapkan pada penelitian ini. Mekanisme tersebut melengkapi lapisan *Rate Limiting* yang berfokus pada perlindungan terhadap serangan *UDP Flood*, sedangkan validasi *Timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server yang dijelaskan pada sub bab berikutnya digunakan untuk memastikan *freshness* data sekaligus mencegah terjadinya *Replay Attack*. Dengan demikian, setiap mekanisme memiliki fungsi keamanan yang berbeda dan saling melengkapi dalam meningkatkan keamanan komunikasi data berbasis UDP pada sistem BIPOL Tracker.

4.3.5 Implementasi *Timestamp*

Implementasi timestamp dilakukan sebagai mekanisme untuk memberikan informasi waktu pengiriman paket (*freshness*) pada setiap data yang dikirimkan dari perangkat ESP32 menuju *UDP Gateway.* Penambahan *timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server bertujuan untuk memastikan bahwa paket yang diterima merupakan paket yang masih berada dalam rentang waktu yang diizinkan sehingga dapat mengurangi risiko *Replay Attack.*

Pada sisi perangkat, ESP32 membangkitkan nilai timestamp berdasarkan waktu yang diperoleh dari modul GPS SIM808. Dengan

demikian, timestamp, *boot_id, sequence number,* serta *duplicate cache* pada sisi server menjadi bagian dari data yang dilindungi oleh HMAC sehingga tidak dapat dimodifikasi tanpa menyebabkan proses verifikasi gagal. Pendekatan tersebut sesuai dengan prinsip bahwa HMAC menjamin autentikasi dan integritas pesan, sedangkan timestamp digunakan untuk memverifikasi kesegaran (*freshness*) paket.

```
uint64_t sentTsMs = gpsToUnixMillis(year, month, day, hour, minute, second);
```

Gambar 4.9 Implementasi *Timestamp* pada ESP32

Berdasarkan Gambar 4.9, perangkat ESP32 membangkitkan nilai *timestamp* menggunakan fungsi gpsToUnixMillis(). Fungsi tersebut mengonversi informasi waktu yang diperoleh dari modul GPS SIM808, meliputi tahun, bulan, tanggal, jam, menit, dan detik. Nilai timestamp selanjutnya dimasukkan ke dalam payload sebelum proses pembentukan HMAC sehingga waktu pengiriman menjadi bagian dari data yang akan diverifikasi oleh *UDP Gateway*.

Selanjutnya, *UDP Gateway* melakukan proses validasi terhadap *timestamp,* setelah paket diterima. Proses validasi dilakukan dengan memeriksa validitas format *timestamp* serta menghitung selisih waktu antara waktu penerimaan paket dengan waktu pengiriman yang tercantum pada *payload*.

```
if (!Number.isFinite(parsed.sentTs) && !parsed.isLegacyPayload) {
    return {
      signatureValid: false,
      invalidReason: 'invalid_timestamp',
      accepted: false
    };
  }

  if (!parsed.isLegacyPayload) {
    const ageMs = recvTs - parsed.sentTs;
    if (ageMs < -5000 || ageMs > UDP_HMAC_MAX_AGE_MS) {
      return {
        signatureValid: false,
        invalidReason: 'stale_timestamp',
        accepted: false
      };
    }
  }
```

Gambar 4.10 Implementasi Validasi Timestamp pada UDP Gateway

Berdasarkan Gambar 4.10, *UDP Gateway* terlebih dahulu memverifikasi bahwa nilai *timestamp* yang diterima memiliki format yang valid. Setelah itu, server menghitung selisih waktu antara waktu penerimaan paket (*recvTs*) dan waktu pengiriman (*parsed.sentTs*). Apabila selisih waktu berada di luar batas toleransi yang telah ditentukan oleh parameter UDP_HMAC_MAX_AGE_MS, paket akan dianggap telah kedaluwarsa (*stale timestamp*) sehingga ditolak dan tidak diproses lebih lanjut. Mekanisme ini memastikan bahwa hanya paket yang masih berada dalam rentang waktu yang diizinkan yang dapat diterima oleh sistem.

Implementasi timestamp, *boot_id, sequence number,* serta *duplicate cache* pada sisi server pada penelitian ini melengkapi mekanisme HMAC dalam penerapan *Defense in Depth*. HMAC digunakan untuk memverifikasi autentikasi dan integritas data, sedangkan *timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server digunakan untuk memastikan kesegaran (*freshness*) paket yang diterima. Kombinasi kedua mekanisme tersebut memungkinkan *UDP Gateway* mendeteksi paket lama yang dikirim

kembali oleh pihak yang tidak berwenang sehingga risiko *Replay Attack* dapat dikurangi.

4.4 Pengujian

Pengujian dilakukan untuk mengevaluasi karakteristik keamanan komunikasi data berbasis UDP pada sistem BIPOL Tracker sebelum dan sesudah penerapan mekanisme Defense in Depth. Pengujian difokuskan pada tiga skenario serangan, yaitu *False Data Injection*, *Replay Attack*, dan *UDP Flood*, untuk menganalisis dampaknya terhadap aspek keamanan yang meliputi *integrity*, *authenticity*, *freshness*, dan *availability*. Hasil pengujian kemudian digunakan sebagai dasar dalam mengevaluasi efektivitas mekanisme *Rate Limiting*, *HMAC* yang dikombinasikan dengan *Timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server yang diterapkan pada sistem.

4.4.1 Deskripsi Pengujian

Pengujian pada penelitian ini bertujuan untuk mengevaluasi karakteristik keamanan komunikasi data berbasis *User Datagram Protocol* (UDP) pada sistem BIPOL Tracker sebelum dan sesudah penerapan mekanisme *Defense in Depth*. Pengujian dilakukan untuk mengetahui dampak serangan terhadap komunikasi data serta mengevaluasi efektivitas mekanisme keamanan yang diterapkan dalam menjaga aspek *integrity*, *authenticity*, *freshness*, dan *availability*.

Pengujian dilakukan melalui lima skenario, yaitu kondisi normal (*baseline*), *False Data Injection, Replay Attack, UDP Flood*, serta pengujian ulang setelah penerapan mekanisme *Defense in Depth* yang terdiri atas *Rate Limiting*, HMAC yang dikombinasikan dengan *Timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server. Setiap skenario diamati berdasarkan parameter yang telah ditentukan untuk mengetahui perubahan kondisi sistem sebelum dan sesudah mekanisme keamanan diterapkan.

Parameter yang diamati selama pengujian meliputi jumlah paket yang diterima server, *latency*, *packet loss*, *throughput*, validitas data pada dashboard, kemampuan sistem dalam memverifikasi keaslian paket,

kemampuan mendeteksi paket *replay*, serta kemampuan mempertahankan ketersediaan layanan ketika menerima trafik UDP dalam jumlah besar. Data hasil pengujian selanjutnya dianalisis untuk mengevaluasi efektivitas penerapan *Defense in Depth* dalam meningkatkan keamanan komunikasi data pada sistem BIPOL Tracker.

4.4.2 Prosedur Pengujian

Pengujian dilakukan secara bertahap untuk memperoleh data mengenai karakteristik keamanan komunikasi data berbasis UDP pada sistem BIPOL Tracker sebelum dan sesudah penerapan mekanisme *Defense in Depth*. Setiap skenario pengujian dilaksanakan pada lingkungan yang sama agar hasil yang diperoleh dapat dibandingkan secara objektif. Tahapan pengujian yang dilakukan adalah sebagai berikut.

1. Menjalankan sistem BIPOL Tracker dalam kondisi normal untuk memperoleh data *baseline* sebagai acuan kondisi komunikasi tanpa adanya serangan.
2. Melakukan pengujian *False Data Injection* dengan mengirimkan paket UDP yang berisi data manipulatif menuju UDP Gateway untuk mengamati kemampuan sistem dalam membedakan paket sah dan paket tidak sah.
3. Melakukan pengujian *Replay Attack* dengan mengirimkan kembali paket UDP yang telah direkam sebelumnya untuk mengevaluasi kemampuan sistem dalam mendeteksi penggunaan ulang paket.
4. Melakukan pengujian *UDP Flood* dengan mengirimkan paket UDP dalam jumlah besar secara terus-menerus menuju server untuk mengukur dampaknya terhadap ketersediaan layanan.
5. Menerapkan mekanisme *Defense in Depth* yang terdiri atas *Rate Limiting*, HMAC yang dikombinasikan dengan *Timestamp, boot_id, sequence number,* serta *duplicate cache* pada sisi server pada sistem komunikasi UDP.
6. Mengulangi seluruh skenario pengujian setelah mekanisme keamanan diterapkan untuk memperoleh data pembanding.

