

DESAIN KEAMANAN PADA KOMUNIKASI DATA SISTEM BANK AIR KAMI V2 MENGGUNAKAN TRIAS CIA



Disusun Oleh:

N a m a : Farkhan Nindyarayhan Dhanendra

NIM : 20523041

**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA**

2024

HALAMAN PENGESAHAN DOSEN PEMBIMBING

**DESAIN KEAMANAN PADA KOMUNIKASI DATA SISTEM
BANK AIR KAMI V2 MENGGUNAKAN TRIAS CIA**

TUGAS AKHIR



N a m a : Farkhan Nindyarayhan Dhanendra

NIM : 20523041



Yogyakarta, 10 Juni 2024

Pembimbing,



(Ari Sujarwo, S.Kom., MIT (Hons))

HALAMAN PENGESAHAN DOSEN PENGUJI

DESAIN KEAMANAN PADA KOMUNIKASI DATA SISTEM BANK AIR KAMI V2 MENGGUNAKAN TRIAS CIA

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 26 Juli 2024

Tim Penguji

Ari Sujarwo, S.Kom., MIT (Hons)

Anggota 1

Nur Wijayaning Rahayu, S.Kom., M.Cs.

Anggota 2

Zainudin Zukhri, S.T., M.IT.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Farkhan Nindyarayhan Dhanendra

NIM : 20523041

Tugas akhir dengan judul:

DESAIN KEAMANAN PADA KOMUNIKASI DATA SISTEM BANK AIR KAMI V2 MENGGUNAKAN TRIAS CIA

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 10 Juni 2024



(Farkhan Nindyarayhan Dhanendra)

HALAMAN PERSEMBAHAN

Alhamdulillah Rabbil 'Alamin. Puji syukur dipanjatkan kepada Allah Subhana Wa Ta'ala yang telah memberikan karunia, ridho, dan rahmat-Nya sehingga peneliti dapat menyelesaikan tugas akhir sebagai syarat. Shalawat serta salam kepada Nabi Muhammad Shalallahu 'Alaihi Wa Sallam, sebagai suri tauladan pembawa risalah kebenaran yang menyempurnakan agama Allah Subhana Wa Ta'ala.

Karya ini kupersembahkan untuk orang tua saya tercinta. Kepada Ibu Noviani tersayang yang tidak pernah letih memberikan kasih sayang dan perhatiannya untuk tumbuh besar menjadi orang yang bermanfaat dan kuat. Kepada Ayah Deni tersayang yang tidak pernah berhenti memberikan kasih sayang, memberikan pedoman hidup untuk selalu menjadi yang terbaik dan terkuat, memberikan motivasi untuk menjadi pelopor kebaikan dan penakluk keburukan, dan tidak pernah lupa mengingatkan untuk menjalankan perintah-Nya.

Kepada teman-teman dan sahabat-sahabatku yang selalu kebersamai, memberikan dukungan, pengalaman, motivasi, dan doa yang telah diberikan.

HALAMAN MOTO

“Kami wasiatkan kepada manusia agar berbuat baik kepada kedua orang tuanya. Ibunya telah mengandungnya dengan susah payah dan melahirkannya dengan susah payah. Mengandung sampai menyapihnya itu selama tiga puluh bulan. Sehingga, apabila telah dewasa dan umurnya mencapai empat puluh tahun, dia (anak itu) berkata, “Wahai Tuhanku, berilah petunjuk agar aku dapat mensyukuri nikmat-Mu yang telah Engkau anugerahkan kepadaku dan kepada kedua orang tuaku, dapat beramal saleh yang Engkau ridai, dan berikanlah kesalehan kepadaku hingga kepada anak cucuku. Sesungguhnya aku bertobat kepada-Mu dan sesungguhnya aku termasuk orang-orang muslim.”

(Q.S Al-Ahqaf: 15)

“Karena itu, ingatlah Aku niscaya Aku ingat pula kepadamu, dan bersyukurlah kepada-Ku, dan janganlah kamu mengingkari nikmat-Ku.”

(Q.S Al-Baqarah: 151)

“Sebaik-baiknya manusia adalah yang paling bermanfaat untuk orang lain.”

(HR. Ahmad)

“Terbentur, terbentur, terbentur, terbentuk!”

(Tan Malaka)

“Kreativitas itu menular, sebarkanlah!”

(Albert Einstein)

KATA PENGANTAR

Alhamdulillah Robbil ‘Alamin, puji syukur atas kehadiran Allah Subhana Wa Ta’ala yang telah memberikan rahmat, hidayah, ridho, dan karunia-Nya sehingga tugas akhir yang berjudul “DESAIN KEAMANAN PADA KOMUNIKASI DATA SISTEM BANK AIR KAMI V2 MENGGUNAKAN TRIAS CIA” berhasil diselesaikan dengan harapan dapat memberikan manfaat pada orang-orang yang membutuhkannya. Shalawat serta salam senantiasa tercurah limpahkan kepada baginda Nabi Muhammad Shalallahu ‘Alaihi Wa Sallam yang telah membawa perdamaian dan penyempurnaan atas agama Allah Subhana Wa Ta’ala, sehingga membawa kita pada zaman yang terang benderang ini.

Laporan tugas akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata-1 (S-1) Jurusan Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia sebagai sarana untuk menerapkan ilmu dan teori yang telah diperoleh selama menjalani studi di Jurusan Informatika.

Izinkan penulis untuk mengutarakan rasa terima kasih dan penghargaan yang setinggi-tingginya atas segala bantuan, motivasi, bimbingan, dan doa. Penulis menyampaikan rasa dan penghargaan tersebut kepada:

1. Orang tua yang selalu memberikan dukungan dalam hal material, spiritual, dan psikologis sehingga penulis dapat menyelesaikan tugas akhir ini.
2. Keluarga dan saudara-saudara yang selalu memberikan dukungan dan doa.
3. Bapak Fathul Wahid, S.T., M.Sc., Ph.D., sebagai Rektor Universitas Islam Indonesia dan seluruh jajaran Rektorat Universitas Islam Indonesia.
4. Bapak Prof. Dr. Ir. Hari Purnomo, M.T., sebagai Dekan Fakultas Teknologi Industri Universitas Islam Indonesia dan seluruh jajaran Dekanat Fakultas Teknologi Industri Universitas Islam Indonesia.
5. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., sebagai Ketua Jurusan Informatika Universitas Islam Indonesia.
6. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., sebagai Ketua Program Studi S1 Informatika.
7. Bapak Ari Sujarwo, S.Kom., MIT (Hons) sebagai Dosen Pembimbing yang telah banyak memberikan edukasi, arahan, bimbingan, motivasi, dan masukan dalam menyelesaikan tugas akhir.
8. Dosen-dosen Jurusan Informatika beserta staf yang telah memberikan ilmu pengetahuan dan bantuannya.

9. Tim penelitian Bank Air Kami v2, Kevin Fernando Atalaric dan Wilwatika Tripawangsa yang telah bekerja sama dalam mengembangkan proyek tugas akhir.
10. Sahabat-sahabat Bimbingan Pak Ari Sujarwo, Unggul Budi Astowo dan Farhan Novan Fauzi yang selalu memberikan dukungan, ilmu pengetahuan, dan motivasi untuk penulis menyelesaikan tugas akhir.
11. Teman-temanku, Sechan, Ferdian, Faiz, Ayie, Nabil, Wizqi, Hakim, dan Marshal, terima kasih atas dukungan, ilmu, pengalaman, dan motivasi yang telah diberikan.
12. Dingo Coffee UII Jakal yang telah memberikan sarana untuk penulis dan pelayanan terbaik sehingga penulis dapat menyelesaikan tugas akhir.
13. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang telah memberikan bantuan atas terciptanya karya ini.

Penulis menyadari bahwa masih terdapat kekurangan dalam penulisan dan penelitian tugas akhir ini, segala kritik dan saran akan diterima dengan senang hati dan menjadi pembelajaran untuk penulis. Terakhir, dengan harapan yang luas laporan tugas akhir ini dapat bermanfaat untuk penelitian Bank Air Kami selanjutnya dan bagi pihak yang membutuhkan.

