

Evaluasi Kinerja Latensi Protokol MQTT dan Telegram Bot pada Sistem Notifikasi Intrusion Detection System (IDS) Berbasis Snort

Narendra Nur Anggita¹, Imam Suharjo²

^{1,2}Informatika, Universitas Mercu Buana Yogyakarta

Email: 1221110001@student.mercubuana-yogya.ac.id, imam@mercubuana-yogya.ac.id

Email Penulis Korespondensi: 1221110001@student.mercubuana-yogya.ac.id

Abstrak– Pertumbuhan jaringan komputer dan Internet of Things (IoT) meningkatkan kompleksitas komunikasi data sekaligus memperbesar potensi ancaman keamanan siber, sehingga Intrusion Detection System (IDS) menjadi komponen penting untuk memantau lalu lintas jaringan secara real-time. Penelitian ini menggunakan Snort sebagai IDS dan bertujuan menganalisis serta membandingkan latensi pengiriman notifikasi antara protokol Message Queuing Telemetry Transport (MQTT) dan Telegram Bot. Dibandingkan penelitian terdahulu yang umumnya mengkaji MQTT dan Telegram Bot secara terpisah, penelitian ini menghadirkan pengujian komparatif langsung pada satu rangkaian eksperimen IDS berbasis Snort dengan tiga skenario serangan berbeda, serta melengkapi analisis rata-rata dengan ukuran sebaran data (median, minimum, maksimum, standar deviasi) untuk menggambarkan konsistensi kinerja tiap protokol. Metode yang digunakan adalah pendekatan kuantitatif eksperimental terhadap tiga skenario serangan, yaitu ICMP, Nmap SYN Scan, dan TCP SYN Flood, masing-masing dalam satu sesi pengujian pada lingkungan virtual berbasis VMware Workstation 17 Pro dengan dua mesin virtual Kali Linux. Latensi diukur dari selisih waktu antara alert terdeteksi oleh Snort dan notifikasi diterima pengguna, kemudian dianalisis menggunakan statistik deskriptif (rata-rata, median, minimum, maksimum, dan standar deviasi). Hasil penelitian menunjukkan MQTT memiliki rata-rata waktu pengiriman alert yang lebih rendah dan lebih konsisten dibandingkan Telegram Bot pada seluruh skenario. Pada serangan Nmap SYN Scan dan SYN Flood, rata-rata latensi MQTT masing-masing 0,15 detik dan 0,38 detik dengan standar deviasi di bawah 0,15 detik, sedangkan Telegram Bot memerlukan rata-rata 41,65 detik dan 47,31 detik dengan standar deviasi di atas 24 detik. Dari sisi Alert Delivery Rate, MQTT meneruskan seluruh alert tanpa kehilangan data, sedangkan Telegram Bot hanya menerima 2,24% alert pada serangan Nmap dan 0,07% pada SYN Flood. Karena analisis pada penelitian ini bersifat deskriptif, istilah “perbedaan besar” pada hasil merujuk pada selisih nilai yang teramati, bukan hasil pengujian signifikansi statistik formal. Berdasarkan hasil tersebut, MQTT lebih sesuai digunakan sebagai media notifikasi pada sistem IDS yang membutuhkan kecepatan dan keandalan tinggi, sedangkan Telegram Bot lebih cocok untuk kebutuhan monitoring dengan frekuensi alert rendah. Penelitian lanjutan disarankan menggunakan uji statistik inferensial serta protokol pembandingan tambahan untuk memperkuat generalisasi temuan.

Kata Kunci: IDS, Snort, MQTT, Telegram Bot, Latensi, Keamanan Jaringan, Alert Delivery Rate

Abstract– The growth of computer networks and the Internet of Things (IoT) has increased the complexity of data communication and expanded the potential for cybersecurity threats, making an Intrusion Detection System (IDS) an essential component for monitoring network traffic in real time. This study uses Snort as the IDS and aims to analyze and compare the notification delivery latency between the MQTT protocol and a Telegram Bot. Unlike prior studies that generally examine MQTT and Telegram Bot separately, this study provides a direct comparative evaluation within a single Snort-based IDS experiment across three attack scenarios, and complements average latency with dispersion measures (median, minimum, maximum, standard deviation) to characterize the consistency of each protocol. A quantitative experimental approach was applied to three attack scenarios, namely ICMP, Nmap SYN Scan, and TCP SYN Flood, each conducted in a single testing session within a virtual environment built using VMware Workstation 17 Pro with two Kali Linux virtual machines. Latency was measured as the time difference between alert detection by Snort and notification receipt by the user, then analyzed using descriptive statistics (mean, median, minimum, maximum, and standard deviation). The results show that MQTT delivered alerts with lower and more consistent latency than the Telegram Bot across all scenarios. For the Nmap SYN Scan and SYN Flood attacks, the average MQTT latency was 0.15 seconds and 0.38 seconds with standard deviations below 0.15 seconds, while the Telegram Bot required 41.65 seconds and 47.31 seconds with standard deviations above 24 seconds. In terms of Alert Delivery Rate, MQTT forwarded every published alert without loss, whereas the Telegram Bot only received 2.24% of alerts during the Nmap attack and 0.07% during the SYN Flood attack. As the analysis is descriptive in nature, references to “large differences” denote the observed magnitude rather than the result of formal significance testing. These findings indicate that MQTT is more suitable for IDS notification systems that require high speed and reliability, while the Telegram Bot is better suited for monitoring needs with a lower alert frequency. Future work is recommended to apply inferential statistical testing and additional comparator protocols to strengthen the generalizability of the findings.

Keywords: IDS, Snort, MQTT, Telegram Bot, Latency, Network Security, Alert Delivery Rate

1. PENDAHULUAN

Pertumbuhan teknologi jaringan komputer dan Internet of Things (IoT) dalam beberapa tahun terakhir telah mendorong peningkatan kompleksitas sistem komunikasi data serta memperbesar potensi ancaman keamanan siber. Estimasi menunjukkan bahwa jumlah perangkat IoT yang terhubung akan mencapai puluhan miliar dalam satu dekade ke depan, yang secara langsung memperluas attack surface bagi pihak yang melakukan serangan [1]. Kondisi tersebut mendorong meningkatnya kebutuhan akan sistem keamanan jaringan yang mampu mendeteksi serangan dengan cepat dan akurat. Intrusion Detection System (IDS) merupakan salah satu solusi yang banyak digunakan untuk memantau lalu lintas jaringan serta mengidentifikasi aktivitas mencurigakan secara real-time [2].