7. Mencatat parameter hasil pengujian yang meliputi jumlah paket yang diterima, *latency*, *packet loss*, *throughput*, validitas data, serta kemampuan sistem dalam mendeteksi dan menolak paket yang tidak sah.
8. Melakukan analisis terhadap seluruh hasil pengujian untuk mengevaluasi efektivitas mekanisme *Defense in Depth* dalam meningkatkan keamanan komunikasi data pada sistem BIPOL Tracker.

4.4.3 Data Hasil Pengujian

a. Hasil Pengujian Kondisi Normal (*Baseline*)

Tabel 4.6 Data Normal

Parameter	Nilai *Baseline*
Jumlah Paket diterima/detik	0.6925 pps
Latency rata-rata (ms)	623.75 ms
Packet loss (%)	0%
Throughput (Kbps)	0.3025 kbps
Validitas data di dashboard	100% valid

```
[2026-05-28T10:45:14.544Z] [INFO] [UDP] B-1234-XYZ | seq=475 | -6.366982,106.822983 | 23.5 | 0 | 1544ms | loss=0.00%
[2026-05-28T10:45:15.643Z] [INFO] [UDP] B-1234-XYZ | seq=476 | -6.366968,106.822922 | 25.1 | 0 | 1643ms | loss=0.00%
[2026-05-28T10:45:16.782Z] [INFO] [UDP] B-1234-XYZ | seq=477 | -6.366957,106.822845 | 27.3 | 0 | 1782ms | loss=0.00%
[2026-05-28T10:45:17.923Z] [INFO] [UDP] B-1234-XYZ | seq=478 | -6.366918,106.822701 | 29.1 | 0 | 923ms | loss=0.00%
[2026-05-28T10:45:19.064Z] [INFO] [UDP] B-1234-XYZ | seq=479 | -6.366887,106.822617 | 30.7 | 0 | 1064ms | loss=0.00%
[2026-05-28T10:45:20.203Z] [INFO] [UDP] B-1234-XYZ | seq=480 | -6.366855,106.822540 | 30.5 | 0 | 1203ms | loss=0.00%
[2026-05-28T10:45:21.345Z] [INFO] [UDP] B-1234-XYZ | seq=481 | -6.366818,106.822464 | 30.8 | 0 | 1345ms | loss=0.00%
[2026-05-28T10:45:22.465Z] [INFO] [UDP] B-1234-XYZ | seq=482 | -6.366778,106.822403 | 29.9 | 0 | 1464ms | loss=0.00%
[2026-05-28T10:45:23.605Z] [INFO] [UDP] B-1234-XYZ | seq=483 | -6.366738,106.822342 | 28.1 | 0 | 1604ms | loss=0.00%
[2026-05-28T10:45:24.077Z] [INFO] [QoS 60s] pps=0.58 | kbps=0.26 | lat(avg/p50/p90/p99)=1366 / 1345 / 1723 / 1805 ms | invalid=0 | loss(total)=
0.00% | worst=B-1234-XYZ (0.00%)
```

Gambar 4.11 Output Log Server Kondisi Normal

Server berhasil menerima data dari perangkat secara berurutan melalui UDP. Setiap baris log menampilkan informasi waktu penerimaan, identitas perangkat, nomor urut paket, koordinat lokasi, kecepatan, serta waktu tunda pengiriman data. Kenaikan nilai sequence number secara berurutan menunjukkan bahwa data dikirim

dan diterima secara konsisten tanpa adanya paket yang terlewat pada cuplikan log tersebut. Selain itu, nilai loss sebesar 0,00% dan invalid sebesar 0 menandakan bahwa tidak terdapat paket yang hilang maupun data yang rusak selama proses pengiriman berlangsung.

Pada bagian ringkasan kinerja, nilai latency ditampilkan dalam bentuk rata-rata, p50, p90, dan p99. Nilai p50 merupakan median yang menunjukkan bahwa 50% data diterima dengan waktu tunda di bawah nilai tersebut. Nilai p90 menunjukkan bahwa 90% data diterima di bawah batas waktu tersebut, sedangkan p99 menggambarkan hampir seluruh data yang diterima server. Berdasarkan log tersebut, rentang *latency* berada pada kisaran 1,3 hingga 1,8 detik, yang menunjukkan waktu dari saat data dihasilkan oleh perangkat hingga diterima oleh server. Besarnya waktu tunda ini dapat dipengaruhi oleh interval pengiriman data, proses pembacaan GPS, buffering pada perangkat atau server, serta kondisi jaringan yang digunakan. Secara keseluruhan, log ini menunjukkan bahwa server berjalan normal, data berhasil diterima dengan baik, dan tidak terjadi kehilangan paket.

b. Hasil Pengujian Normal Setelah Defense in Depth

Tabel 4.7 Hasil Pengujian Kondisi Normal dengan *Defense in Depth*

Parameter	Nilai *Baseline*
Jumlah Paket diterima/detik	0.7075 pps
Latency rata-rata (ms)	1706.58 ms
Packet loss (%)	0%
Throughput (KBps)	0.8417 kbps
Validitas data di dashboard	100% valid

Pengujian kondisi normal dengan *Defense in Depth* dilakukan pada tanggal 1 Juni 2026 menggunakan perangkat IoT (B-1234-XYZ) yang telah dikonfigurasi dengan *secret key* yang valid. Sebanyak 70 paket UDP berhasil direkam selama sesi pengujian berlangsung, dengan nomor urut paket (*sequence number*) dari 180 hingga 249, yang menunjukkan tidak ada paket yang hilang selama sesi berlangsung. Tabel 4.7 menyajikan parameter performa komunikasi pada kondisi ini.

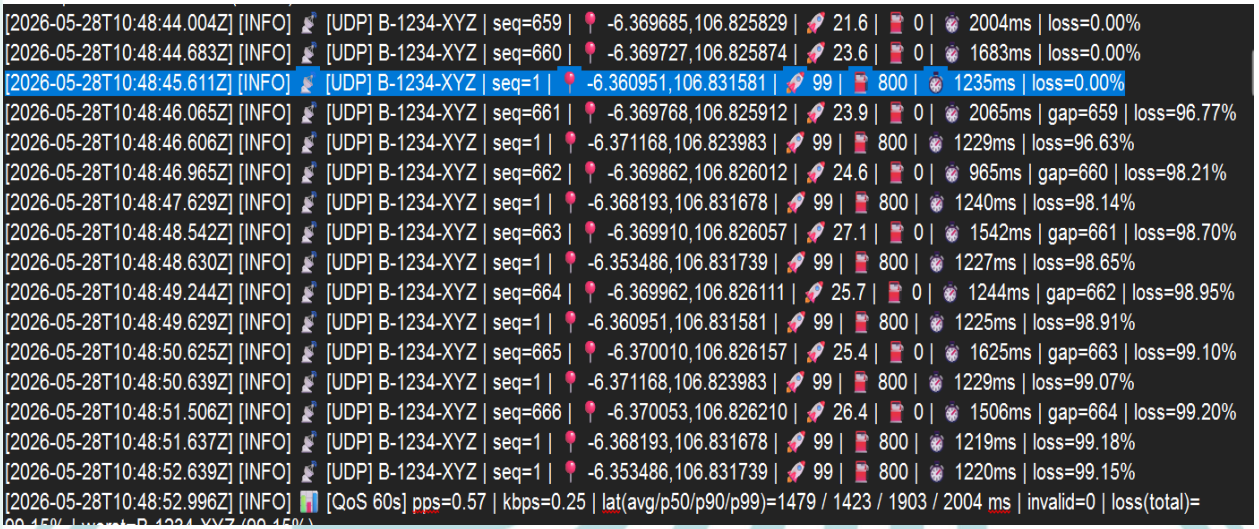
c. Hasil Pengujian False Data Injection

Pengujian skenario *false data injection* dilakukan dengan mensimulasikan pengiriman paket UDP palsu ke server BIPOL Tracker menggunakan skrip berbasis Node.js pada device penyerang. Paket yang dikirimkan identik dengan format komunikasi perangkat IoT resmi pada sistem BIPOL Tracker, yaitu *BUS_ID*,*boot_id*, *seq*, *sentTimestamp*, *latitude, longitude, speed, gas*.