Yogyakarta, 29 Mei 2024

(Farkhan Nindyarayhan Dhanendra)

SARI

Memastikan proliferasi generasi unggul dalam konteks pembangunan bangsa merupakan kewajiban bagi seluruh masyarakat untuk ikut berkontribusi. Salah satu asupan terpenting dalam kehidupan manusia dan semua makhluk hidup adalah air. Konsumsi air yang layak dan berkualitas untuk kesehatan adalah sesuatu yang harus terpenuhi untuk semua orang, sehingga upaya untuk memastikan kualitas air yang dikonsumsi oleh masyarakat perlu dilakukan. Hal ini yang menjadi salah satu motivasi terciptanya alat pemantauan kualitas air yang bernama Bank Air Kami. Alat yang mengimplementasi konsep *Internet of Things* (IoT) untuk fungsionalitas sistem mengandalkan internet untuk menjalankan perintah, menerima pesan, dan berkomunikasi. Komunikasi seperti ini rentan terhadap serangan pada jaringan internet dengan serangan yang dapat merusak kinerja sistem, sehingga produksi data kualitas air dapat terganggu. Penelitian ini bertujuan untuk menginisiasi fitur-fitur keamanan pada komunikasi data sistem sebagai pengembangan pada versi sebelumnya yang dinamakan Bank Air Kami v2. Desain keamanan mengacu pada aspek-aspek trias CIA (*Confidentiality, Integrity, Availability*) dalam menentukan fitur-fitur keamanan pada sistem. Aspek Kerahasiaan (*Confidentiality*) berkaitan dengan menjaga kerahasiaan komunikasi data, Aspek Integritas (*Integrity*) berkaitan dengan kepercayaan atas data, dan Aspek Ketersediaan (*Availability*) berkaitan dengan ketersediaan data. Penelitian ini berhasil mengimplementasi desain keamanan pada *Virtual Machine* Raspberry Pi yang menerapkan *Transport Layer Security* (TLS) pada protokol *Message Queueing Telemetry Transport* (MQTT) untuk mencegah *Eavesdropping*, menerapkan algoritma *hash* menggunakan *SHA-256* yang diimplementasi pada simulasi transmisi data melalui platform kontroler Node-RED untuk mencegah *Data Tampering*, dan menerapkan *Snort Network Intrusion Detection System* (NIDS) dan *Firewall* untuk mencegah serangan *Distributed Denial of Service* (DDoS) dan *Port Scanning*.

Kata kunci: Trias CIA, Keamanan Siber, MQTT, Node-RED, Snort.

GLOSARIUM

Alamat IP	Label numerik yang diberikan pada setiap perangkat yang terkoneksi kepada jaringan yang menggunakan protokol internet.
Broker	Program perantara komunikasi antara perangkat yang menggunakan protokol MQTT untuk berkomunikasi.
Debug	Proses mengidentifikasi dan menganalisis luaran dari sebuah program untuk memantau adanya galat atau kesalahan.
File	Kontainer pada sistem komputer untuk menyimpan informasi seperti teks, gambar, suara, dan video.
Flow	Alur yang terdiri dari sekumpulan perintah pada titik yang digunakan pada platform Node-RED.
Flowchart	Diagram alir untuk mendefinisikan suatu proses.
Interface	Poin interaksi antara dua sistem atau komponen untuk saling berkomunikasi.
Generate	Proses untuk menciptakan data yang baru menggunakan algoritma yang terprogram luarannya.
Hash	Fungsi yang mengonversi masukan seperti pesan menjadi kumpulan huruf sepanjang <i>byte</i> dari pesan untuk verifikasi identitas.
Node	Blok pada Node-RED yang berisi fungsi untuk melakukan perintah pada <i>Flow</i> .
Port	Titik akhir komunikasi pada jaringan komputer untuk mengizinkan layanan jaringan untuk berada pada satu perangkat fisik yang sama
Publish	Aksi untuk membuat informasi atau data tersedia bagi perangkat lain.
Rule	Panduan atau prinsip pada komputasi untuk melakukan aksi.
Subnet	Bagian dari jaringan IP yang diciptakan oleh belahan jaringan yang lebih besar menjadi lebih kecil yang lebih dapat dikelola
Subsrcibe	Aksi untuk meminta dan menerima pembaruan atau notifikasi terkait informasi atau kejadian spesifik.

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN DOSEN PEMBIMBING.....	ii
HALAMAN PENGESAHAN DOSEN PENGUJI.....	iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR	iv
HALAMAN PERSEMBAHAN	v
HALAMAN MOTO	vi
KATA PENGANTAR	vii
SARI	ix
GLOSARIUM.....	x
DAFTAR ISI.....	xi
DAFTAR TABEL.....	xiv
DAFTAR GAMBAR	xv
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah	3
1.3 Lingkup Masalah	3
1.4 Tujuan Penelitian.....	3
1.5 Manfaat Penelitian.....	3
1.6 Metode Penelitian	4
1.6.1 Analisis Kebutuhan.....	4
1.6.2 Studi Pustaka	5
1.6.3 Perancangan Desain.....	5
1.6.4 Implementasi dan Pengujian.....	5
1.6.5 Penyusunan Laporan.....	5
1.7 Sistematika Penulisan	5
BAB II TINJAUAN PUSTAKA	7
2.1 Keamanan Siber pada <i>Internet of Things</i> (IoT)	7
2.1.1 Kasus Ancaman Keamanan Siber pada <i>Internet of Things</i> (IoT)	8
2.1.2 Tujuan Keamanan pada <i>Internet of Things</i> (IoT)	9
2.2 Model Trias CIA (<i>Confidentiality, Integrity, Availability</i>).....	20
2.3 Teknik Keamanan Trias CIA berdasarkan Relevansinya dengan Lapisan IoT	21
2.4 Domain Model CIA pada Keamanan Siber Bank Air Kami v2	22

2.4.1	Aspek Kerahasiaan (<i>Confidentiality</i>).....	23
2.4.2	Aspek Integritas (<i>Integrity</i>).....	23
2.4.3	Aspek Kerahasiaan (<i>Availability</i>).....	24
2.5	Hukum Indonesia terhadap Perlindungan Data	25
2.6	Regulasi terhadap Data Sensitif dan Relevansinya pada Bank Air Kami v2	26
2.7	Protokol <i>Message Queueing Telemetry Transport</i> (MQTT).....	28
2.8	Platform Kontroler Node-RED.....	31
2.9	<i>Intrusion Detection System</i> (IDS).....	32
2.10	Snort	33
BAB III METODOLOGI PENELITIAN		35
3.1	Identifikasi Kebutuhan Keamanan Bank Air Kami v2.....	35
3.2	Rencana Desain Keamanan Data pada Sistem Bank Air Kami v2	37
3.3	Tujuan Desain Keamanan.....	38
3.3.1	Tujuan Aspek Kerahasiaan (<i>Confidentiality</i>)	38
3.3.2	Tujuan Aspek Integritas (<i>Integrity</i>)	39
3.3.3	Tujuan Aspek Ketersediaan (<i>Availability</i>).....	39
3.4	Tahapan Implementasi Desain Keamanan	40
3.4.1	Rencana Implementasi MQTT	41
3.4.2	Rencana Implementasi Node-RED.....	42
3.4.3	Rencana Implementasi Snort	44
3.4.4	Rencana Implementasi UFW <i>Firewall</i>	45
3.5	Rencana Pengujian pada Desain Keamanan.....	45
3.5.1	Aspek Kerahasiaan (<i>Confidentiality</i>).....	45
3.5.2	Aspek Integritas (<i>Integrity</i>).....	46
3.5.3	Aspek Ketersediaan (<i>Availability</i>)	47
BAB IV IMPLEMENTASI DAN PENGUJIAN.....		48
4.1	Implementasi Desain Keamanan	48
4.1.1	Implementasi Aspek Kerahasiaan (<i>Confidentiality</i>).....	48
4.1.2	Implementasi Aspek Integritas (<i>Integrity</i>).....	53
4.1.3	Implementasi Aspek Ketersediaan (<i>Availability</i>)	60
4.2	Pengujian Desain Keamanan	63
4.2.1	Pengujian pada Implementasi Aspek Kerahasiaan (<i>Confidentiality</i>)	63
4.2.2	Pengujian pada Implementasi Aspek Integritas (<i>Integrity</i>)	66
4.2.3	Pengujian pada Implementasi Aspek Ketersediaan (<i>Availability</i>).....	68