Salah satu IDS yang banyak digunakan adalah Snort, yaitu sistem berbasis signature yang mendukung analisis paket secara mendalam (deep packet inspection) [3]. Namun demikian, efektivitas IDS tidak hanya ditentukan oleh kemampuan deteksi, tetapi juga oleh kecepatan sistem dalam menyampaikan notifikasi kepada administrator. Penelitian menunjukkan bahwa keterlambatan dalam penyampaian informasi hasil deteksi dapat memperlambat respons terhadap serangan dan meningkatkan potensi dampak yang ditimbulkan, terutama pada lingkungan yang membutuhkan penanganan ancaman secara real-time [4].

Dalam mendukung sistem notifikasi IDS, pemilihan protokol komunikasi menjadi faktor penting yang memengaruhi performa sistem, khususnya dalam hal latensi pengiriman pesan. Salah satu protokol yang banyak digunakan adalah Message Queuing Telemetry Transport (MQTT), yang dirancang sebagai protokol ringan berbasis publish-subscribe untuk mendukung komunikasi dengan latensi rendah dan efisiensi bandwidth tinggi [5][6]. Hasil evaluasi performa menunjukkan bahwa MQTT memiliki efisiensi komunikasi yang lebih baik dibandingkan pendekatan berbasis request-response seperti HTTP pada berbagai skenario IoT [7][8].

Di sisi lain, penggunaan layanan berbasis API seperti Telegram Bot menjadi alternatif populer dalam sistem notifikasi karena kemudahan integrasi dan ketersediaan infrastruktur berbasis cloud. Telegram Bot memungkinkan pengiriman pesan secara cepat kepada pengguna melalui jaringan internet dengan implementasi yang relatif sederhana. Beberapa penelitian menunjukkan bahwa Telegram efektif digunakan dalam sistem monitoring dan notifikasi berbasis jaringan secara real-time [9][10], meskipun performanya dipengaruhi oleh kondisi jaringan serta ketergantungan terhadap layanan pihak ketiga dan kebijakan rate limit dari penyedia layanan [11][12].

a. Posisi Penelitian terhadap Studi Terdahulu (Research Gap)

Untuk memperjelas posisi penelitian ini secara objektif, Tabel 1 merangkum beberapa penelitian terdahulu yang relevan beserta fokus, metode, dan celah yang masih terbuka.

Tabel 1. Perbandingan Penelitian Terdahulu dan Posisi Penelitian Ini

Peneliti (Ref)	Fokus	Metode/Metrik	Celah yang Masih Terbuka
Jeddou et al. [5]; Jara Ochoa et al. [7]	Performa MQTT pada IoT (konsumsi daya, latensi)	Pengukuran latensi/daya pada jaringan IoT umum	Tidak dikaji dalam konteks notifikasi IDS
Bestari & Wibowo [9]; Andriani & Sa'di [10]	Telegram Bot sebagai media notifikasi monitoring	Implementasi dan uji fungsional	Tidak dibandingkan langsung dengan protokol lain secara kuantitatif
Sokibi [11]; Ardani et al. [12]	Telegram Bot untuk notifikasi monitoring jaringan (ICMP/SNMP)	Studi implementasi, deskriptif	Tidak mengukur latensi secara sistematis maupun membandingkan dengan MQTT
Penelitian ini	MQTT vs Telegram Bot pada notifikasi IDS berbasis Snort	Eksperimen kuantitatif komparatif; 3 skenario serangan; statistik deskriptif lengkap (mean, median, min, maks, SD) & Alert Delivery Rate	Perbandingan langsung dua protokol pada IDS yang sama; ukuran sebaran data dan keandalan pengiriman disajikan eksplisit

Berdasarkan Tabel 1, penelitian terkait MQTT umumnya membahas performa latensi dan efisiensi pada lingkungan IoT secara umum [5][7], sedangkan penelitian terkait Telegram Bot lebih menekankan kemudahan implementasi sebagai media notifikasi tanpa perbandingan kuantitatif terhadap protokol lain

[9][10][11][12]. Penelitian yang membandingkan kedua protokol secara langsung dalam satu rangkaian eksperimen IDS berbasis Snort, dengan skenario serangan yang identik dan pengukuran statistik deskriptif yang menyeluruh, masih sangat terbatas. Celah inilah yang menjadi dasar penelitian ini.

b. Kontribusi dan Kebaruan Penelitian

Kebaruan (novelty) penelitian ini terletak pada tiga aspek. Pertama, pengujian dilakukan secara komparatif-langsung terhadap MQTT dan Telegram Bot pada satu sistem IDS berbasis Snort yang identik, mencakup tiga jenis serangan dengan karakteristik volume alert yang berbeda (rendah pada ICMP, tinggi pada Nmap SYN Scan, dan sangat tinggi pada SYN Flood). Kedua, evaluasi kinerja tidak hanya menggunakan nilai rata-rata latensi, tetapi juga median, nilai minimum, nilai maksimum, dan standar deviasi, sehingga konsistensi dan variabilitas kinerja masing-masing protokol dapat digambarkan secara lebih menyeluruh, bukan hanya kecepatan rata-ratanya. Ketiga, penelitian ini mengukur Alert Delivery Rate sebagai indikator keandalan pengiriman, yang menunjukkan bahwa kecepatan dan keandalan pengiriman perlu dipertimbangkan bersamaan, bukan secara terpisah, dalam memilih protokol notifikasi untuk sistem keamanan jaringan real-time.

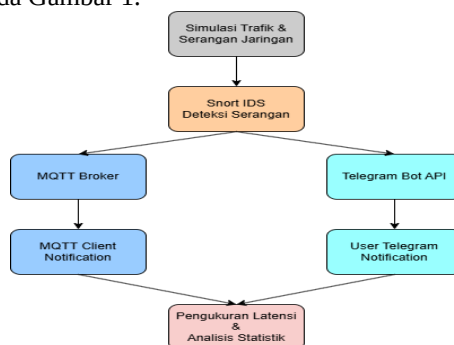
Penelitian ini bertujuan untuk menganalisis dan membandingkan latensi pengiriman notifikasi antara protokol MQTT dan Telegram Bot pada sistem Intrusion Detection System berbasis Snort. Variabel independen dalam penelitian ini adalah jenis protokol komunikasi yang digunakan, yaitu MQTT dan Telegram Bot, sedangkan variabel dependen adalah latensi pengiriman notifikasi yang diukur berdasarkan selisih waktu antara deteksi serangan oleh IDS dan diterimanya notifikasi oleh pengguna. Berdasarkan karakteristik MQTT sebagai protokol ringan berbasis publish-subscribe dengan koneksi persisten [5][6], penelitian ini menduga bahwa MQTT akan menunjukkan latensi rata-rata yang lebih rendah dan lebih konsisten dibandingkan Telegram Bot yang bergantung pada pemrosesan permintaan berurutan melalui layanan cloud pihak ketiga. Karena analisis yang digunakan bersifat deskriptif komparatif tanpa uji inferensial, dugaan ini diuji dengan membandingkan besaran selisih nilai deskriptif (rata-rata, median, dan standar deviasi) antar protokol, bukan dengan pengujian signifikansi statistik formal. Hasil penelitian ini diharapkan dapat menjadi referensi bagi pengembang sistem dalam memilih protokol komunikasi yang optimal untuk mendukung respons keamanan jaringan secara real-time, dengan batasan penelitian pada penggunaan Snort sebagai IDS, implementasi MQTT dan Telegram Bot sebagai media notifikasi, serta fokus pada parameter latensi dan keandalan pengiriman.