Tabel 4.8 Data *False Data Injection*

Aspek yang Diamati	Hasil
Paket palsu diterima server	Ya
Data muncul di dashboard	Ya
Data tersimpan di database (Supabase)	Ya

Sistem mampu membedakan paket sah vs palsu	Tidak
Data yang di tolak	100%



Gambar 4.12 Output Log Server *False Data Injection*

Berdasarkan hasil pada Gambar 4.12, sistem BIPOL Tracker tidak menunjukkan kemampuan untuk membedakan paket yang berasal dari perangkat IoT yang sah dengan paket yang dikirimkan oleh pihak tidak berwenang. Data manipulatif berhasil melewati seluruh lapisan pemrosesan server, mulai dari penerimaan paket UDP, proses parsing dan validasi format, hingga penyimpanan ke database dan penayangan pada dashboard real-time.

Tabel 4.9 Hasil 30 Kali Pengujian FDI Sebelum *Defense in Depth*

Aspek yang Diamati	Hasil
Jumlah trial	30
Paket palsu diterima server	30/30 (100%)
Data muncul di dashboard	30/30 (100%)
Data tersimpan di database	30/30(100%)
Tingkat penolakan paket	0%

Berdasarkan Tabel 4.9, dari 30 kali pengujian *False Data Injection* diperoleh hasil bahwa seluruh paket UDP palsu berhasil diterima oleh server, ditampilkan pada dashboard, dan tersimpan di database dengan tingkat keberhasilan 100%. Hasil tersebut menunjukkan bahwa mekanisme komunikasi UDP pada sistem BIPOL Tracker sebelum penerapan *Defense in Depth* belum memiliki mekanisme autentikasi maupun verifikasi integritas paket. Akibatnya, server tidak mampu membedakan paket yang dikirim oleh perangkat IoT yang sah dengan paket palsu yang dikirim oleh penyerang. Kondisi ini menunjukkan bahwa serangan *False Data Injection* berhasil mengganggu integritas dan autentisitas data, karena informasi yang tidak valid tetap diproses sebagai data yang sah.

d. Hasil Pengujian *False Data Injection* Setelah *Defense in Depth*

Tabel 4.10 Data *False Data Injection*

Aspek yang Diamati	Hasil
Paket palsu diterima server	Tidak
Data muncul di dashboard	Tidak
Data tersimpan di database (Supabase)	Tidak
Sistem mampu membedakan paket sah vs palsu	Ya
Paket ditolak karena HMAC tidak valid	Ya

Tabel 4.11 Hasil Pengujian 30 kali pengujian Setelah FDI

Aspek yang Diamati	Hasil
Jumlah trial	30
Paket palsu diterima server	0/30(0%)
Data muncul di dashboard	0/30(0%)
Data tersimpan di database	0/30(0%)
Tingkat penolakan paket	100%

Berdasarkan Tabel 4.10, seluruh paket palsu yang dikirimkan selama 30 kali pengujian berhasil ditolak oleh server sehingga tidak ditampilkan pada *dashboard* maupun disimpan ke *database*. Tingkat penolakan paket mencapai 100%, yang menunjukkan bahwa mekanisme HMAC berhasil melakukan autentikasi dan verifikasi integritas setiap paket UDP yang diterima. Paket yang tidak memiliki nilai HMAC yang valid dianggap tidak berasal dari perangkat IoT yang sah sehingga langsung ditolak oleh server. Hasil ini menunjukkan bahwa penerapan *Defense in Depth* mampu meningkatkan keamanan komunikasi UDP dengan menjaga integritas dan autentisitas data terhadap serangan *False Data Injection*.

e. Hasil Pengujian Replay Attack

recv_ts	bus_id	seq	signature_valid	invalid_reason
2026-07-14 08:44:33.727+00	B-1234-XYZ	923	true	NULL
2026-07-14 08:44:45.185+00	B-1234-XYZ	923	true	NULL
2026-07-14 08:44:45.699+00	B-1234-XYZ	923	true	NULL
2026-07-14 08:44:46.227+00	B-1234-XYZ	923	true	NULL

Gambar 4.13 Hasil Replay Attack Sebelum Defense in Depth

Gambar 4.13 menunjukkan hasil pengujian *Replay Attack* sebelum penerapan *Defense in Depth*. Terlihat bahwa paket dengan *sequence number* yang sama diterima berulang kali oleh server. Seluruh paket memiliki status *signature_valid* = true dan *invalid_reason* = NULL, yang menunjukkan bahwa server tidak memiliki mekanisme untuk mendeteksi paket *replay* sehingga paket tetap diproses.

Tabel 4.12 Hasil Pengujian 30 kali pengujian Sebelum *Replay Attack*

Aspek yang Diamati	Hasil
Jumlah trial	30

Paket palsu diterima server	30/30 (100%)
Data muncul di dashboard	30/30 (100%)
Data tersimpan di database	30/30 (100%)
Tingkat penolakan paket replay	0%

Berdasarkan Tabel 4.12, seluruh paket *replay* yang dikirimkan selama 30 kali pengujian berhasil diterima oleh server, ditampilkan pada *dashboard*, dan disimpan ke *database*. Hasil tersebut menunjukkan bahwa sistem komunikasi UDP pada BIPOL Tracker sebelum penerapan *Defense in Depth* belum memiliki mekanisme untuk memverifikasi freshness paket yang diterima. Server memperlakukan paket lama yang dikirim ulang sebagai paket baru karena tidak terdapat mekanisme *validasi timestamp* maupun pemeriksaan duplikasi paket. Kondisi ini menunjukkan bahwa sistem rentan terhadap *Replay Attack* yang dapat menyebabkan data lama diproses kembali sehingga mengganggu autentisitas dan freshness informasi yang diterima server.

f. Hasil Pengujian Replay Attack Setelah Defense in Depth

seq int8	latency_m	signature_valid bool	invalid_reason
1145	14056	TRUE	replay_duplicate
1145	1325	TRUE	NULL
1145	13548	TRUE	replay_duplicate
1145	14570	TRUE	replay_duplicate

Gambar 4.14 Hasil Replay Attack Setelah Defense in Depth

Gambar 4.14 menunjukkan hasil pengujian *Replay Attack* setelah penerapan *Defense in Depth*. Paket pertama dengan *sequence number* 1145 diterima oleh server (*invalid_reason* = NULL). Ketika paket yang sama dikirim kembali, server memberikan status *invalid_reason* = *replay_duplicate* walaupun nilai *signature_valid*

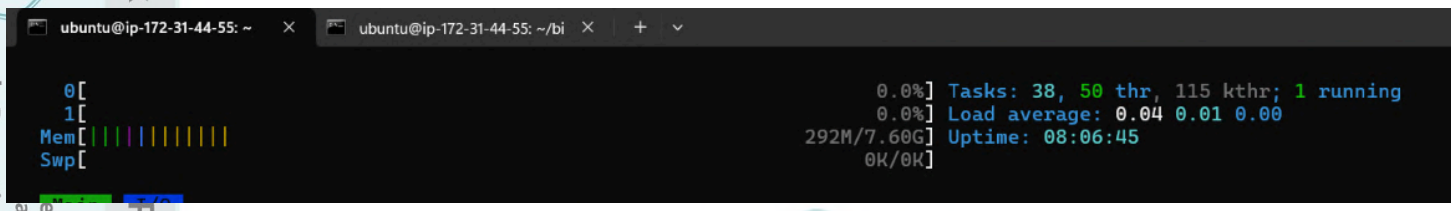
tetap bernilai *TRUE*. Hal tersebut menunjukkan bahwa integritas paket masih valid, namun server berhasil mengenali bahwa paket tersebut merupakan paket duplikat sehingga ditolak.