4.3 Hasil Implementasi dan Pengujian	76
4.4 Kekurangan Sistem.....	77
BAB V KESIMPULAN DAN SARAN	78
5.1 Kesimpulan.....	78
5.2 Saran.....	78
DAFTAR PUSTAKA	79
LAMPIRAN.....	83

DAFTAR TABEL

Tabel 2.1 Teknik Keamanan berdasarkan Lapisan IoT	22
Tabel 2.2 Pasal-pasal UU PDP	27
Tabel 2.3 Komparasi Literatur pada Protokol Komunikasi	28
Tabel 3.1 Desain keamanan data pada Bank Air Kami v2	37
Tabel 3.2 Tujuan aspek kerahasiaan	38
Tabel 3.3 Tujuan aspek integritas	39
Tabel 3.4 Tujuan aspek ketersediaan	39
Tabel 3.5 Data-data untuk dilakukan simulasi	42
Tabel 4.1 Hasil Implementasi	76
Tabel 4.2 Kekurangan Sistem	77

DAFTAR GAMBAR

Gambar 1.1 Diagram <i>Waterfall</i> Metode Penelitian	4
Gambar 2. 1 Taksonomi kebutuhan keamanan IoT pada Level Operasional	9
Gambar 2.2 Tiga struktur lapisan IoT	11
Gambar 2.3 Strategi keamanan pada lapisan struktur IoT	12
Gambar 2.4 Empat struktur lapisan IoT	15
Gambar 2.5 Tujuan Mitigasi Risiko.....	18
Gambar 2.6 Model Trias CIA	20
Gambar 2.7 Domain Trias CIA pada Bank Air Kami v2	23
Gambar 2.8 Klasifikasi Data Pribadi Berdasarkan UU PDP Pasal 3	26
Gambar 2.9 Skema Node-RED	31
Gambar 2.10 Contoh Penulisan Snort Rules.....	33
Gambar 2.11 <i>Snort Rules</i> mode <i>Packet Sniffing</i>	33
Gambar 2.12 <i>Snort Rules</i> mode <i>Packet Logging</i>	34
Gambar 2.13 <i>Snort Rules</i> mode NIDS	34
Gambar 3.1 Alur komunikasi data Bank Air Kami v2	35
Gambar 3.2 Topologi pengujian pada VirtualBox Machine.....	40
Gambar 3.3 Arsitektur Desain Keamanan	41
Gambar 3.4 <i>Flow</i> 1 Node-RED	43
Gambar 3.5 <i>Flow</i> 2 pada skenario normal	44
Gambar 3.6 <i>Flow</i> 2 pada skenario terjadi serangan	44
Gambar 3.7 <i>Flowchart</i> pengujian aspek kerahasiaan	45
Gambar 3.8 <i>Flowchart</i> pengujian aspek integritas	46
Gambar 3.9 <i>Flowchart</i> pengujian pada aspek ketersediaan.....	47
Gambar 4.1 Alur koneksi broker MQTT	49
Gambar 4.2 Sintaks perintah melakukan <i>generate private key</i>	50
Gambar 4.3 Sintaks perintah melakukan <i>generate</i> sertifikat untuk CA	50
Gambar 4.4 Sintaks perintah melakukan <i>generate server key</i>	50
Gambar 4.5 Sintaks perintah melakukan generate CSR	50
Gambar 4.6 Sintaks perintah melengkapi sertifikat <i>server</i> dengan sertifikat CA.....	50
Gambar 4.7 Direktori <i>file</i> sertifikat dan kunci	51
Gambar 4.8 Konfigurasi TLS pada <i>mosquitto.conf</i>	52
Gambar 4.9 Mengubah perizinan pada <i>file</i>	52

Gambar 4.10 Memberikan hak kepemilikan pada Mosquitto.....	52
Gambar 4.11 Konfigurasi TLS broker pada Node-RED	53
Gambar 4.12 Autentikasi pengguna pada Node-RED	54
Gambar 4.13 <i>Flow</i> untuk melakukan <i>generate</i> data	54
Gambar 4.14 kode <i>Javascript</i> untuk melakukan <i>generate</i> data	55
Gambar 4.15 <i>Debug</i> pada <i>flow generate</i> data	55
Gambar 4.16 <i>Flow</i> melakukan <i>publish</i> ke topik <i>sensor/data</i>	56
Gambar 4.17 <i>Flow</i> melakukan <i>subscribe</i> topik <i>sensor/data</i>	56
Gambar 4.18 <i>Debug</i> pada <i>flow publish</i> dan <i>subscribe</i>	57
Gambar 4.19 Merubah kepemilikan dan perizinan pada file <i>settings.js</i>	58
Gambar 4.20 Kode yang ditambahkan pada <i>settings.js</i>	58
Gambar 4.21 Kode <i>Javascript</i> untuk melakukan <i>hashing</i>	58
Gambar 4.22 <i>Flow subscribe</i> yang ditambahkan fungsi <i>hash</i>	59
Gambar 4.23 Perbandingan data sebelum dan sesudah <i>hashing</i>	59
Gambar 4.24 <i>Rule</i> mendeteksi <i>ICMP Ping</i>	60
Gambar 4.25 Kali Linux melakukan <i>Ping</i>	61
Gambar 4.26 Hasil deteksi Snort pada <i>ICMP Ping</i>	61
Gambar 4.27 <i>Rules</i> pada Snort untuk deteksi serangan	62
Gambar 4.28 Konfigurasi <i>Firewall ufw</i> pada Raspberry Pi.....	63
Gambar 4.29 Tangkapan paket pada <i>port 1883</i>	64
Gambar 4.30 Tangkapan paket pada <i>port 8883</i>	64
Gambar 4.31 Muatan pesan pada <i>port 1883</i>	65
Gambar 4.32 Konten pesan yang terbaca oleh Wireshark	65
Gambar 4.33 Kode Javascript untuk validasi <i>hash</i>	66
Gambar 4.34 Kode Javascript untuk melakukan <i>Data Tampering</i>	66
Gambar 4.35 <i>Flow 2</i> ketika melakukan <i>Data Tampering</i>	67
Gambar 4.36 <i>Debug</i> ketika <i>hash</i> valid.....	67
Gambar 4.37 <i>Debug</i> ketika <i>hash</i> tidak valid.....	67
Gambar 4.38 Perintah melakukan <i>LAND Attack</i>	68
Gambar 4.39 Tangkapan paket ketika pengujian <i>LAND Attack</i>	69
Gambar 4.40 Lalu lintas jaringan ketika pengujian <i>LAND Attack</i>	69
Gambar 4.41 Deteksi Snort pada <i>LAND Attack</i>	70
Gambar 4.42 Perintah melakukan <i>SYN Flood Attack</i>	71

Gambar 4.43 Tangkapan paket ketika pengujian <i>SYN Flood Attack</i>	71
Gambar 4.44 Sistem monitor ketika pengujian <i>SYN Flood Attack</i>	71
Gambar 4.45 Deteksi Snort pada <i>SYN Flood Attack</i>	72
Gambar 4.46 Perintah melakukan <i>Smurf Attack</i>	73
Gambar 4.47 Tangkapan paket ketika pengujian <i>Smurf Attack</i>	73
Gambar 4.48 Sistem monitor ketika pengujian <i>Smurf Attack</i>	73
Gambar 4.49 Deteksi Snort pada <i>Smurf Attack</i>	74
Gambar 4.50 Perintah melakukan <i>TCP ACK Scan</i>	74
Gambar 4.51 Deteksi Snort pada <i>TCP ACK Scan</i>	75
Gambar 4.52 Permintaan <i>subscribe</i> pada topik	75
Gambar 4.53 Permintaan <i>subscribe</i> tidak diterima	76

BAB I

PENDAHULUAN

1.1 Latar Belakang

Menciptakan generasi unggul dalam upaya peningkatan kualitas kesejahteraan bangsa adalah kewajiban bagi semua orang untuk meneladaninya. Pemenuhan kebutuhan manusia dalam kehidupannya merupakan sebagian dari upaya mempersiapkan generasi unggul berkelanjutan dengan mempertahankan kualitas asupan manusia, khususnya pada kebutuhan air. Akses air bersih untuk kebutuhan manusia adalah hal terpenting untuk masyarakat yang sehat, dan produktif sebagai modal dasar dalam mewujudkan bangsa yang adil, makmur, dan sejahtera (Purwanto, 2020). Selain digunakan untuk kebutuhan pokok bagi konsumsi dan sanitasi manusia (Kornita, 2020), air juga berperan terhadap daerah pertanian, perikanan, industri, pemukiman, energi, wisata, dan lainnya (Lomi et al., 2021). Potensi dalam ketersediaan air bersih di Indonesia cukup melimpah, tetapi beberapa kalangan masyarakat masih mengalami kesulitan dalam mengakses air bersih untuk memenuhi kebutuhannya, target *Sustainable Development Goals* (SDGs) untuk capaian air bersih di Indonesia adalah 100%, sementara di Indonesia baru mencapai 72.55% (Kurniawati et al., 2020). Hal ini didukung dengan keterbatasan masyarakat dalam mengakses air bersih yang berjumlah 87.75% dari populasi pada tahun 2018, hanya 20,14% populasi yang mengakses air bersih layak minum (Purwanto, 2020). Segala bentuk upaya untuk mencegah terjadinya penyempitan ketersediaan air bersih perlu dilakukan oleh semua orang sebagai bentuk kesadaran dari pentingnya air bersih untuk kehidupan semua makhluk.