2. METODOLOGI PENELITIAN

Penelitian ini menggunakan pendekatan experimental research dengan metode kuantitatif komparatif untuk mengetahui perbedaan performa MQTT dan Telegram Bot dalam kondisi pengujian yang sama. Metodologi penelitian disusun secara sistematis mulai dari tahap perancangan lingkungan simulasi, konfigurasi IDS, implementasi sistem notifikasi, pengujian serangan jaringan, hingga analisis hasil pengukuran latensi.

2.1 Kerangka dan Lingkungan Penelitian

Penelitian dilakukan menggunakan lingkungan simulasi berbasis virtual machine pada satu perangkat komputer pribadi peneliti, yang terdiri atas beberapa node virtual dengan fungsi berbeda, yaitu attacker, server IDS, MQTT broker, dan client penerima notifikasi. Pendekatan simulasi dipilih agar proses pengujian dapat dilakukan secara terkontrol tanpa memerlukan infrastruktur laboratorium jaringan fisik. Alur penelitian ditunjukkan pada Gambar 1.



Gambar 1. Blok Diagram Penelitian

Gambar 1 menunjukkan bahwa proses penelitian diawali dengan simulasi trafik dan serangan jaringan pada lingkungan virtual, kemudian dipantau oleh Snort IDS untuk mendeteksi pola serangan berdasarkan rule yang telah dikonfigurasi. Ketika serangan berhasil terdeteksi, sistem mengirimkan notifikasi menggunakan dua mekanisme berbeda, yaitu melalui protokol MQTT dan Telegram Bot. Pada skenario MQTT, alert dari Snort diteruskan menuju MQTT broker sebelum diterima oleh MQTT subscriber, sedangkan pada skenario Telegram, notifikasi dikirimkan melalui Telegram Bot API menuju akun pengguna Telegram. Tahap selanjutnya adalah pengukuran latensi dengan menghitung selisih waktu antara saat serangan terdeteksi oleh Snort dan saat notifikasi diterima oleh pengguna, kemudian data dianalisis menggunakan statistik deskriptif komparatif.

Tabel 2. Perangkat Keras Penelitian

No	Komponen	Spesifikasi
1	Perangkat	ASUS ROG G752VM
2	Processor	Intel Core i7-6700HQ
3	RAM	16 GB DDR4
4	GPU	NVIDIA GeForce GTX 970M (3GB GDDR5)
5	Penyimpanan	1 TB NVMe SSD + 1 TB HDD
6	Sistem Operasi	Windows 10 Home 64-bit
7	Virtualisasi	VMware Workstation 17 Pro

Tabel 3. Perangkat Lunak Penelitian

No	Perangkat Lunak	Versi	Fungsi
1	Kali Linux	2026.2 (Rolling Release)	Sistem operasi pada VM IDS Server dan Attacker
2	VMware Workstation	17.6.2	Platform virtualisasi lingkungan pengujian
3	Snort	3.12.2.0	IDS untuk mendeteksi serangan jaringan
4	Eclipse Mosquitto	2.1.2	Broker MQTT untuk meneruskan alert Snort
5	Python	3.13.12	Integrasi sistem, pencatatan log, notifikasi
6	Telegram Bot API	Cloud Service	Layanan API pengiriman notifikasi Telegram

Untuk memastikan akurasi pengukuran latensi, seluruh mesin virtual disinkronkan menggunakan Network Time Protocol (NTP), sehingga timestamp yang dihasilkan oleh sistem IDS, MQTT broker, dan client notifikasi mengacu pada referensi waktu yang sama. Langkah ini merupakan bagian dari prosedur validasi hasil pengujian, karena tanpa sinkronisasi waktu, selisih latensi antar-mesin virtual dapat mengandung bias yang tidak berkaitan dengan performa protokol itu sendiri.

2.2 Perancangan Sistem

Snort digunakan sebagai IDS utama untuk mendeteksi aktivitas serangan jaringan dan menghasilkan alert secara real-time. Alert yang dihasilkan Snort disimpan pada berkas `snort_alert.log`, kemudian diproses oleh program Python untuk diteruskan menuju dua media notifikasi berbeda, yaitu MQTT Broker menggunakan metode `publish-subscribe`, dan Telegram Bot menggunakan Telegram API dengan metode `sendMessage` melalui protokol HTTPS. Pada implementasi MQTT, sistem menggunakan broker Mosquitto sebagai perantara komunikasi antara publisher dan subscriber, sejalan dengan praktik pengamanan jalur komunikasi MQTT pada sistem bermuatan sensitif [13]; pada Telegram Bot, sistem memanfaatkan layanan cloud Telegram untuk mengirimkan pesan notifikasi kepada pengguna.

2.3 Skenario Pengujian dan Prosedur Validasi Data

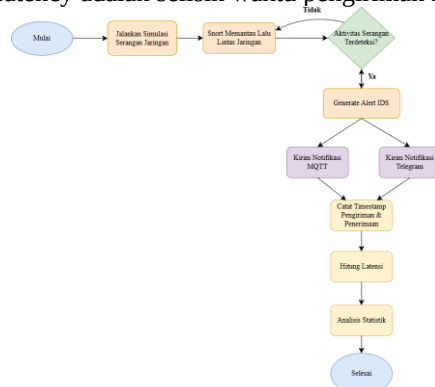
Pengujian dilakukan dengan menjalankan tiga jenis serangan jaringan terhadap server target pada lingkungan simulasi: ICMP menggunakan utilitas `ping`, Port Scanning menggunakan `Nmap`, dan TCP SYN Flood menggunakan `hping3`. Snort dikonfigurasi menggunakan pendekatan `signature-based` sesuai karakteristik masing-masing serangan [14], termasuk analisis hingga tingkat `payload` untuk mengenali pola serangan volumetrik seperti SYN Flood [15]. Setiap skenario dijalankan dalam satu sesi pengujian

berkelanjutan yang menghasilkan alert secara kontinu sesuai karakteristik serangannya; karena volume paket yang dikirim tinggi (khususnya pada Nmap SYN Scan dan SYN Flood), satu sesi pengujian tetap menghasilkan populasi data yang besar untuk setiap skenario, yaitu 2.953 alert pada Nmap SYN Scan dan 99.804 alert pada SYN Flood yang dipublikasikan MQTT. Pendekatan satu sesi per skenario dengan volume alert yang besar merupakan keterbatasan sekaligus kekuatan desain: keterbatasannya adalah tidak adanya replikasi sesi pada kondisi jaringan atau waktu yang berbeda (dibahas pada bagian keterbatasan), sedangkan kekuatannya adalah jumlah observasi latensi per skenario yang tetap besar secara statistik.