Tabel 4.13 Hasil Pengujian 30 kali pengujian Setelah *Replay Attack*

Aspek yang Diamati	Hasil
Jumlah trial	30
Paket palsu diterima server	0/30(0%)
Data muncul di dashboard	0/30(0%)
Data tersimpan di database	0/30(0%)
Tingkat penolakan paket replay	100%

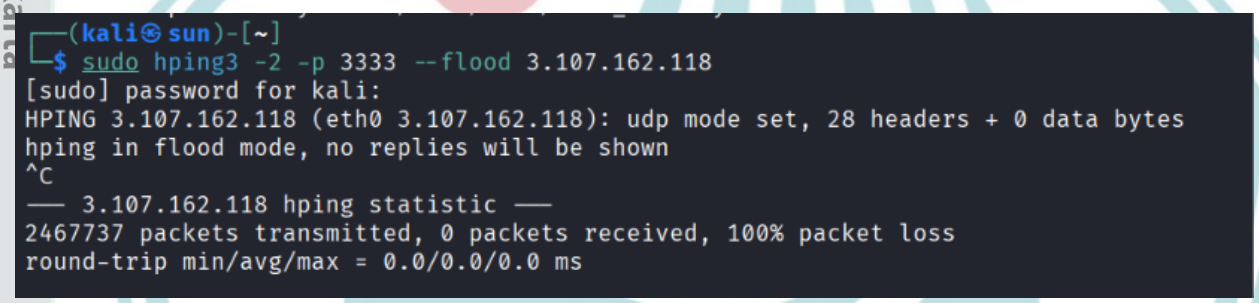
Berdasarkan Tabel 4.13, seluruh paket replay yang dikirimkan selama 30 kali pengujian berhasil ditolak oleh server sehingga tidak ditampilkan pada *dashboard* maupun disimpan ke *database*. Hasil tersebut menunjukkan bahwa mekanisme *Defense in Depth* yang terdiri atas HMAC, *timestamp*, *boot_id, sequence number,* dan *duplicate cache* mampu memverifikasi keaslian serta *freshness* setiap paket UDP yang diterima. Paket yang teridentifikasi sebagai paket lama atau paket duplikat tidak diproses lebih lanjut oleh server. Dengan demikian, *Replay Attack* tidak berhasil mempengaruhi data pada sistem BIPOL Tracker. Hasil ini menunjukkan bahwa mekanisme keamanan yang diterapkan efektif dalam menjaga autentisitas dan *freshness* komunikasi data berbasis UDP.

g. Hasil Pengujian UDP Flood



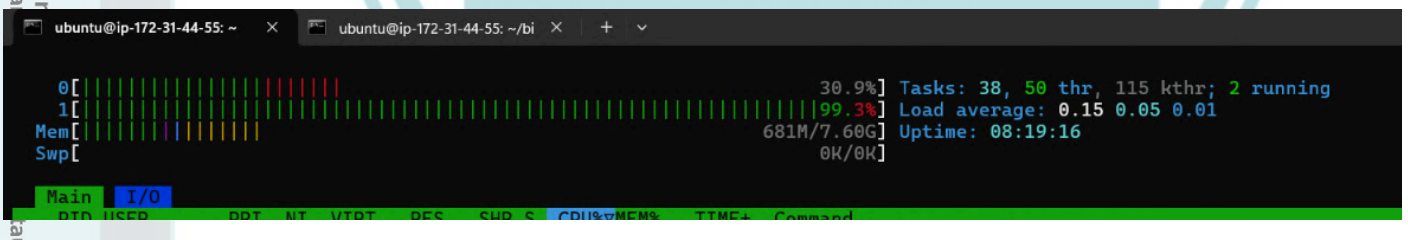
Gambar 4.15 Kondisi Server Sebelum UDP Flood

Sebelum serangan dilakukan, penggunaan CPU sebesar sekitar 0,8% dan penggunaan memori sekitar 292 MB sehingga VPS berada pada kondisi normal.



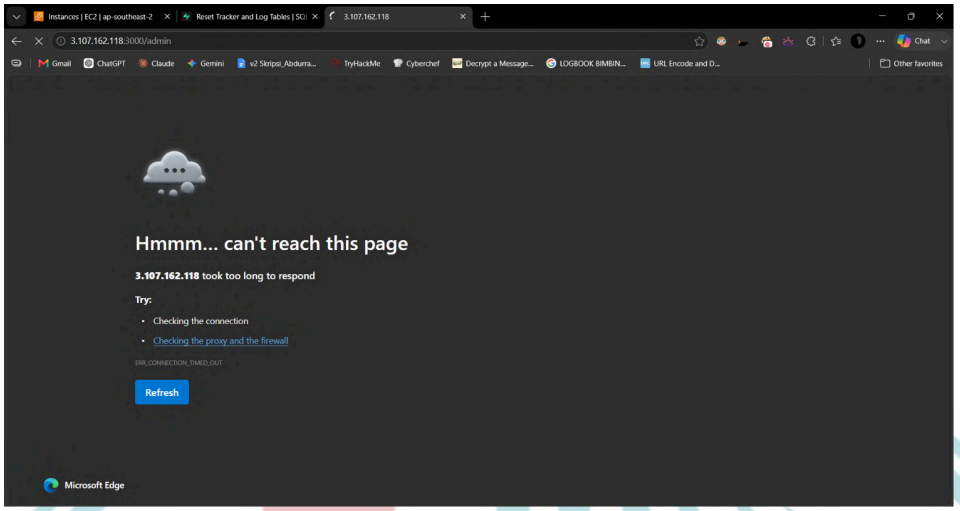
Gambar 4.16 Hasil UDP Flood menggunakan hping3

Serangan dilakukan menggunakan aplikasi hping3 dengan mode flood sehingga paket UDP dikirim secara terus-menerus menuju UDP Gateway pada port 3333, dengan total 2.467.737 packet.



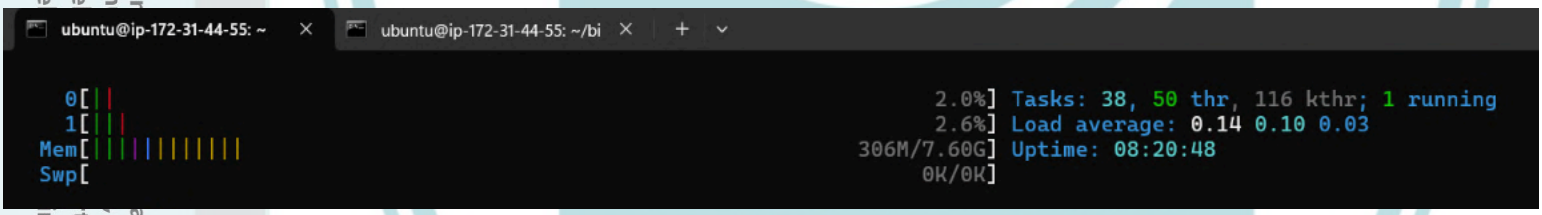
Gambar 4.17 Kondisi Server saat Saat UDP Flood

Selama serangan berlangsung, penggunaan CPU meningkat hingga sekitar 99% dan penggunaan memori meningkat menjadi sekitar 681MB.



Gambar 4.18 Dampak Terhadap Web Server

Selama UDP Flood berlangsung, aplikasi Bipol Tracker tidak dapat diakses karena server mengalami penurunan ketersediaan layanan.



Gambar 4.19 Kondisi Setelah UDP Flood

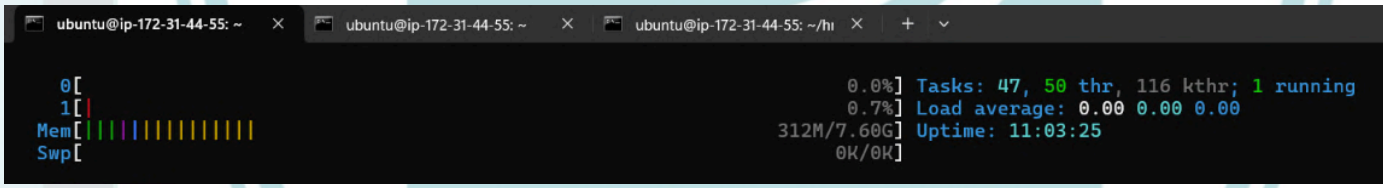
Setelah serangan dihentikan, penggunaan CPU dan memori kembali menurun sehingga server kembali mampu memberikan layanan secara normal.