Hal itu yang mendasari diciptakannya alat pemantau kualitas air bernama Bank Air Kami yang diciptakan oleh peneliti sebelumnya. Alat ini menggunakan teknologi *Internet of Things* (IoT) dalam proses operasionalnya. Bank Air Kami berfungsi sebagai pengukuran kualitas air pada sumber mata air yang digunakan oleh masyarakat sekitar di Terban, Gondokusuman, Yogyakarta. Dengan tujuan untuk menghasilkan informasi kualitas air yang dapat dilihat oleh masyarakat melalui *KMS* sebagai bentuk peningkatan kesadaran bagi konsumen air (Syahputra et al., 2019). Bank Air Kami mengukur dengan sensor yang dipasangkan pada sumber air dan mentransmisikan datanya kepada internet untuk divisualisasi oleh *KMS*. Dengan menggunakan *Internet of Things* (IoT), perangkat keras seperti sensor untuk pengukuran air dan perangkat lunak seperti *website* mampu berkomunikasi dan membentuk jaringan komunikasi.

Akan tetapi, pada sistem ini terdapat beberapa masalah, yaitu kurangnya infrastruktur keamanan yang menyebabkan adanya kerentanan pada sistem dan alur komunikasi yang tidak efektif. Sehingga menginisiasi dilakukannya peningkatan dan pembaharuan pada sistem untuk mengatasi hal tersebut. Dengan hal itu, akan diciptakan Bank Air Kami v2 dengan memprioritaskan pada proses komunikasi data yang efektif dan terjamin keamanannya. Pengembangan Bank Air Kami v2 ini dijalankan oleh tiga orang dengan sub penelitian yang berbeda, peneliti akan merancang sebuah desain keamanan pada komunikasi data yang sesuai dengan kebutuhan sistem untuk menunjang kinerja sistem Bank Air Kami v2. Kevin Fernando Atalaric akan merancang infrastruktur fisik terkait sistem dengan mengadaptasi sistem sebelumnya yang diberikan pembaharuan. Wilwatika Tripawangsa akan mengembangkan sebuah *web dashboard* sebagai visualisasi data kondisi air yang dapat diakses oleh masyarakat ataupun pihak yang membutuhkannya.

Pada sub penelitian ini, akan menciptakan desain keamanan pada komunikasi data sistem Bank Air Kami v2 untuk mencegah terjadinya kerentanan yang berisiko terhentinya proses pengukuran pada air dan komunikasi data dengan mengacu pada aspek ketersediaan dan kepercayaan data. Kerentanan pada sistem dapat berupa ancaman terhadap keamanan secara fisik, ancaman keamanan pada perangkat lunak, dan jenis ancaman lain yang menyebabkan kegagalan sistem, hal ini disebut sebagai keamanan siber. Keamanan siber adalah kumpulat alat, kebijakan, konsep keamanan, perlingungan keamanan, pedoman, pendekatan manajemen risiko, tindakan, pelatihan, praktik terbaik, jaminan dan teknologi yang dapat digunakan untuk melindungi lingkungan siber dan organisasi serta aset pengguna (Craig et al., 2014).

Dengan maraknya penggunaan *Internet of Things* (IoT) untuk mempermudah pekerjaan manusia, tidak luputnya berbagai macam serangan terhadap informasi, sehingga keamanan informasi pada IoT perlu menjadi perhatian dan pertimbangan (Reshan, 2021). Sama halnya dengan meningkatkan keamanan siber pada sistem. Maka dari itu, peneliti akan menggunakan kerangka kerja keamanan bernama Trias CIA (*Confidentiality, Integrity, Availability*) sebagai acuan dalam perancangan desain komunikasi dan keamanan pada Bank Air Kami v2. Trias CIA mengisyaratkan untuk menjaga kerahasiaan (*Confidentiality*) pada data, menjaga kepercayaan atau integritas (*Integrity*) pada data, dan memastikan sistem bekerja dengan baik dan terjaga ketersediaannya (*Availability*) pada setiap proses pengolahan data. Trias CIA telah lama digunakan dalam keamanan informasi untuk membantu mengevaluasi dan membuat sistem informasi, digunakan dengan cara melindungi ketiga aspek tersebut. Apabila sistem

tidak bekerja dengan baik pada ketiga aspek tersebut, maka sistem tidak memadai (Hoffman & Zahadat, 2018).

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, diperoleh rumusan masalah yang berupa: Bagaimana kerangka desain keamanan pada komunikasi data sistem Bank Air Kami v2 yang meliputi fitur-fitur keamanan untuk memastikan data yang terjaga kerahasiaanya, kepercayaanya, dan ketersediaannya dengan mengacu pada ketiga aspek trias CIA (*Confidentiality, Integrity, Availability*)?

1.3 Lingkup Masalah

Untuk menjaga fokus pada penelitian ini maka ditentukan lingkup masalah pada penelitian “Desain Keamanan pada Komunikasi Data Sistem Bank Air Kami v2 Menggunakan Trias CIA”, adalah:

- a. Penelitian ini hanya berfokus pada pengembangan desain keamanan pada komunikasi data sistem Bank Air Kami v2 dan mengujinya, sehingga hal-hal yang berkaitan dengan pengembangan dan perancangan infrastruktur tidak dibahas pada penelitian ini.
- b. Dalam menciptakan desain keamanan pada komunikasi data sistem Bank Air Kami v2, peneliti hanya mengacu pada elemen trias CIA (*Confidentiality, Integrity, Availability*) untuk pengembangan, perancangan, dan pengujian desain, sehingga tidak menerapkan kerangka kerja lain.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah menciptakan desain keamanan pada komunikasi data sistem Bank Air Kami v2 yang didasari dengan kebutuhan sistem untuk menunjang kinerja secara efektif, aman, dan kokoh dari segi keamanan yang mengacu pada trias CIA (*Confidentiality, Integrity, Availability*) dan melakukan pengujian untuk menguji fungsionalitas.