Saat serangan terdeteksi oleh Snort, sistem secara otomatis mengirimkan notifikasi melalui MQTT dan Telegram Bot secara terpisah. Pengukuran dilakukan dengan mencatat waktu deteksi alert oleh Snort, waktu penerimaan notifikasi MQTT, dan waktu penerimaan notifikasi Telegram. Latensi didefinisikan sebagai selisih waktu antara saat alert berhasil dihasilkan oleh Snort IDS dan saat notifikasi diterima oleh pengguna, sebagaimana dirumuskan pada Persamaan (1).

$$Latency = Treceive - Tdetect \quad (1)$$

Dengan Treceive adalah waktu notifikasi diterima oleh pengguna, Tdetect adalah waktu alert terdeteksi oleh Snort IDS, dan Latency adalah selisih waktu pengiriman notifikasi dalam satuan detik.



Gambar 2. Flowchart Sistem Pengujian

2.4 Teknik Pengumpulan dan Analisis Data

Seluruh data hasil pengujian dicatat secara otomatis menggunakan program Python yang dikembangkan pada penelitian ini. Data penerimaan notifikasi MQTT disimpan ke dalam berkas mqtt_log.csv, sedangkan data penerimaan notifikasi Telegram disimpan ke dalam berkas telegram_log.csv. Sebagai prosedur validasi, kedua berkas tersebut dipadankan (merge) berdasarkan identitas alert yang sama sebelum dihitung nilai latensinya; hanya pasangan data yang berhasil dicocokkan pada kedua sisi yang dimasukkan ke dalam analisis latensi komparatif. Oleh karena itu, jumlah data yang dianalisis pada statistik latensi (n) mengikuti jumlah pasangan yang berhasil dicocokkan, yaitu 83 pasang pada ICMP, 66 pasang pada Nmap SYN Scan, dan 71 pasang pada SYN Flood — jumlah ini lebih kecil dibandingkan total alert yang dipublikasikan MQTT pada skenario volume tinggi, karena Alert Delivery Rate Telegram Bot yang lebih rendah membatasi jumlah pasangan data yang lengkap (lihat Subbab 3.4).

Analisis data dilakukan menggunakan metode statistik deskriptif kuantitatif, meliputi nilai rata-rata (mean), median, minimum, maksimum, dan standar deviasi, untuk membandingkan performa latensi antara MQTT dan Telegram Bot. Selain kecepatan pengiriman, penelitian ini juga menghitung Alert Delivery Rate, yaitu persentase alert yang berhasil diterima dibandingkan jumlah alert yang dipublikasikan, sebagai indikator keandalan pengiriman. Penelitian ini secara eksplisit membatasi diri pada analisis deskriptif dan tidak melakukan uji statistik inferensial (misalnya independent t-test atau Mann–Whitney U Test); implikasi dan rekomendasi terkait keterbatasan ini dibahas pada Subbab 3.5.

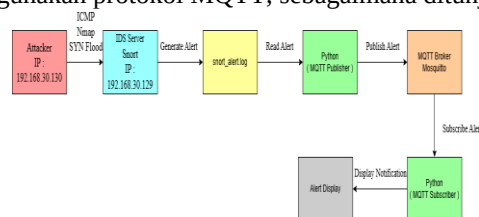
3. HASIL DAN PEMBAHASAN

3.1 Implementasi Sistem

Implementasi sistem pada penelitian ini dilakukan untuk menguji mekanisme pengiriman notifikasi dari Snort IDS menggunakan dua protokol komunikasi yang berbeda, yaitu MQTT dan Telegram Bot. Implementasi dimulai ketika Snort IDS mendeteksi aktivitas serangan berdasarkan rule yang telah dikonfigurasi, kemudian menghasilkan alert yang selanjutnya dikirimkan melalui masing-masing protokol sesuai mekanisme yang diterapkan.

a. Implementasi Pengiriman Alert Menggunakan MQTT

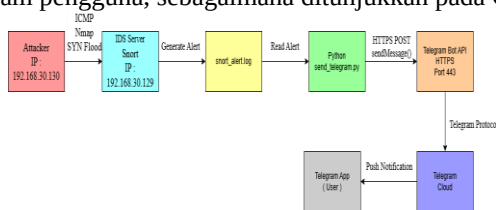
Implementasi pengiriman notifikasi menggunakan protokol MQTT dilakukan dengan memanfaatkan arsitektur publish-subscribe yang disediakan oleh Mosquitto Broker. Mekanisme ini dimulai ketika perangkat attacker mengirimkan lalu lintas jaringan berupa serangan ICMP, Nmap SYN Scan, maupun SYN Flood menuju IDS Server yang menjalankan Snort IDS. Setelah Snort IDS mendeteksi aktivitas yang sesuai dengan rule deteksi, sistem menghasilkan alert yang disimpan ke dalam berkas snort_alert.log. Selanjutnya, program Python yang berperan sebagai MQTT Publisher membaca setiap alert baru dari berkas log tersebut dan mengirimkan informasi alert ke Mosquitto Broker menggunakan topik snort/alert. Mosquitto Broker kemudian mendistribusikan pesan tersebut kepada MQTT Subscriber yang telah melakukan subscribe pada topik yang sama, sehingga informasi notifikasi ditampilkan sebagai hasil akhir proses pengiriman alert menggunakan protokol MQTT, sebagaimana ditunjukkan pada Gambar 3.