Tabel 4.14 Hasil Pengujian UDP Flood

Kondisi	CPU	RAM	Kondisi Layanan
Sebelum serangan	0,8%	292MB	Berjalan Normal
Saat UDP Flood	99%	681MB	Web mengalami timeout
Setelah serangan dihentikan	2%	306MB	Berjalan kembali normal

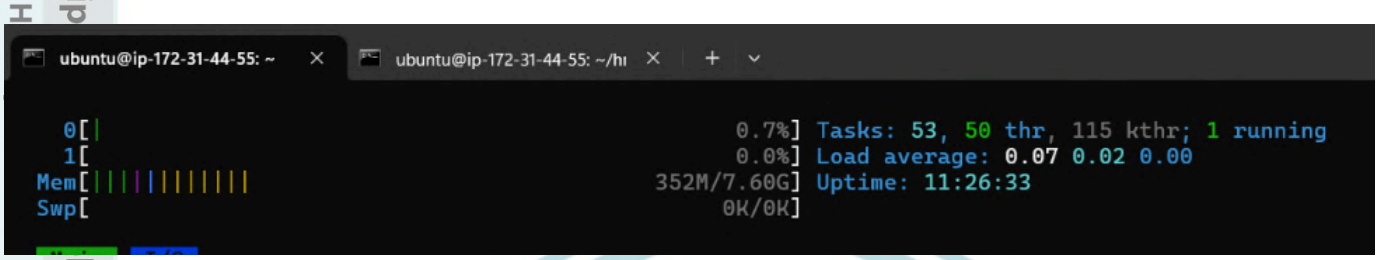
Berdasarkan Tabel 4.10, kondisi server sebelum serangan UDP Flood menunjukkan penggunaan sumber daya yang relatif rendah, yaitu CPU sebesar 0,8% dan RAM sebesar 292 MB, sehingga layanan berjalan normal. Ketika serangan UDP Flood berlangsung, penggunaan CPU meningkat secara signifikan hingga mencapai 99% dan penggunaan RAM meningkat menjadi 681 MB. Kondisi tersebut menyebabkan layanan web mengalami *timeout*, yang menunjukkan bahwa serangan berhasil mengganggu aspek *availability* pada sistem. Setelah serangan dihentikan, penggunaan CPU dan RAM kembali menurun menjadi masing-masing sebesar 2% dan 306 MB, serta layanan kembali berjalan normal. Hasil ini menunjukkan bahwa serangan *UDP Flood* memberikan beban yang sangat tinggi terhadap sumber daya server sehingga mengakibatkan penurunan ketersediaan layanan.

h. Hasil Pengujian UDP Flood Setelah Defense in Depth



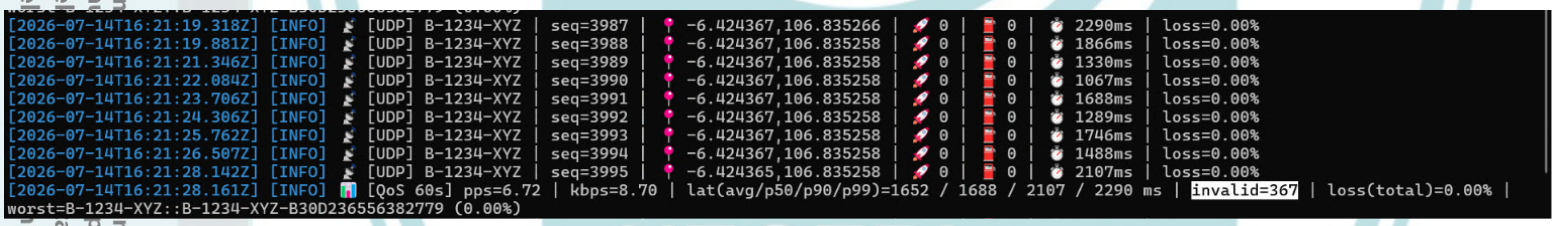
Gambar 4.20 Kondisi Server Sebelum UDP Flood

Sebelum pengujian dilakukan, penggunaan CPU berada pada kisaran 0% dengan penggunaan memori sekitar 312 MB. Kondisi tersebut menunjukkan server berada dalam keadaan normal dan siap menerima lalu lintas jaringan.



Gambar 4.21 Kondisi Server Setelah UDP Flood

Berdasarkan Gambar 4.21, penggunaan CPU tetap berada pada kisaran 0%, sedangkan penggunaan memori hanya meningkat menjadi sekitar 352 MB. Selain itu, proses Node.js UDP Gateway tetap berjalan tanpa mengalami penghentian layanan. Hasil tersebut menunjukkan bahwa mekanisme *Rate Limiting* mampu mengurangi beban pemrosesan paket *UDP Flood* sehingga sumber daya VPS tetap stabil selama pengujian berlangsung.



Gambar 4.22 Log UDP Gateway Selama Pengujian UDP Flood

Berdasarkan Gambar 4.22, *UDP Gateway* tetap menerima paket dari perangkat *Bipol Tracker* selama serangan *UDP Flood* berlangsung. *Sequence number* terus bertambah secara berurutan, sedangkan nilai loss(total) tetap sebesar 0,00%. Hal tersebut menunjukkan bahwa komunikasi perangkat yang sah masih dapat diproses oleh server meskipun terjadi peningkatan trafik akibat *UDP Flood*. Selain itu, terdapat paket yang ditandai sebagai invalid, yang menunjukkan bahwa mekanisme validasi tetap berjalan selama pengujian.

```
ubuntu@ip-172-31-44-55:~$ sudo iptables -nvL
Chain INPUT (policy ACCEPT 68012 packets, 27M bytes)
 pkts bytes target     prot opt in     out     source               destination
12476  935K ACCEPT     17   --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:3333 limit: up to 10/se
c burst 20 mode srcip
 146K 6897K DROP       17   --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:3333
```

Gambar 4.23 Hasil Rate Limiting Pada iptables

Berdasarkan Gambar 4.23, rule *ACCEPT* menunjukkan paket yang masih diperbolehkan masuk sesuai batas *Rate Limiting*, sedangkan rule *DROP* menunjukkan paket UDP yang melebihi batas telah dibuang oleh *firewall*. Jumlah paket *DROP* yang lebih besar dibandingkan paket *ACCEPT* menunjukkan bahwa mekanisme *Rate Limiting* berhasil menyaring sebagian besar lalu lintas *UDP Flood* sebelum mencapai aplikasi.

Tabel 4.15 Hasil Pengujian UDP Flood

Parameter	Hasil
Serangan	*UDP Flood* menggunakan *hping3*
Mekanisme *Defense in Depth*	*Rate Limiting (iptables)*
Memori VPS	312MB - 352MB
Status *UDP Gateway*	Tetap berjalan
Komunikasi perangkat sah	Tetap diterima
Availability	Tetap terjaga

Hasil pengujian ini menunjukkan bahwa penerapan *Rate Limiting* sebagai lapisan pertama dalam pendekatan *Defense in Depth* efektif dalam meningkatkan ketahanan sistem terhadap serangan *UDP Flood*. Meskipun mekanisme ini tidak bertujuan melindungi aspek autentikasi maupun integritas data, pembatasan laju paket yang masuk mampu menjaga ketersediaan layanan sehingga komunikasi dari perangkat IoT yang sah tetap berlangsung secara normal. Dengan demikian, setiap lapisan pada *Defense in Depth* memiliki

fungsi yang saling melengkapi, di mana *Rate Limiting* berfokus pada perlindungan *availability*, sedangkan lapisan HMAC dengan *timestamp, boot_id, sequence number,* dan *duplicate cache* berperan dalam menjaga *authenticity, integrity,* dan *freshness* komunikasi.

4.4.4 Analisis Data

a. Analisis Hasil *False Data Injection*

Berdasarkan hasil pengujian, sebelum penerapan *Defense in Depth* server UDP Bipol Tracker masih menerima paket *False Data Injection* selama format paket sesuai dengan struktur yang diharapkan. Karena belum terdapat mekanisme autentikasi dan verifikasi integritas, server tidak dapat membedakan paket yang berasal dari perangkat sah dengan paket yang dikirim oleh penyerang. Akibatnya, data palsu dapat tersimpan pada *database* sehingga mengganggu aspek *integrity* dan *authenticity* komunikasi data.

Setelah *Defense in Depth* diterapkan, paket *False Data Injection* tidak lagi diterima oleh server. Setiap paket harus melewati proses verifikasi HMAC menggunakan *secret key* yang hanya diketahui oleh perangkat Bipol Tracker dan server. Apabila nilai HMAC tidak sesuai, paket langsung ditolak sehingga tidak diteruskan ke database. Hasil ini menunjukkan bahwa mekanisme HMAC mampu memastikan keaslian pengirim sekaligus menjaga integritas data yang dikirimkan.