1.5 Manfaat Penelitian

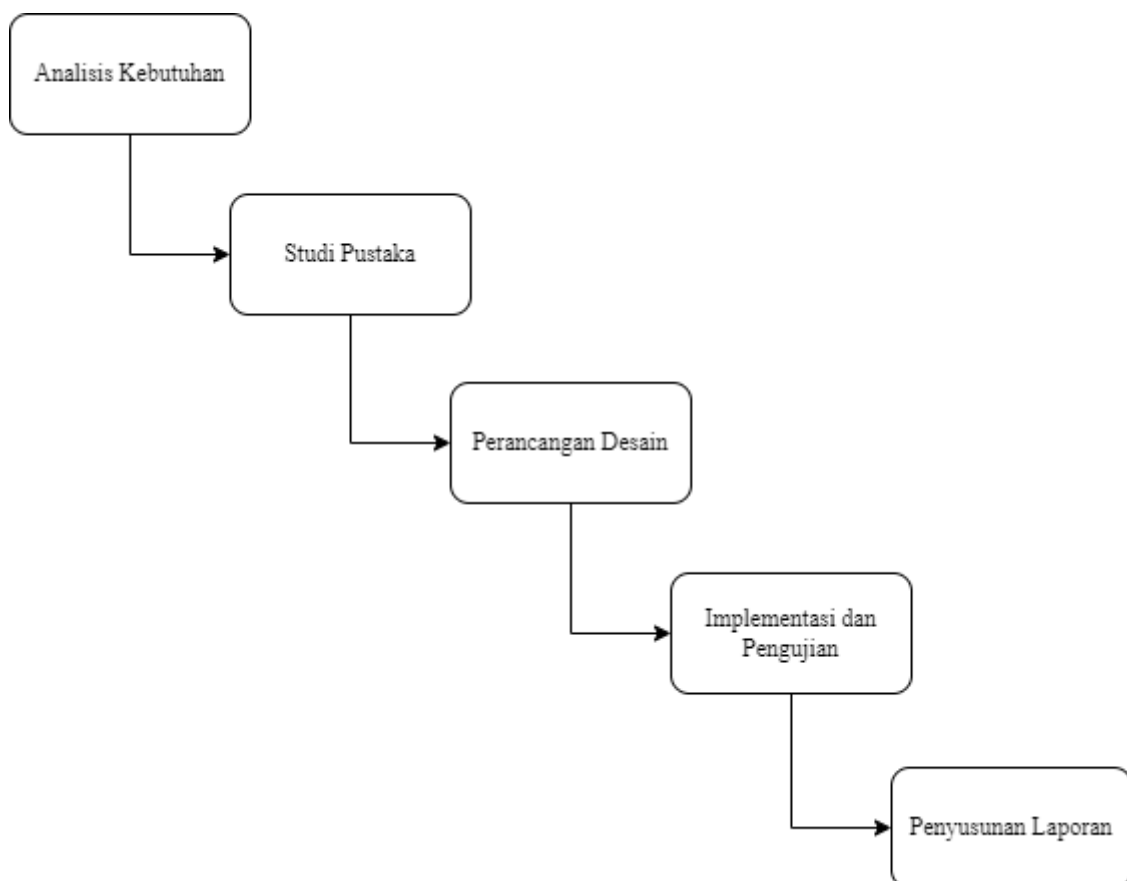
Manfaat yang dapat diambil dari penelitian ini adalah:

- a. Membuat perencanaan dalam mengatasi masalah keamanan pada komunikasi data sistem Bank Air Kami v1.

- b. Membuat desain keamanan pada komunikasi data sistem menggunakan trias CIA (*Confidentiality, Integrity, Availability*).
- c. Melakukan pengujian pada desain sehingga mampu mengetahui berbagai jenis serangan dan mitigasinya pada perangkat IoT (*Internet of Things*).

1.6 Metode Penelitian

Untuk menyelesaikan rumusan masalah dan mendapatkan hasil yang diharapkan, akan dilakukan beberapa tahapan dalam menyelesaikan penelitian ini. Tahapan-tahapan tersebut meliputi:



Gambar 1.1 Diagram *Waterfall* Metode Penelitian

1.6.1 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk mengidentifikasi kebutuhan terkait sistem yang meliputi masalah-masalah yang terjadi pada Bank Air Kami v1, kebutuhan untuk pengembangan pada versi keduanya, dan tujuan yang akan dicapai pada penelitian ini. Untuk mengumpulkan informasi yang dibutuhkan pada penelitian ini, peneliti melakukan wawancara

kepada Pak Ari Sujarwo, S.Kom., MIT (Hons) selaku dosen pembimbing peneliti dan inisiator pada pengembangan Bank Air Kami v1. Informasi yang didapat akan dijadikan sebagai acuan dalam penyelesaian masalah, pemenuhan kebutuhan, dan tujuan utama dalam mencapai kesuksesan pengembangan Bank Air Kami v2, terutama pada tujuan perancangan keamanan pada komunikasi datanya.

1.6.2 Studi Pustaka

Tahap ini dilakukan untuk mengumpulkan literatur atau penelitian yang relevan sebagai landasan atau bahan pertimbangan dalam perancangan desain keamanan pada komunikasi data, sehingga menghasilkan kerangka desain yang kokoh untuk menunjang kinerja sistem.

1.6.3 Perancangan Desain

Desain keamanan pada komunikasi sistem Bank Air Kami v2 akan diciptakan berdasarkan rumusan masalah dan tahap studi pustaka untuk merumuskan langkah dan tindakan terkait perancangan keamanan yang kokoh. Setelah desain terbentuk, desain keamanan diimplementasi sebagai simulasi fitur-fitur keamanan pada sistem yang menyesuaikan dengan infrastruktur untuk komunikasi data pada Bank Air Kami v2.

1.6.4 Implementasi dan Pengujian

Implementasi desain keamanan yang telah dirancang sebelumnya sebagai simulasi. Pengujian dilakukan untuk menguji ketahanan pada rancangan desain keamanan terhadap skenario komunikasi data sistem Bank Air Kami v2 dan kemungkinan serangan melalui metode yang mengacu pada trias CIA (*Confidentiality, Integrity, Availability*).

1.6.5 Penyusunan Laporan

Tahapan terakhir dalam penelitian ini merupakan penyusunan laporan, mencakup ringkasan hasil penelitian dan kesimpulan mengenai pengujian desain yang telah dilakukan untuk memberikan pandangan terhadap desain keamanan yang mengacu pada trias CIA (*Confidentiality, Integrity, Availability*).

1.7 Sistematika Penulisan

Tujuan penyusunan sistematika penulisan adalah untuk memudahkan pembaca dalam memahami isi penelitian. Terdapat 5 bab dalam penelitian ini, yaitu: BAB I memuat

Pendahuluan. Pada bab ini, peneliti akan menjelaskan mengenai masalah umum yang mendasari penelitian ini, identifikasi masalah, tujuan penelitian, ruang lingkup penelitian, manfaat penelitian, dan sistematika penulisan. BAB II memuat Tinjauan Pustaka. Pada bab ini akan memaparkan terkait teori dan metode berdasarkan penelitian atau literatur yang relevan dengan penelitian sebagai landasan perancangan desain, serta pengambilan keputusan pada kerangka desain. BAB III memuat Metodologi Penelitian. Pada bab ini menjelaskan rancangan desain secara keseluruhan yang meliputi rencana implementasi dan rencana pengujian pada desain keamanan. BAB IV memuat Implementasi dan Pengujian penelitian. Pada bab ini memuat implementasi dan pengujian pada desain keamanan yang telah dirancang untuk menguji fungsionalitas pada infrastruktur komunikasi data sistem Bank Air Kami v2 berdasarkan ketiga aspek trias CIA. BAB V memuat Kesimpulan dan Saran. Bab ini berisi kesimpulan yang didapatkan serta saran yang dapat menunjang dalam pengembangan sistem lebih lanjut.

BAB II

TINJAUAN PUSTAKA

2.1 Keamanan Siber pada *Internet of Things* (IoT)

Internet of Things (IoT) memungkinkan berbagai macam perangkat di lingkungan kita berkomunikasi satu sama lain dalam satu jaringan, hal ini memungkinkan untuk mengenal perangkat satu sama lain dan bertindak secara mandiri tanpa adanya interaksi manusia. Keuntungan *Internet of Things* (IoT) salah satunya mampu mempermudah sebagian pekerjaan manusia dan menghemat waktu, dan juga mampu membuka peluang baru untuk perkembangan, inovasi, dan pertukaran ilmu pengetahuan antar entitas. Butun et al. (2019) menyebutkan bahwa diperkirakan pasar IoT akan bertumbuh dari sekitar 15 miliar perangkat di 2015 menjadi lebih dari 75 miliar di 2023, sehingga rata-rata manusia di bumi akan mempunyai setidaknya 25 perangkat IoT personal dalam 7 tahun. *Wireless Sensors Network* (WSN) akan diintegrasikan ke dalam IoT dan banyak sekali *node* sensor akan masuk ke internet dengan tujuan untuk bekerja sama antara *nodes* pada periode ini. IoT akan menyediakan interaksi antara orang-orang dengan lingkungan menggunakan WSN lebih banyak di masa depan. Penelitian *Cisco* terkini juga mengatakan bahwa diperkirakan 43% kenaikan koneksi IoT dari tahun ke tahun. Sayangnya, angka kenaikan tersebut diikuti dengan meningkatnya insiden keamanan informasi dan potensi pembobolan data personal dengan dampak yang signifikan pada privasi individu di seluruh dunia (Vojkovic & Milenkovic, 2019).