Gambar 3. Implementasi Pengiriman Alert Menggunakan MQTT

b. Implementasi Pengiriman Alert Menggunakan Telegram Bot

Implementasi pengiriman alert menggunakan Telegram Bot dilakukan dengan memanfaatkan layanan Telegram Bot API sebagai media komunikasi antara sistem IDS dan aplikasi Telegram pada perangkat pengguna. Setelah Snort IDS menghasilkan alert dan menyimpannya ke berkas snort_alert.log, program Python yang berperan sebagai Telegram Sender membaca setiap alert baru dan mengirimkan pesan notifikasi ke Telegram Bot API menggunakan metode sendMessage melalui protokol HTTPS pada port 443. Telegram Bot API kemudian meneruskan pesan tersebut ke Telegram Cloud sehingga notifikasi dapat dikirimkan ke aplikasi Telegram pengguna, sebagaimana ditunjukkan pada Gambar 4.



Gambar 4. Implementasi Pengiriman Alert Menggunakan Telegram Bot

3.2 Hasil Pengujian per Skenario Serangan

Pengujian dilakukan pada tiga skenario serangan. Tabel 4 menunjukkan contoh data pengiriman alert ICMP oleh kedua protokol; sampel data untuk skenario Nmap SYN Scan dan SYN Flood memiliki struktur serupa dan tidak ditampilkan seluruhnya untuk efisiensi ruang.

Tabel 4. Sampel Data Pengiriman Alert pada Pengujian ICMP

No	Waktu Alert Snort	Jenis Alert	Waktu Diterima MQTT	Waktu Diterima Telegram
1	09:35:21.093185	ICMP DETECTED	09:36:40.852957	09:36:40.854189
2	09:35:21.093205	ICMP DETECTED	09:36:40.855818	09:36:42.130528
3	09:35:22.093829	ICMP DETECTED	09:36:40.858941	09:36:43.416549
4	09:35:22.093854	ICMP DETECTED	09:36:40.860091	09:36:44.646031
5	09:35:23.102975	ICMP DETECTED	09:36:40.861620	09:36:45.860819

Seluruh alert pada Tabel 4 berasal dari komunikasi ICMP antara IP sumber 192.168.30.130 dan IP tujuan 192.168.30.129. Pada pengujian ICMP, MQTT dan Telegram Bot sama-sama berhasil menerima seluruh 83 alert yang cocok (matched), dengan rata-rata latensi 90,38 detik untuk MQTT dan 118,64 detik untuk Telegram Bot. Nilai latensi yang relatif tinggi pada kedua protokol ini terutama dipengaruhi oleh keterlambatan pembangkitan alert oleh rule ICMP pada Snort IDS selama pengujian, bukan oleh performa

protokol komunikasi itu sendiri — sehingga skenario ICMP kurang representatif untuk menilai kecepatan murni kedua protokol dan lebih tepat dibaca sebagai kondisi kontrol.

3.3 Analisis Kecepatan dan Konsistensi Pengiriman Alert

Ringkasan hasil pengujian ketiga skenario ditunjukkan pada Tabel 5, sedangkan statistik deskriptif lengkap (mean, median, minimum, maksimum, standar deviasi) dari data latensi yang telah dipadankan ditunjukkan pada Tabel 6.

Tabel 5. Ringkasan Alert dan Rata-rata Latensi MQTT dan Telegram Bot

Jenis Serangan	Alert MQTT (dipublikasikan)	Alert Telegram (diterima)	Rata-rata MQTT (s)	Rata-rata Telegram (s)	Selisih (s)
ICMP	83	83	90,38	118,64	28,26
Nmap SYN Scan	2.953	66	0,15	41,65	41,51
SYN Flood	99.804	71	0,38	47,31	46,93

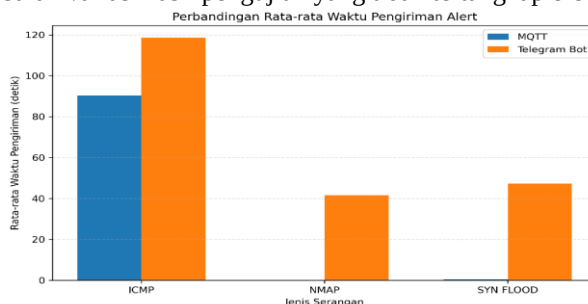
Tabel 6. Statistik Deskriptif Latensi (Mean, Median, Minimum, Maksimum, Standar Deviasi) MQTT dan Telegram Bot pada Data yang Telah Dipadankan

Jenis Serangan (n)	Protokol	Mean (s)	Median (s)	Min (s)	Maks (s)	Std. Deviasi (s)
ICMP (n=83)	MQTT	90,38	77,75	0,608	161,46	32,021
ICMP (n=83)	Telegram	118,64	115,80	58,275	167,26	24,426
Nmap SYN Scan (n=66)	MQTT	0,15	0,15	0,055	0,21	0,044
Nmap SYN Scan (n=66)	Telegram	41,65	41,60	0,064	83,20	24,577
SYN Flood (n=71)	MQTT	0,38	0,39	0,076	0,56	0,148
SYN Flood (n=71)	Telegram	47,31	47,35	0,175	94,95	27,551

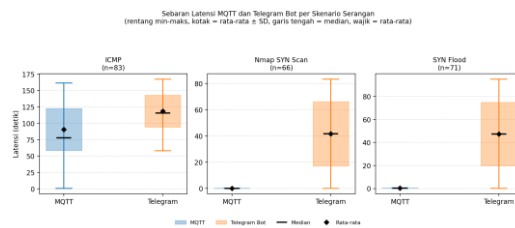
Berdasarkan Tabel 5 dan Tabel 6, MQTT menunjukkan waktu pengiriman alert yang lebih rendah dibandingkan Telegram Bot pada seluruh skenario pengujian. Pada serangan Nmap SYN Scan, MQTT mencatat rata-rata 0,15 detik dengan standar deviasi hanya 0,044 detik (rentang 0,055–0,21 detik), sedangkan Telegram Bot mencatat rata-rata 41,65 detik dengan standar deviasi 24,577 detik (rentang 0,064–83,20 detik). Pola serupa terjadi pada SYN Flood: MQTT mencatat rata-rata 0,38 detik dengan standar deviasi 0,148 detik, sedangkan Telegram Bot mencatat rata-rata 47,31 detik dengan standar deviasi 27,551 detik dan nilai maksimum mencapai 94,95 detik.

Nilai standar deviasi yang jauh lebih kecil pada MQTT menunjukkan bahwa waktu pengirimannya tidak hanya lebih cepat secara rata-rata, tetapi juga jauh lebih konsisten dan dapat diprediksi; sebaliknya, rentang nilai Telegram Bot yang lebar (misalnya 0,064–83,20 detik pada Nmap SYN Scan) menunjukkan bahwa performanya sangat bervariasi antar-alert, sehingga kecepatan rata-rata saja tidak cukup untuk menilai kelayakan protokol pada sistem notifikasi real-time. Pada skenario ICMP, nilai median MQTT (77,75 detik) yang lebih rendah dari rata-ratanya (90,38 detik) mengindikasikan adanya sejumlah alert dengan latensi tinggi (outlier) yang menarik nilai rata-rata ke atas, sehingga median lebih representatif untuk menggambarkan kecepatan tipikal pada skenario ini.