Berdasarkan hasil tersebut, dapat disimpulkan bahwa penerapan *Defense in Depth* pada lapisan autentikasi berhasil mengurangi risiko *False Data Injection* pada komunikasi UDP Bipol Tracker. Sebelum penerapan pengamanan, server masih menerima data palsu, sedangkan setelah penerapan HMAC hanya paket yang memiliki signature valid yang dapat diproses. Dengan demikian, penerapan mekanisme ini meningkatkan keamanan komunikasi terutama pada aspek *integrity* dan *authenticity*.

b. Analisis Hasil *Replay Attack*

Berdasarkan hasil pengujian, sebelum penerapan *Defense in Depth* server UDP Bipol Tracker masih menerima paket hasil *Replay Attack*. Paket yang sebelumnya telah dikirim dapat dikirim ulang oleh penyerang dan tetap diproses oleh server karena belum terdapat mekanisme untuk memverifikasi keterbaruan (*freshness*) paket maupun mendeteksi paket duplikat. Akibatnya, data hasil *replay* kembali tersimpan pada *database* sehingga mengganggu aspek *authenticity* dan *freshness* komunikasi data.

Setelah *Defense in Depth* diterapkan, paket *Replay Attack* tidak lagi diterima oleh server. Setiap paket harus melalui proses verifikasi HMAC, validasi timestamp, pemeriksaan boot_id, *sequence number*, serta pengecekan *duplicate cache*. Paket yang teridentifikasi sebagai paket lama atau paket yang telah diterima sebelumnya akan ditolak sehingga tidak diteruskan ke *database*. Hasil ini menunjukkan bahwa mekanisme yang diterapkan mampu memastikan bahwa hanya paket yang masih valid dan belum pernah diproses yang dapat diterima oleh sistem.

Berdasarkan hasil tersebut, dapat disimpulkan bahwa penerapan *Defense in Depth* berhasil mengurangi risiko *Replay Attack* pada komunikasi UDP Bipol Tracker. Sebelum penerapan pengamanan, paket *replay* masih dapat diterima dan diproses oleh server, sedangkan setelah mekanisme validasi diterapkan seluruh paket *replay* berhasil ditolak. Dengan demikian, penerapan HMAC yang dikombinasikan dengan validasi *timestamp*, *boot_id*, *sequence number,* dan *duplicate cache* meningkatkan keamanan komunikasi terutama pada aspek *authenticity* dan *freshness*, sehingga paket yang dikirim ulang tidak dapat digunakan untuk memanipulasi data sistem.

c. Analisis Hasil *UDP Flood*

Berdasarkan hasil pengujian, sebelum penerapan *Defense in Depth* server UDP Bipol Tracker menerima seluruh paket yang dikirim

selama serangan *UDP Flood*. Kondisi tersebut menyebabkan peningkatan beban pemrosesan pada server sehingga berpotensi menurunkan kualitas layanan apabila serangan berlangsung secara terus-menerus. Kerentanan ini terutama mempengaruhi aspek *availability*, karena sumber daya server digunakan untuk memproses paket dalam jumlah besar tanpa adanya mekanisme pembatasan laju paket.

Setelah *Defense in Depth* diterapkan, mekanisme *rate limiting* pada sisi VPS berhasil membatasi jumlah paket UDP yang diproses oleh server. Paket yang melebihi batas yang telah ditentukan secara otomatis dibuang (*drop*), sedangkan paket yang memenuhi ketentuan tetap diproses. Berdasarkan hasil pengujian menggunakan *htop*, *log server*, dan pengamatan terhadap aplikasi Bipol Tracker, layanan masih dapat menerima data dari perangkat IoT dan tetap dapat diakses selama pengujian berlangsung. Hasil ini menunjukkan bahwa penerapan *rate limiting* mampu mengurangi dampak *UDP Flood* terhadap ketersediaan layanan.

Berdasarkan hasil tersebut, dapat disimpulkan bahwa penerapan *Defense in Depth* melalui mekanisme *rate limiting* berhasil meningkatkan aspek *availability* pada komunikasi UDP Bipol Tracker. Meskipun serangan *UDP Flood* masih menghasilkan lalu lintas jaringan yang tinggi, server tetap mampu mempertahankan layanan dengan membatasi paket yang diproses sehingga dampak serangan terhadap operasional sistem dapat diminimalkan.

d. Analisis Efektivitas Defense in Depth

Berdasarkan seluruh hasil pengujian, penerapan *Defense in Depth* terbukti meningkatkan keamanan komunikasi UDP pada sistem Bipol Tracker melalui penerapan beberapa lapisan pengamanan yang saling melengkapi. Setiap lapisan memiliki fungsi yang berbeda dalam menghadapi ancaman yang diuji sehingga tidak bergantung pada satu mekanisme keamanan saja. Pendekatan ini sesuai dengan konsep *Defense in Depth* yang menerapkan beberapa kontrol keamanan untuk mengurangi risiko apabila satu lapisan perlindungan tidak mampu menghentikan seluruh jenis serangan.

Pada pengujian *False Data Injection*, mekanisme HMAC berhasil memastikan bahwa hanya perangkat yang memiliki *secret key* yang dapat menghasilkan *signature* yang valid sehingga paket palsu ditolak oleh server. Pada pengujian *Replay Attack*, validasi HMAC, *timestamp*, *boot_id*, *sequence number*, dan *duplicate cache* berhasil mendeteksi paket yang dikirim ulang sehingga paket *replay* tidak diproses kembali. Sementara itu, pada pengujian *UDP Flood*, penerapan *rate limiting* berhasil membatasi jumlah paket UDP yang diproses oleh server sehingga layanan tetap dapat menerima data dari perangkat Bipol Tracker selama pengujian berlangsung.

Hasil tersebut menunjukkan bahwa penerapan *Defense in Depth* mampu meningkatkan aspek *integrity, authenticity, freshness*, dan *availability* pada komunikasi UDP Bipol Tracker. Oleh karena itu, tujuan penelitian untuk menganalisis penerapan *Defense in Depth* sebagai metode peningkatan keamanan komunikasi UDP telah tercapai berdasarkan hasil pengujian terhadap ketiga skenario serangan.

Tabel 4.16 Hasil Pengujian Sebelum dan Setelah Penerapan Defense in Depth

Jenis Serangan	Sebelum *Defense in Depth*	Setelah *Defense in Depth*	Mekanisme Pertahanan
False Data Injection	Paket palsu diterima server	Paket ditolak	HMAC
Replay Attack	Paket *replay* diterima server	Paket *replay* ditolak	HMAC, *timestamp, boot_id, sequence number*, duplicate cache
UDP Flood	Beban server meningkat dan seluruh paket diproses	Paket berlebih dibatasi (*drop*) sehingga layanan tetap berjalan	*Rate limiting* (iptables)

Berdasarkan Tabel 4.12, penerapan *Defense in Depth* berhasil meningkatkan keamanan komunikasi data pada sistem BIPOL Tracker. Sebelum mekanisme keamanan diterapkan, paket hasil *False Data Injection* dan *Replay Attack* masih dapat diterima oleh server, sedangkan serangan *UDP Flood* menyebabkan peningkatan beban server. Setelah penerapan HMAC, *timestamp, boot_id, sequence number, duplicate cache,* dan *Rate Limiting*, paket tidak sah berhasil ditolak, paket *replay* dapat dicegah, serta paket UDP berlebih dibatasi sehingga layanan tetap berjalan normal. Hasil ini menunjukkan bahwa setiap lapisan *Defense in Depth* saling melengkapi dalam menjaga aspek *integrity, authenticity, freshness,* dan *availability* pada komunikasi data berbasis UDP.

e. Analisis UDP Sebelum dan Sesudah Defense in Depth

Pada subbab ini dilakukan analisis terhadap kualitas komunikasi UDP pada kondisi normal, yaitu sebelum dan sesudah penerapan *Defense in Depth* tanpa adanya skenario serangan. Analisis dilakukan untuk mengetahui apakah mekanisme keamanan yang diterapkan memberikan pengaruh terhadap kualitas komunikasi data antara perangkat IoT dan server. Parameter yang dianalisis meliputi

jumlah paket diterima per detik, latency rata-rata, packet loss, dan throughput.