Praktik dari perlindungan data sensitif pada internet dan pada perangkat dari serangan, penghancuran, atau akses tidak berizin dikenal sebagai keamanan siber atau *cybersecurity*. Tujuan dari keamanan siber adalah untuk menciptakan lingkungan yang aman dan terjaga di mana data, jaringan, dan perangkat mampu terlindungi dari serangan siber atau *cyber-attacks* (Khandare et al., 2023). Keamanan pada IoT dan WSN adalah sebuah isu penting, khususnya pada skenario yang memprioritaskan tugas kritis, seperti contoh pada bidang kesehatan. Butun et al. (2019) mengungkapkan bahwa kebanyakan sistem yang digunakan gagal menanamkan sistem keamanan yang mampu melindungi privasi pasien, hal ini menyebabkan kebocoran data-data sensitif pasien karena kegagalan sistem atau perilaku *nodes* yang buruk. Meneghello et al. (2019) menyatakan bahwa keamanan siber tidak hanya terdapat pada aspek teknis, tetapi penting juga untuk mengembangkan budaya keamanan siber pada *stakeholders* IoT, khususnya produsen dan pengguna. Beberapa perangkat seperti pengontrol lampu, *CCTV*, otomasi rumah, dan lain-lain. di desain untuk bekerja pada sistem terisolasi, untuk itu ancaman keamanan lebih

terbatas. Sebagai konsekuensi, banyak produsen tidak memiliki keahlian yang kuat dalam keamanan siber dan mungkin tidak menyadari risiko keamanan terkait koneksi perangkat terhadap jaringan global. Begitu juga dengan pengguna perlu meningkatkan kesadaran terhadap implementasi keamanan siber pada perangkat IoT.

2.1.1 Kasus Ancaman Keamanan Siber pada *Internet of Things* (IoT)

Vojkovic & Milenkovic, (2019) menunjukkan beberapa kasus ancaman keamanan siber pada IoT yang terjadi dalam beberapa tahun. Kasus-kasus tersebut berupa serangan pada *platform* penyedia konten *streaming*, percobaan distribusi *Malware* secara masif, dan peretasan perangkat medis.

- a. Serangan *Distributed Denial of Service* (DDoS) terbesar pernah terjadi pada penyedia layanan *Dyn* menggunakan *botnet* IoT pada tahun 2016, diikuti dengan kegagalan dan ketidaktersediaan bagian besar dari internet, termasuk *Twitter*, *Guardian*, *Netflix*, *Reddit*, dan *CNN*. IoT botnet telah diaktifkan oleh perangkat lunak berbahaya bernama *Mirai*. Sekali terinfeksi oleh *Mirai*, komputer secara konstan mencari internet untuk kerentanan pada perangkat IoT, lalu mengontaminasi mereka dengan *Malware* yang menggunakan kredensial. Di antara perangkat yang diserang adalah kamera digital dan pemutar *DVR*.
- b. Badan Pengawas *Food and Drug Administration* (FDA) Amerika Serikat mengonfirmasi adanya beberapa perangkat medis yang memiliki kerentanan. Hal ini menyebabkan peretas mampu mengakses perangkat seperti alat pemacu jantung dan *defibrillator* yang digunakan untuk mengontrol kerja jantung pasien dan mencegah terjadinya serangan jantung. Karena kerentanan yang terdapat pada *transmitter*, peretas mampu mengendalikan guncangan, mengelola tempo yang salah, dan menguras baterai.
- c. Olaniyi et al. (2023) menunjukan sebuah kasus mengenai peretasan yang terjadi pada perusahaan *TRENDnet* pada tahun 2013, yaitu peretasan pada kamera keamanan yang menyebabkan tereksposnya rekaman video langsung dari 700 pelanggan mereka. *Federal Trade Commission* (FTC) menginvestigasi kejadian ini dan mengungkapkan bahwa kamera dilengkapi dengan perangkat lunak yang cacat. Kerentanan pada perangkat lunak ini menyebabkan akses tidak disetujui kepada rekaman video langsung, menghasilkan pengawasan yang tidak sah, menguping, dan intrusi ilegal kepada privasi individu. Insiden ini disorot sebagai celah antara keamanan IoT dan perangkat lunak yang teruji serta perlindungan privasi pada perangkat yang terhubung.

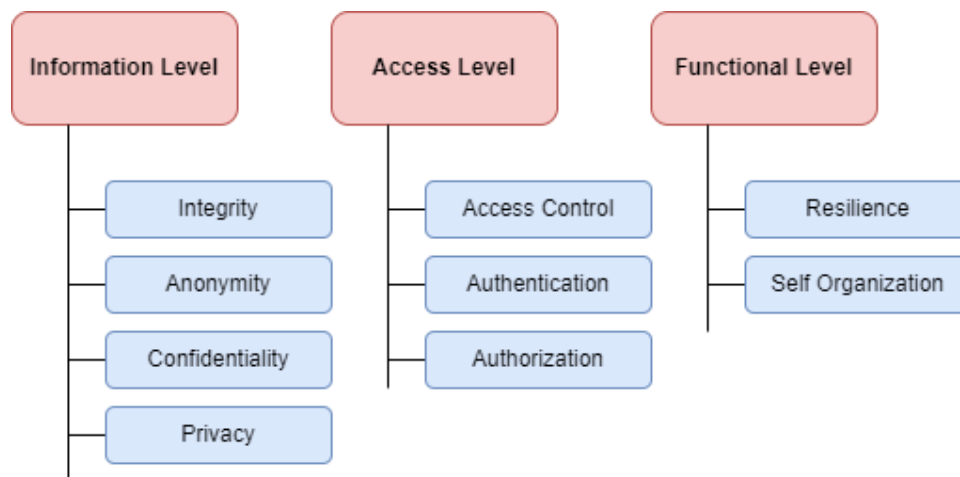
Hal tersebut membuat isu keamanan menjadi lebih kompleks dan rumit antara legal dan teknologi. Sehingga edukasi tentang keamanan informasi merupakan bagian yang tidak bisa terpisahkan dari literasi digital, dan menggunakan metode yang tepat untuk perlindungan informasi.

2.1.2 Tujuan Keamanan pada *Internet of Things* (IoT)

Serangan pada perangkat IoT seringkali sederhana dan mudah dilakukan. Hal ini berdampak pada pelanggaran privasi pengguna dan kebocoran informasi sensitif. Data yang terkumpul dapat berupa temperatur ruangan, pengukur kelembapan, sinyal detak jantung, lokasi pengguna, dan kebiasaan hidup. Berikut beberapa pendapat mengenai tujuan atau strategi keamanan dari berbagai literatur dan penelitian berdasarkan domain struktur IoT:

Tujuan Keamanan Berdasarkan Level Operasional

Meneghello et al. (2019) dalam penelitiannya mengemukakan beberapa strategi atau kebutuhan keamanan pada perangkat IoT. Terdapat 3 level operasional, yaitu *Information*, *Access*, dan *Functional Level*.



Gambar 2.1 Taksonomi kebutuhan keamanan IoT pada Level Operasional

a. *Information Level*

Pada level ini, keamanan perlu menjamin beberapa kebutuhan:

1. *Integrity*: Data yang diterima tidak boleh diubah selama transmisi.
2. *Anonymity*: Identitas sumber data selalu tersembunyi dari pihak ketiga.

3. *Confidentiality*: Data tidak dapat dibaca oleh pihak ketiga. Hubungan yang dapat dipercaya harus dibangun antara 3 perangkat IoT untuk bertukar informasi yang dilindungi. Pesan yang direplikasi harus mudah dikenali.
4. *Privacy*: Informasi pribadi klien tidak boleh diungkapkan selama pertukaran data. Penyadap akan sulit untuk menyimpulkan informasi yang didapat.

b. *Access Level*

Menetapkan beberapa mekanisme keamanan untuk mengontrol akses ke jaringan, berikut beberapa fungsionalitas yang tersedia:

1. *Access Control*

Menjamin hanya pengguna yang sah mampu mengakses perangkat dan jaringan untuk tugas-tugas administratif, seperti pemrograman ulang jarak jauh atau kontrol jaringan, dan perangkat IoT.