Perbandingan rata-rata waktu pengiriman alert ditunjukkan pada Gambar 5, sedangkan sebaran datanya (rentang minimum–maksimum, median, dan rata-rata $\pm$ standar deviasi) ditunjukkan pada Gambar 6 untuk melengkapi gambaran variasi hasil pengujian yang tidak tertangkap oleh nilai rata-rata saja.



Gambar 5. Perbandingan Rata-rata Waktu Pengiriman Alert MQTT dan Telegram Bot



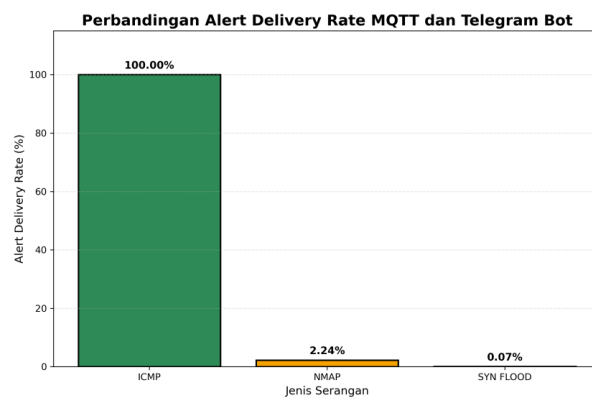
Gambar 6. Sebaran Latensi MQTT dan Telegram Bot per Skenario Serangan (Rentang Min-Maks, Median, dan Rata-rata $\pm$ Standar Deviasi)

3.4 Analisis Alert Delivery Rate

Selain kecepatan pengiriman, penelitian ini mengevaluasi kemampuan masing-masing protokol dalam menyampaikan seluruh alert yang dihasilkan Snort IDS melalui parameter Alert Delivery Rate, yaitu persentase alert yang berhasil diterima terhadap jumlah alert yang dipublikasikan. Hasil pengujian ditunjukkan pada Tabel 7 dan Gambar 7.

Tabel 7. Alert Delivery Rate MQTT dan Telegram Bot

Serangan	MQTT Alert	Telegram Alert	Alert Delivery Rate (%)
ICMP	83	83	100,0
Nmap SYN Scan	2.953	66	2,24
SYN Flood	99.804	71	0,07



Gambar 7. Perbandingan Alert Delivery Rate MQTT dan Telegram Bot

MQTT berhasil meneruskan seluruh alert pada ketiga skenario pengujian tanpa kehilangan data (Alert Delivery Rate 100% relatif terhadap jumlah yang dipublikasikan MQTT sendiri). Telegram Bot hanya menerima 66 dari 2.953 alert (2,24%) pada Nmap SYN Scan dan 71 dari 99.804 alert (0,07%) pada SYN Flood. Alert Delivery Rate Telegram Bot menurun tajam seiring meningkatnya volume alert yang dikirim dalam waktu singkat, menunjukkan bahwa keandalan Telegram Bot berbanding terbalik dengan intensitas trafik notifikasi.

3.5 Pembahasan

Secara keseluruhan, MQTT menunjukkan kinerja yang lebih baik dibandingkan Telegram Bot baik dari aspek kecepatan, konsistensi (standar deviasi), maupun keandalan pengiriman (Alert Delivery Rate). Temuan ini konsisten dengan karakteristik arsitektur kedua protokol yang telah dilaporkan pada literatur: MQTT menggunakan mekanisme publish-subscribe dengan koneksi persistent yang ringan dan overhead rendah [5][6], sehingga tidak perlu membangun koneksi baru untuk setiap pesan dan mampu mempertahankan performa pada volume pengiriman tinggi, sebagaimana juga ditunjukkan pada implementasi MQTT untuk kondisi trafik padat pada sistem deteksi lain [16]. Sebaliknya, Telegram Bot bergantung pada layanan API berbasis cloud yang memproses setiap pesan secara berurutan melalui server eksternal serta tunduk pada kebijakan rate limit yang berada di luar kendali pengembangan sistem [11][12]; ketika volume alert meningkat drastis dalam waktu singkat, antrean permintaan pada sisi Telegram Bot API menjadi penghambat utama sehingga banyak alert gagal diteruskan ke pengguna. Pola keterbatasan serupa

juga dilaporkan pada berbagai implementasi Telegram Bot untuk notifikasi monitoring jaringan lain, yang secara konsisten mengandalkan frekuensi notifikasi yang relatif rendah agar tetap andal [17][18][19].

Pada pengujian ICMP, kedua protokol menunjukkan waktu pengiriman yang jauh lebih tinggi dibandingkan skenario lain (90,38 detik untuk MQTT dan 118,64 detik untuk Telegram Bot). Nilai ini lebih banyak dipengaruhi oleh keterlambatan pembangkitan alert oleh rule ICMP pada Snort IDS selama pengujian berlangsung — sejalan dengan temuan bahwa efektivitas dan konfigurasi rule sangat memengaruhi kecepatan deteksi Snort [20], dan bahwa proses ini dapat ditelusuri lebih lanjut melalui analisis log deteksi secara forensik [21] — bukan oleh performa jalur notifikasi itu sendiri, sehingga selisih waktu antar-protokol pada skenario ini kurang tepat dijadikan dasar tunggal untuk menilai kecepatan murni MQTT maupun Telegram Bot.

Perlu ditegaskan bahwa seluruh perbandingan pada penelitian ini bersifat deskriptif: istilah “lebih baik”, “lebih konsisten”, atau “berbeda besar” merujuk pada selisih nilai rata-rata, median, dan standar deviasi yang teramati pada data hasil pengujian, bukan pada hasil uji signifikansi statistik formal (misalnya independent t-test atau Mann–Whitney U Test), karena pengujian tersebut memang tidak dilakukan pada penelitian ini. Dengan populasi data latensi yang cukup besar ($n=66$ dan $n=71$ pada skenario volume tinggi) serta selisih rata-rata yang jauh melampaui standar deviasi masing-masing kelompok, pola keunggulan MQTT teramati konsisten pada seluruh skenario; namun kesimpulan mengenai signifikansi statistik dari perbedaan tersebut memerlukan pengujian inferensial lebih lanjut, dan hal ini secara eksplisit diusulkan sebagai penelitian lanjutan pada Bab 4.