Tabel 4.17 Perbandingan Kinerja Komunikasi UDP pada Kondisi Normal

Parameter	Sebelum Defense in Depth	Sesudah Defense in Depth
Jumlah Paket diterima/detik	0.6925 pps	0.7075 pps
Latency rata-rata (ms)	623.75 ms	1706.58 ms
Packet loss (%)	0%	0%
Throughput (Kbps)	0.3025 kbps	0.8417 kbps

Berdasarkan hasil pengujian pada kondisi normal, penerapan *Defense in Depth* tidak memberikan dampak yang signifikan terhadap kinerja komunikasi UDP. Jumlah paket diterima per detik, *latency, packet loss,* dan *throughput* tetap berada pada rentang yang stabil. Dengan demikian, mekanisme keamanan yang diterapkan mampu meningkatkan aspek autentikasi, integritas, dan *freshness* tanpa mengurangi kualitas komunikasi data pada sistem BIPOL Tracker. Hasil ini menjadi dasar untuk mengevaluasi efektivitas *Defense in Depth* pada pengujian *False Data Injection, Replay Attack,* dan *UDP Flood.*

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil analisis keamanan komunikasi data berbasis UDP pada sistem BIPOL Tracker menggunakan pendekatan *Defense in Depth*, diperoleh beberapa kesimpulan sebagai berikut.

1. Komunikasi data berbasis UDP pada sistem BIPOL Tracker sebelum penerapan mekanisme keamanan memiliki kerentanan terhadap serangan *False Data Injection, Replay Attack*, dan *UDP Flood*. Tidak adanya mekanisme autentikasi, verifikasi integritas, serta perlindungan terhadap paket yang dikirim ulang menyebabkan server tidak mampu membedakan paket yang sah dengan paket hasil manipulasi, sehingga berpotensi mengganggu aspek *integrity, authenticity, freshness,* dan *availability.*

   Hasil pengujian menunjukkan bahwa serangan *False Data Injection* sebelum penerapan *Defense in Depth* berhasil mengirimkan data GPS palsu ke server sehingga data tersebut diteruskan ke aplikasi dan *database*. Setelah diterapkan mekanisme HMAC, server mampu memverifikasi keaslian dan integritas setiap paket sehingga paket tanpa autentikasi yang valid ditolak dan tidak diproses lebih lanjut.

   Pada pengujian *Replay Attack*, paket yang sebelumnya sah masih dapat diterima kembali oleh server sebelum mekanisme keamanan diterapkan. Setelah penerapan HMAC, *timestamp, boot_id, sequence number*, dan *duplicate cache*, server mampu mendeteksi paket yang telah digunakan sebelumnya maupun paket yang telah melewati batas waktu valid, sehingga paket replay yang digunakan pada skenario pengujian berhasil dideteksi dan ditolak oleh server, sehingga aspek freshness komunikasi dapat dipertahankan. dan aspek *freshness* komunikasi dapat dipertahankan.

   Hasil pengujian *UDP Flood* menunjukkan bahwa serangan mampu meningkatkan beban server secara signifikan pada kondisi tanpa mekanisme perlindungan. Setelah diterapkan *Rate Limiting* sebagai lapisan pertama *Defense in Depth*, server tetap mampu mempertahankan ketersediaan layanan selama skenario pengujian berlangsung, dengan

membatasi jumlah paket UDP yang diproses dari sumber yang sama. Meskipun sejumlah paket dibuang (*drop*), layanan komunikasi tetap berjalan dan server tetap dapat memproses paket yang memenuhi kebijakan keamanan.

2. Penerapan *Defense in Depth* yang terdiri atas *Rate Limiting* sebagai lapisan pertama serta HMAC yang dikombinasikan dengan *timestamp, boot_id, sequence number, dan duplicate cache* sebagai lapisan kedua terbukti meningkatkan keamanan komunikasi data pada sistem BIPOL Tracker. Setiap mekanisme memberikan perlindungan terhadap ancaman yang berbeda dan saling melengkapi, sehingga mampu meningkatkan *integrity, authenticity, freshness,* serta *availability* komunikasi UDP dibandingkan kondisi sebelum mekanisme keamanan diterapkan. Hasil penelitian ini menunjukkan bahwa pendekatan keamanan berlapis lebih efektif dibandingkan penggunaan satu mekanisme pengamanan saja dalam melindungi komunikasi IoT berbasis UDP.

5.2 Saran

Penelitian selanjutnya dapat mempertimbangkan penggunaan *Private APN* yang dipadukan dengan mekanisme *IP whitelist* pada server sehingga hanya perangkat IoT dengan alamat IP yang telah diotorisasi yang dapat mengakses layanan UDP. Pendekatan tersebut berpotensi meningkatkan pengendalian akses jaringan serta mengurangi risiko komunikasi dari sumber yang tidak sah. Selain itu, penggunaan perangkat IoT yang mendukung jaringan LTE atau teknologi seluler yang lebih baru dapat dipertimbangkan agar memungkinkan implementasi protokol *Datagram Transport Layer Security* (DTLS) pada komunikasi berbasis UDP. Alternatif lain yang dapat dievaluasi adalah migrasi ke protokol *Message Queuing Telemetry Transport Secure* (MQTTS) yang menyediakan mekanisme keamanan berbasis *Transport Layer Security* (TLS), sehingga keamanan komunikasi data IoT dapat terus ditingkatkan.

DAFTAR PUSTAKA

Adam, M., Hammoudeh, M., Alrawashdeh, R. and Alsulaimy, B., 2024. A survey on security, privacy, trust, and architectural challenges in IoT systems. IEEE Access, 12, pp.57128–57149. https://doi.org/10.1109/ACCESS.2024.3382709.

A. El Hanjri, A. Sekkaki, dan A. Erroutbi, "Secure and Lightweight HMAC Mutual Authentication Protocol for Communication between IoT Devices and Fog Nodes," dalam 2020 International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET), 2020, hlm. 251–257. https://doi.org/10.1109/IRASET48871.2020.9092215.

Al-Shareeda, M.A., Manickam, S., Laghari, S.A. dan Jaisan, A. (2022) 'Replay-attack detection and prevention mechanism in Industry 4.0 landscape for secure SECS/GEM communications', *Sustainability*, 14(23), 15900. https://doi.org/10.3390/su142315900.

A. Mosteiro-Sanchez, M. Barcelo, J. Astorga, dan A. Urbieta, "Securing IIoT Using Defence-in-Depth: Towards an End-to-End Secure Industry 4.0," Journal of Manufacturing Systems, vol. 57, hlm. 367–378, 2020. https://doi.org/10.1016/j.jmsy.2020.10.011.

B. Bostami, M. Ahmed, dan S. Choudhury, "False Data Injection Attacks in Internet of Things," dalam Performability in Internet of Things, F. Al-Turjman, Ed. Cham, Switzerland: Springer International Publishing, 2019, hlm. 47–58. https://doi.org/10.1007/978-3-319-93557-7_4.

Dadel, A. and Ansari, G.M., 2023. Cyber security challenges in the Internet of Things (IoT). International Journal of Advanced Research in Science, Communication and Technology, pp.1–6. https://doi.org/10.48175/IJARSCT-28743.

Habib, A.A., Hasan, M.K., Alkhayyat, A., Islam, S., Sharma, R. and Alkwai, L.M., 2023. False data injection attack in smart grid cyber physical system: Issues, challenges, and future direction. Computers and Electrical Engineering, 107, 108638. https://doi.org/10.1016/j.compeleceng.2023.108638.

Fereidouni, H., Fadeitcheva, O. and Zalai, M., 2025. IoT and man-in-the-middle attacks. Security and Privacy, 8, e70016. https://doi.org/10.1002/spy2.70016.

Kumar, S., Kumar, D., Dangi, R., Choudhary, G., Dragoni, N. and You, I., 2024. A review of lightweight security and privacy for resource-constrained IoT devices. Computers, Materials and Continua, 78(1), pp.31–63. https://doi.org/10.32604/cmc.2023.047084.

Khan, N., Awang, A. and Abdul Karim, S.A., 2022. Security in Internet of Things: A review. IEEE Access, 10, pp.103947–103971. https://doi.org/10.1109/ACCESS.2022.3209355.

Mrdovic, S. & Perunicic, B., 2008. *Kerckhoffs' principle for intrusion detection*. Dalam: *Proceedings of Networks 2008 – The 13th International Telecommunications Network Strategy and Planning Symposium*, Budapest, Hungary, pp.1–8. https://doi.org/10.1109/NETWKS.2008.6231360.

Pritika, Shanmugam, B. and Azam, S., 2023. Risk assessment of heterogeneous IoMT devices: A review. Technologies, 11(1), 31. https://doi.org/10.3390/technologies11010031.