2. *Authentication*

Melakukan pengecekan apakah sebuah perangkat memiliki hak untuk mengakses jaringan dan apakah sebuah jaringan memiliki hak untuk terkoneksi kepada perangkat. Seperti halnya operasi pertama kali yang dilakukan oleh sebuah *node* ketika bergabung dengan jaringan baru. Setiap perangkat perlu menyediakan prosedur autentikasi yang kuat untuk menghindari ancaman keamanan. Seperti contoh, jika semua perangkat IoT diproduksi oleh produsen yang sama dikonfigurasi dengan kredensial autentikasi yang sama, maka peretasan pada satu perangkat dapat membahayakan semua aspek keamanan di tingkat informasi.

3. *Authorization*

Memastikan bahwa hanya perangkat resmi pengguna yang mendapatkan akses ke layanan atau sumber daya jaringan.

c. *Functional Level*

Pada level ini mendefinisikan kebutuhan keamanan dalam beberapa kriteria berikut:

1. *Resilience*

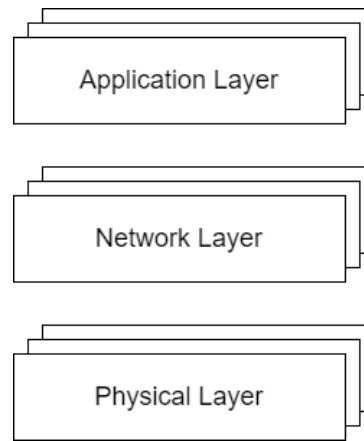
Ini merujuk pada kapasitas jaringan untuk memastikan keamanan perangkatnya, bahkan jika terjadi serangan dan kegagalan.

2. *Self Organization*

Menunjukkan kemampuan sistem IoT untuk menyesuaikan dirinya agar tetap beroperasi bahkan jika terjadi kegagalan pada beberapa bagian karena malfungsi atau serangan berbahaya secara terus menerus.

Tujuan Keamanan Berdasarkan Lapisan Struktur

Andrea et al. (2015) dalam literaturnya menyatakan bahwa IoT pada dasarnya tersusun dari 3 lapisan, yaitu *Application Layer*, *Network Layer*, dan *Physical Layer*.



Gambar 2.2 Tiga struktur lapisan IoT

a. *Physical Layer*

Lapisan paling bawah dari arsitektur adalah lapisan yang bertanggung jawab untuk interkoneksi perangkat. Tujuan utamanya adalah untuk melakukan identifikasi perangkat dan menyediakan penemuan layanan. Perangkat ini bisa berbagai jenis, seperti *Arduino*, *Raspberry*, *Zigbee*, dan lain-lain.

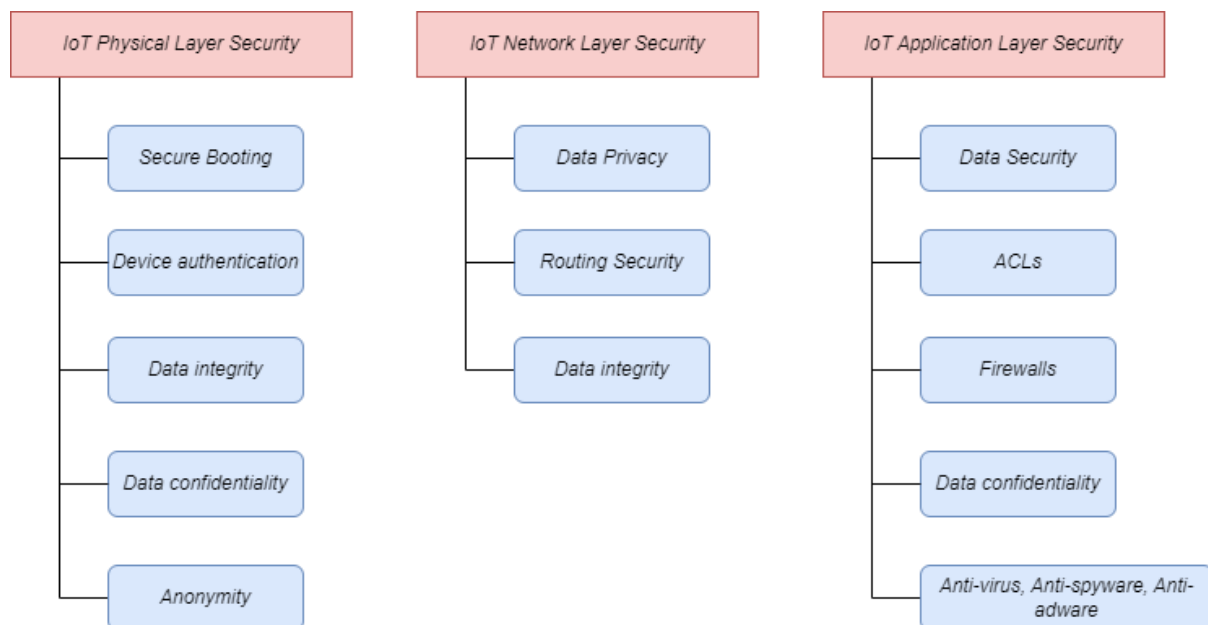
b. *Network Layer*

Selayaknya model *Network Layer* pada umumnya, lapisan ini terdapat antarmuka jaringan, saluran komunikasi, pengelolaan jaringan, pemeliharaan informasi, dan pemrosesan cerdas. Lapisan ini bertanggung jawab untuk komunikasi dan konektivitas seluruh perangkat pada sistem IoT dengan bantuan berbagai protokol komunikasi. Protokol paling umum yang digunakan adalah *Message Queueing Telemetry Transport* (MQTT) dan *Constrained Application Protocol* (CoAP). *Network Layer* mengumpulkan dan menggabungkan komunikasi dari berbagai perangkat dan menyediakan kemampuan untuk mengirim komunikasi kepada perangkat spesifik.

c. *Application Layer*

Lapisan ini berorientasi pada layanan, yang mana memastikan layanan dengan tipe yang sama antar perangkat yang terhubung. Lapisan ini mampu menyimpan data pada *database* yang menyediakan kemampuan penyimpanan data yang terkumpul.

Andrea et al. (2015) dalam penelitiannya merumuskan beberapa tujuan keamanan pada domain lapisan struktur IoT. Ketiga lapisan struktur IoT pada Gambar 2.2 memiliki kerentanan dan serangan keamanan yang berbeda, sehingga dikemukakan beberapa strategi yang dikhususkan untuk setiap lapisan, di antaranya:



Gambar 2.3 Strategi keamanan pada lapisan struktur IoT

a. Keamanan pada *Physical Layer*

1. *Secure Booting*

Autentikasi dan integritas dari *software* pada perangkat seharusnya diverifikasi menggunakan *cryptographic hash algorithms*. Namun, karena kebutuhan pemrosesan dengan daya rendah menyebabkan *cryptographic hash algorithms* tidak dapat diimplementasi.

2. *Device Authentication*

Ketika perangkat baru tersambung ke internet, seharusnya mampu mengautentikasi dirinya sendiri sebelum menerima atau mentransmisi data, untuk memastikan hal tersebut teridentifikasi secara benar sebelum otorisasi dan mencegah perangkat jahat masuk ke dalam sistem.

3. *Data Integrity*

Mekanisme deteksi kesalahan seharusnya terdapat pada setiap perangkat, untuk memastikan tidak terjadi gangguan pada data sensitive. Mekanisme konsumsi daya rendah seperti *Cyclic Redundancy Checks (CRC)*, *Checksum*, *Parity Bit* dapat digunakan.

4. *Data Confidentiality*

Data harus dienkripsi pada setiap perangkat sebelum transmisi data untuk memastikan kerahasiaan. Namun, karena konsumsi daya rendah, fungsi enkripsi kriptografis kuat seperti *AES* tidak dapat diimplementasi. Sebagai pengganti, *Blowfish* atau *RSA* memiliki konsumsi daya rendah dapat diimplementasi pada perangkat *physical layer*.