Dari sisi implikasi praktis, temuan ini memberikan pertimbangan bagi pengembang sistem keamanan jaringan dalam menentukan strategi notifikasi. Pada lingkungan yang menuntut respons keamanan seketika, seperti server produksi dengan trafik tinggi, penggunaan MQTT sebagai jalur notifikasi utama dapat meminimalkan risiko keterlambatan maupun kehilangan alert. Telegram Bot tetap relevan sebagai kanal notifikasi pelengkap bagi administrator yang membutuhkan akses cepat melalui perangkat mobile tanpa menjalankan aplikasi MQTT Subscriber khusus, terutama pada skenario dengan frekuensi alert rendah seperti ICMP atau notifikasi ringkasan berkala.

Penelitian ini memiliki beberapa keterbatasan yang perlu diperhatikan dalam menginterpretasikan hasil. Pertama, pengujian dilakukan pada lingkungan simulasi berbasis virtual machine dalam satu perangkat komputer, sehingga kondisi jaringan yang dihasilkan belum sepenuhnya merepresentasikan kompleksitas jaringan produksi, misalnya variasi latensi akibat jarak geografis maupun beban server yang lebih dinamis. Kedua, performa Telegram Bot turut dipengaruhi oleh kebijakan rate limit dari layanan Telegram Bot API yang bersifat eksternal dan berada di luar kendali peneliti, sehingga hasil pengujian berpotensi berbeda apabila diterapkan pada kondisi jaringan atau periode waktu lain. Ketiga, setiap skenario serangan hanya dijalankan dalam satu sesi pengujian tanpa replikasi pada waktu atau kondisi jaringan yang berbeda; meskipun volume alert per sesi cukup besar, replikasi antar-sesi akan memperkuat generalisasi hasil. Keempat, analisis yang digunakan terbatas pada statistik deskriptif tanpa uji inferensial dan tanpa protokol pembanding lain di luar MQTT dan Telegram Bot, sehingga kekuatan evaluasi masih terbatas pada dua protokol yang diuji. Keterbatasan-keterbatasan ini menjadi dasar rekomendasi penelitian lanjutan pada Bab 4.

4. KESIMPULAN

Penelitian ini memberikan kontribusi berupa perbandingan langsung dan terukur antara protokol MQTT dan Telegram Bot sebagai media notifikasi pada sistem IDS berbasis Snort, yang mencakup tiga skenario serangan dengan karakteristik volume alert berbeda serta evaluasi ganda pada aspek kecepatan (mean, median, standar deviasi) dan keandalan (Alert Delivery Rate) — sebuah kombinasi evaluasi yang belum banyak dijumpai pada penelitian-penelitian sebelumnya yang umumnya mengkaji kedua teknologi secara terpisah.

Snort IDS berhasil mendeteksi seluruh skenario serangan yang diujikan, yaitu ICMP, Nmap SYN Scan, dan SYN Flood, kemudian meneruskan alert melalui protokol MQTT dan Telegram Bot. Pada serangan Nmap SYN Scan dan SYN Flood, MQTT memiliki rata-rata waktu pengiriman masing-masing sebesar 0,15 detik dan 0,38 detik dengan standar deviasi di bawah 0,15 detik, sedangkan Telegram Bot memerlukan rata-rata 41,65 detik dan 47,31 detik dengan standar deviasi di atas 24 detik, menunjukkan bahwa MQTT tidak hanya lebih cepat tetapi juga jauh lebih konsisten. Pada serangan ICMP, rata-rata waktu pengiriman MQTT sebesar 90,38 detik dan Telegram Bot sebesar 118,64 detik, yang lebih banyak dipengaruhi oleh keterlambatan pembangkitan alert pada rule ICMP di Snort IDS selama penelitian dibandingkan performa jalur notifikasi itu sendiri.

Dari sisi Alert Delivery Rate, MQTT mampu menangani seluruh alert yang dipublikasikan selama pengujian (delivery rate 100%), sedangkan Telegram Bot hanya menerima 66 dari 2.953 alert (2,24%) pada serangan Nmap SYN Scan dan 71 dari 99.804 alert (0,07%) pada serangan SYN Flood. Berdasarkan hasil tersebut, MQTT lebih sesuai digunakan sebagai media pengiriman notifikasi pada sistem IDS yang membutuhkan kecepatan dan keandalan tinggi, sedangkan Telegram Bot lebih sesuai digunakan untuk kebutuhan monitoring dengan frekuensi alert yang relatif rendah. Karena penelitian ini menggunakan analisis deskriptif tanpa uji inferensial, kesimpulan di atas menggambarkan pola dan besaran selisih yang teramati pada data, bukan kesimpulan signifikansi statistik yang telah diuji secara formal.

Penelitian selanjutnya disarankan untuk: (1) menerapkan uji statistik inferensial (misalnya independent t-test atau Mann–Whitney U Test) guna menguji signifikansi perbedaan latensi antar-protokol secara formal; (2) mereplikasi pengujian pada beberapa sesi dan kondisi jaringan yang berbeda untuk memperkuat generalisasi hasil; (3) mengembangkan mekanisme penyusunan dan optimasi rule pada Snort IDS agar proses pembangkitan alert berlangsung lebih cepat dan konsisten, khususnya pada skenario ICMP; dan (4) memperluas pengujian dengan membandingkan protokol komunikasi lain, seperti AMQP, CoAP, atau WebSocket, sehingga performa sistem notifikasi IDS dapat dievaluasi secara lebih komprehensif dan tidak terbatas pada dua protokol yang diuji dalam penelitian ini.

UCAPAN TERIMAKASIH

Penulis mengucapkan terima kasih kepada dosen pembimbing atas bimbingan, arahan, dan masukan selama proses penelitian. Penulis juga menyampaikan terima kasih kepada Program Studi Informatika Universitas Mercu Buana Yogyakarta yang telah memberikan dukungan dan fasilitas sehingga penelitian ini dapat diselesaikan dengan baik.