Schiller, E., Aidoo, A., Fuhrer, J., Stahl, J., Ziörjen, M. and Stiller, B., 2022. Landscape of IoT security. Computer Science Review, 44, 100467. https://doi.org/10.1016/j.cosrev.2022.100467.

Sebestyen, H., Popescu, D.E. and Zmaranda, R.D., 2025. A literature review on security in the Internet of Things: Identifying and analysing critical categories. Computers, 14(2), 61. https://doi.org/10.3390/computers14020061.

Şeker, Ö., Dalkılıç, G., & Çabuk, U. C. (2023). MARAS: Mutual Authentication and Role-Based Authorization Scheme for Lightweight Internet of

Things Applications. Sensors, 23(12), 5674. https://doi.org/10.3390/s23125674.

Silberschatz, A., Korth, H.F. & Sudarshan, S. (2019). Database System Concepts. 7th ed. New York: McGraw-Hill Education.

S. Javanmardi, M. Ghahramani, M. Shojafar, M. Alazab, dan A. M. Caruso, "M-RL: A Mobility and Impersonation-Aware IDS for DDoS UDP Flooding Attacks in IoT-Fog Networks," Computers & Security, vol. 140, Art. no. 103778, 2024. https://doi.org/10.1016/j.cose.2024.103778.

S. Lazzaro, V. De Angelis, A. M. Mandalari, dan F. Buccafurri, "Is Your Kettle Smarter Than a Hacker? A Scalable Tool for Assessing Replay Attack Vulnerabilities on Consumer IoT Devices," dalam 2024 IEEE International Conference on Pervasive Computing and Communications (PerCom), Biarritz, France, 2024, hlm. 114–124. https://doi.org/10.1109/PerCom59722.2024.10494466.

Williams, P., Dutta, I. K., Daoud, H., & Bayoumi, M. (2022). A survey on security in internet of things with a focus on the impact of emerging technologies. Internet of Things, 19, 100564. https://doi.org/10.1016/j.iot.2022.100564.

Wójcicki, K., Biegańska, M., Paliwoda, B. and Górna, J., 2022. Internet of Things in industry: Research profiling, application, challenges and opportunities A review. Energies, 15(5), 1806. https://doi.org/10.3390/en15051806.

DAFTAR RIWAYAT HIDUP

Abdurrahman Ammar Ihsan

Lahir di Jakarta, 27 Juni 2004. Penulis menyelesaikan pendidikan dasar di SDN Kebon Baru 09 pada tahun 2016, pendidikan menengah pertama di SMPN 265 Jakarta pada tahun 2019, dan pendidikan menengah atas di SMK Tunas Bangsa Depok (DECES) pada tahun 2022. Sejak tahun 2022 penulis menempuh pendidikan pada Program Studi Sarjana Terapan Teknik Multimedia dan Jaringan, Jurusan Teknik Informatika dan Komputer, Politeknik Negeri Jakarta. Selama masa perkuliahan, penulis memiliki minat pada bidang *Internet of Things* (IoT), Jaringan Komputer, dan Keamanan Siber.

LAMPIRAN

Lampiran 1 - Source Code Pengujian *False Data Injection*

```
const dgram = require("dgram");
const client = dgram.createSocket("udp4");

// Konfigurasi
const BUS_ID = "B-1234-XYZ";
const TARGET_PORT = PORT_TARGET; // Sesuaikan dengan .env backend jika beda
const TARGET_HOST = "HOST_TARGET";

// Daftar Koordinat Halte (Berdasarkan gambar geofences)
// Urutan: St. UI -> PNJ -> Pondok Cina -> Menwa -> Balik lagi
const routePoints = [
    { name: "Halte St. UI", lat: -6.360951, lng: 106.831581 },
    { name: "Halte PNJ", lat: -6.371168, lng: 106.823983 },
    { name: "Halte Pondok Cina", lat: -6.368193, lng: 106.831678 },
    { name: "Halte Menwa", lat: -6.353486, lng: 106.831739 }
];

let currentIndex = 0;
const GAS_LEVEL = 800; // Statik di bawah 600 sesuai request
const SPEED = 99.0;    // Speed pura-pura jalan

console.log(`Starting UDP Blink Simulation for ${BUS_ID}...`);
console.log("Press Ctrl+C to stop");

setInterval(() => {
    // Ambil koordinat saat ini
    const point = routePoints[currentIndex];

    // build CSV: DEVICE_ID,SEQ,SENT_TS_MS,LAT,LNG,SPEED,GAS_LEVEL
    // Penting: Gunakan TITIK (.) untuk desimal, KOMA (,) hanya untuk pemisah data
                                                const payload =
`${BUS_ID},1,${Date.now()},${point.lat},${point.lng},${SPEED},${GAS_LEVEL}`;

    const message = Buffer.from(payload);

    client.send(message, TARGET_PORT, TARGET_HOST, (err) => {
        if (err) {
```

```
            console.error("Gagal kirim:", err);
        } else {
                    console.log(`[${new Date().toLocaleTimeString()}] Sent to ${point.name}:
${payload}`);
        }
    });

    // Pindah ke titik berikutnya (looping)
    currentIndex = (currentIndex + 1) % routePoints.length;

}, 1000); // Kirim setiap 1 detik agar sempat terlihat di map
```

Lampiran 2 - Source Code Pengujian *UDP Flood*

```
const dgram = require("dgram");
const client = dgram.createSocket("udp4");

const TARGET_HOST = "TARGET_HOST";
const TARGET_PORT = TARGET_PORT;

const BUS_COUNT = 500;

console.log(`Simulating ${BUS_COUNT} buses...`);

setInterval(() => {

    for (let i = 0; i < BUS_COUNT; i++) {

        const lat = -6.36 + (Math.random() * 0.02);
        const lng = 106.82 + (Math.random() * 0.02);

        const payload =
            `BUS-${i},1,${Date.now()},${lat},${lng},80,700`;

        client.send(
            Buffer.from(payload),
            TARGET_PORT,
            TARGET_HOST
        );
    }
```

```
  console.log(
    `[${new Date().toLocaleTimeString()}] ${BUS_COUNT} buses sent`
  );

}, 1000);
```

Lampiran 3 - Source Code Pengujian *Replay Attack*

```
const dgram = require('dgram');

const client = dgram.createSocket('udp4');

// Attacker-mode replay test:
// - no .env
// - no access to HMAC secrets
// - only re-send captured raw packet bytes
const CONFIG = {
 targetHost: '127.0.0.1',
 targetPort: 3333,
 fastReplayDelayMs: 500,
 staleReplayDelayMs: 500,
 rawPacket: 'MASUKAN_RAW_PAKET'
};

function sleep(ms) {
 return new Promise((resolve) => setTimeout(resolve, ms));
}
function sendUdp(payload, label) {
 return new Promise((resolve, reject) => {
  client.send(Buffer.from(payload), CONFIG.targetPort, CONFIG.targetHost, (err) => {
   if (err) return reject(err);
   console.log(`[${new Date().toISOString()}] ${label} -> ${payload}`);
   resolve();
  });
 });
}
async function run() {
 try {
  if (!CONFIG.rawPacket || !String(CONFIG.rawPacket).trim()) {
   throw new Error('Set CONFIG.rawPacket first with a captured UDP payload string.');
```

```
  }

  console.log('Replay attack test started');
  console.log(`Target UDP: ${CONFIG.targetHost}:${CONFIG.targetPort}`);
  console.log('Mode: replay existing RAW_PACKET (captured traffic only)');
    console.log(`Fast replay delay: ${CONFIG.fastReplayDelayMs}ms | stale replay delay:
${CONFIG.staleReplayDelayMs}ms`);

  const packet = String(CONFIG.rawPacket).trim();
  await sendUdp(packet, '1) original/captured packet');
  await sleep(CONFIG.fastReplayDelayMs);
  await sendUdp(packet, '2) fast replay (should pass if still inside max age window)');
  await sleep(CONFIG.staleReplayDelayMs);
  await sendUdp(packet, '3) stale replay (should be rejected as stale_timestamp)');

  console.log('\nCheck backend log / packet log table:');
  console.log('- Packet #2 usually still accepted in current implementation (duplicate seq only
flagged)');
        console.log('- Packet #3 should be rejected when timestamp age exceeds
UDP_HMAC_MAX_AGE_MS');
 } catch (err) {
  console.error('Replay test failed:', err.message);
  process.exitCode = 1;
 } finally {
  client.close();
 }
}

run();
```