REFERENCES

- [1] D. Viet, T. Nguyen, D. Minh, T. Ngoc, B. Hoang, and D. T. Son, "Cybersecurity Threats and Defense Mechanisms in IoT network," Jan. 2026. [Online]. Available: <https://arxiv.org/pdf/2601.00556>
- [2] S. K. Sahu and K. Mazumdar, "Exploring security threats and solutions techniques for Internet of Things (IoT): from vulnerabilities to vigilance," *Front. Artif. Intell.*, vol. 7, May 2024, doi: 10.3389/frai.2024.1397480.
- [3] N. V. Sharma, Kavita, G. Aggarwal, and S. Sharma, "Performance Study of Snort and Suricata for Intrusion Detection System," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1099, no. 1, p. 012009, Mar. 2021, doi: 10.1088/1757-899X/1099/1/012009.
- [4] H. Maosa, K. Ouazzane, and M. C. Ghanem, "A Hierarchical Security Events Correlation Model for Real-time Cyber Threat Detection and Response," Dec. 2023. [Online]. Available: <https://arxiv.org/pdf/2312.01219>
- [5] S. Jeddou, F. Fernández, L. Diez, A. Baina, N. Abdallah, and R. Agüero, "Delay and Energy Consumption of MQTT over QUIC: An Empirical Characterization Using Commercial-Off-The-Shelf Devices," *Sensors*, vol. 22, no. 10, p. 3694, May 2022, doi: 10.3390/s22103694.
- [6] S. Bagas Bhaskoro, H. Supriyanto, B. B. Aji, B. Pamungkas, and P. M. Bandung, "Perbandingan Performansi Latency Protokol Komunikasi HTTP dan MQTT Pada Internet of Things," *Jurnal Teknologi Terapan*, vol. 8, no. 2, 2022.
- [7] H. J. Jara Ochoa, R. Peña, Y. Ledo Mezquita, E. Gonzalez, and S. Camacho-Leon, "Comparative Analysis of Power Consumption between MQTT and HTTP Protocols in an IoT Platform Designed and Implemented for Remote Real-Time Monitoring of Long-Term Cold Chain Transport Operations," *Sensors*, vol. 23, no. 10, p. 4896, May 2023, doi: 10.3390/s23104896.
- [8] L. Nurfiqin, Z. Sari, F. Dwi, and S. Sumadi, "Analisis Quality of Service (QoS) Protokol MQTT dan HTTP Pada Sistem Smart Metering Arus Listrik," *Jurnal Repositor*, vol. 3, no. 1, pp. 121–130, Jan. 2021, doi: 10.22219/REPOSITOR.V3I1.31040.
- [9] D. N. Bestari and A. Wibowo, "IoT Based Real-Time Weather Monitoring System Using Telegram Bot and Thingsboard Platform," *International Journal of Interactive Mobile Technologies (ijim)*, vol. 17, no. 06, pp. 4–19, Mar. 2023, doi: 10.3991/ijim.v17i06.34129.
- [10] R. Andriani and A. Sa'di, "Implementation of Telegram Bot Notification on Network Device Monitoring System," *SISTEMASI*, vol. 13, no. 1, p. 74, Jan. 2024, doi: 10.32520/stmsi.v13i1.3110.
- [11] P. Sokibi, "Perancangan Sistem Monitoring Perangkat Jaringan Berbasis ICMP dengan Notifikasi Telegram," *ITEJ (Information Technology Engineering Journals)*, 2017.

- [12] A. H. Ardani, R. Azhar, and O. Asroni, "Sistem Monitoring Multi Perangkat Jaringan Berbasis Simple Network Management Protocol (SNMP) dengan Notifikasi Telegram," *RIGGS: Journal of Artificial Intelligence and Digital Business*, vol. 5, no. 1, pp. 7355–7365, Mar. 2026, doi: 10.31004/riggs.v5i1.7125.
- [13] D. Rachmadini, I. Puspasari, and Jusak, "Implementasi dan Analisis Fitur Keamanan Protokol MQTT pada Telehealthcare," *Journal of Technology and Informatics (JoTI)*, vol. 2, no. 1, pp. 10–19, Oct. 2020, doi: 10.37802/JOTI.V2I2.117.
- [14] F. A. Febian, D. Djumhadi, and W. N. Alimyaningtias, "Analisis Perbandingan Teknik Signature-Based dan Anomaly-Based Detection Pada Snort dan Zeek Dalam Mencegah Intrusi Jaringan," *Jaits: Journal of Applied Information Technology Solution*, vol. 1, no. 2, pp. 8–12, Aug. 2024. [Online]. Available: <https://journal.universitasmulia.ac.id/index.php/jaits/article/view/654>
- [15] A. R. Syujak, K. Diantoro, V. Yuni T, A. Soderi, and P. A. Sucipto, "Integrasi Deep Packet Inspection dengan Intrusion Detection System (IDS) untuk Identifikasi Serangan DDoS dalam Jaringan Skala Besar," *Jurnal Minfo Polgan*, vol. 13, no. 2, pp. 1971–1975, Dec. 2024, doi: 10.33395/jmp.v13i2.14324.
- [16] A. Hanif and R. Amri, "Implementasi Internet of Things Pada Protokol MQTT dan HTTP Dalam Sistem Pendeteksi Banjir," vol. 8, no. 2, 2023.
- [17] M. Mukmin, P. Purnawansyah, and M. Hasnawi, "Implementasi Bot Telegram Untuk Monitoring Jaringan Dengan Pendekatan Security Policy Development Life Cycle Pada Kementerian Kelautan dan Perikanan Untia," *Buletin Sistem Informasi dan Teknologi Islam (BUSITI)*, vol. 3, no. 2, pp. 127–133, May 2022, doi: 10.33096/BUSITI.V3I2.1162.
- [18] Noor Arief Kurniawan, Diyah Ruswanti, and Astri Charolina, "Implementasi Telegram Bot sebagai Media Monitoring Gangguan Pelanggan Telkom dengan Pembaruan Berkala Berbasis Mobile," *INSOLOGI: Jurnal Sains dan Teknologi*, vol. 4, no. 5, pp. 1226–1240, Oct. 2025, doi: 10.55123/insologi.v4i5.6234.
- [19] W. Adhiwibowo, F. W. Christanto, and A. F. Daru, "Implementasi API Bot Telegram Untuk Sistem Notifikasi Pada The Dude Network Monitoring System," *Seminar Nasional Penelitian dan Pengabdian Kepada Masyarakat 2025*, pp. 593–599, Nov. 2021. [Online]. Available: <https://prosiding.uhb.ac.id/index.php/SNPPKM/article/view/894>
- [20] Suci Sekar Sari and Agus Tedyana, "Analisis Efektivitas Rule Snort dalam Mendeteksi Serangan Jaringan," *Repeater: Publikasi Teknik Informatika dan Jaringan*, vol. 2, no. 4, pp. 01–15, Aug. 2024, doi: 10.62951/repeater.v2i4.194.
- [21] E. Kusuma Dewi, P. Kasih, and N. PGRI Kediri, "Analisis Log Snort Menggunakan Network Forensic." [Online]. Available: <https://www.snort.org/>