5. *Anonymity*

Beberapa kasus menyembunyikan informasi sensitif seperti lokasi dan identitas *node* adalah krusial. Meskipun pendekatan *Zero-Knowledge* mampu menjadi solusi optimal untuk anonimitas, itu tidak bisa diimplementasikan pada perangkat daya rendah karena itu algoritma yang sangat kuat dan membutuhkan banyak kekuatan pemrosesan, karena itu pendekatan *K-anonymity* ideal untuk digunakan pada perangkat daya rendah.

b. Keamanan pada *Network Layer*

1. *Data Privacy*

Akses ilegal kepada *nodes* sensor dapat dicegah, menggunakan mekanisme autentikasi dan enkripsi *point-to-point*.

2. *Routing Security*

Routing yang aman adalah hal yang sangat penting bagi penggunaan jaringan sensor untuk banyak aplikasi, akan tetapi mayoritas penggunaan protokol *routing* tidak aman. Namun, keamanan *routing* dapat dipastikan dengan cara menyediakan beberapa jalur bagi *routing* data yang meningkatkan kemampuan sistem untuk mendeteksi kesalahan dan tetap berjalan pada terdapatnya kegagalan pada sistem

yang diketahui. Mekanisme enkripsi dan autentikasi meningkatkan tingkat keamanan pada *routing* data.

3. *Data Integrity*

Dengan menggunakan *Cryptographic Hash Function* (CHF), mampu memastikan integritas data yang diterima. Dalam kasus pembuktian gangguan data, mekanisme koreksi kesalahan perlu diterapkan untuk mitigasi masalah.

c. Keamanan pada *Application Layer*

1. *Data Security*

Enkripsi autentikasi dan mekanisme integritas merupakan hal yang kritis pada tingkat ini untuk memastikan keamanan dan privasi untuk keseluruhan sistem dan melindungi dari pencurian data. Mencegah akses tidak sah pada sistem dan memastikan kerahasiaan pada data sistem.

2. *Access Control Lists* (ACLs)

Pengaturan kebijakan dan izin pada siapa yang mampu mengakses dan mengontrol sistem, adalah bagian yang krusial karena memastikan privasi data, dan kesejahteraan sistem. *Access Control Lists* (ACLs) mampu memblokir atau mengizinkan masuk atau keluar lalu lintas, dan memberikan atau memblokir akses untuk permintaan dari pengguna yang berbeda dari dalam atau luar jaringan.

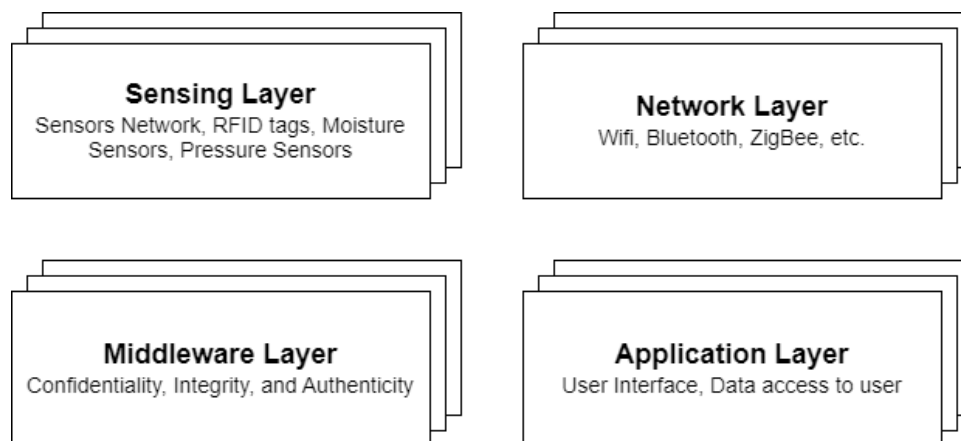
3. *Firewalls*

Firewalls mampu menjadi lapisan tambahan yang efektif yang membantu memblokir serangan daripada metode autentikasi, enkripsi dan *Access Control Lists* (ACLs) tidak bisa lakukan. Kata sandi autentikasi dan enkripsi dapat dipecahkan jika terdiri dari pemilihan sandi yang lemah. *Firewall* mampu menyaring paket yang mereka terima, memblokir paket yang tidak diinginkan, Upaya *login* yang tidak ramah, dan serangan *Denial of Services* (DoS) bahkan sebelum proses autentikasi dimulai.

4. *Anti-virus, Anti-spyware, dan Anti-adware*

Keamanan perangkat lunak seperti antivirus atau anti *spyware* penting untuk keandalan, keamanan, integritas dan kerahasiaan pada sistem IoT.

Sementara menurut Patel (2020), ada 4 lapisan arsitektur IoT. Terdapat satu lapisan tambahan yang diperkenalkan pada dunia IoT karena masalah keamanan, yaitu *Sensing Layer*, *Network Layer*, *Application Layer*, dan *Middleware Layer*.



Gambar 2.4 Empat struktur lapisan IoT

a. *Sensing Layer*

Sensing Layer berada pada lapisan dasar dari arsitektur IoT. Lapisan ini terdiri dari jaringan berbagai jenis sensor data yang mengumpulkan data dan informasi dari lingkungan dan mentransmisikannya ke jaringan. Lapisan ini sangat penting pada arsitektur IoT.

b. *Network Layer*

Network Layer adalah kumpulan dari berbagai jenis teknologi komunikasi yang berfungsi *routing* data dan transmisi informasi ke berbagai *hub* IoT dan perangkat. Pada lapisan ini, platform yang berbeda seperti *cloud computing*, *internet gateways*, dan perangkat *routing* menggunakan berbagai teknologi internet seperti *3G/4G*, *LTE*, *WiFi*, *Bluetooth*, *ZigBee*, dan lain-lain untuk mentransmisikan data ke internet.

c. *Middleware Layer*

Middleware Layer adalah lapisan perangkat lunak di antara *Network Layer* dan *Application Layer*. Lapisan ini adalah rentang keamanan di antara *Network Layer* dan *Application Layer*. Lapisan ini berfokus pada pertukaran data. Menyediakan kerahasiaan, integritas, dan keaslian pada pertukaran data.

d. *Application Layer*

Lapisan ini berada di paling atas dari arsitektur IoT. Lapisan ini menyediakan lingkungan yang ramah pengguna untuk mengakses data dari berbagai sensor. Ini membantu pengguna untuk mengidentifikasi tren dan pengambilan Keputusan.

Patel (2020) mengemukakan strategi atau teknik keamanan pada setiap lapisan arsitektur IoT yang memastikan prinsip *Confidentiality, Integrity, Availability* (CIA) terpenuhi di antaranya:

a. Teknik keamanan pada *Sensing Layer*

Sebagai lapisan yang paling bawah pada arsitektur IoT, perlu menyediakan beberapa fitur keamanan pada perangkat keras.

1. *Authenticity*

Untuk keaslian dapat mengaplikasikan *digital signature* menggunakan *cryptographic hash algorithm* yang dapat mengalahkan segala kemungkinan *Side-channel Attack* (SCA).

2. *Data Privacy*

Dapat mengaplikasikan algoritma enkripsi kriptografi ringan, yang mampu mencegah akses tidak sah pada data sensor.

b. Teknik keamanan pada *Network Layer*

Pada lapisan ini terdapat beberapa perangkat nirkabel, sehingga pemantauan komunikasi dapat dilakukan pada lapisan ini dengan mudah.

1. *Authenticity*

Menyediakan proses autentikasi yang kokoh menggunakan algoritma enkripsi *point-to-point* untuk mencegah akses tidak sah pada *nodes*.

2. *Routing Security*

Perlu adanya algoritma *routing* untuk memastikan data *routing* dalam sistem. Perlu adanya beberapa jalur *routing* untuk memastikan keamanan pada *routing* dan deteksi kesalahan.

3. *Data Privacy*

Dapat menggunakan mekanisme kontrol untuk mendeteksi intrusi dan beberapa metode integritas pada data untuk memastikan ketepatan data.

c. Teknik keamanan pada *Middleware Layer*

Lapisan ini menyediakan antarmuka antara *Network Layer* dan *Application Layer*.

1. *Confidentiality*

Algoritma kriptografi yang ringan lebih baik untuk memastikan kerahasiaan pada pertukaran data.

2. *Data Storage*

Penyimpanan data yang baik dengan enkripsi pada data yang tersimpan mampu menciptakan proteksi data yang efisien pada penyimpanan data. Menggunakan *Blockchain* terdesentralisasi daripada layanan *cloud* tersentralisasi untuk keandalan dan proteksi data yang aman.

d. Teknik keamanan pada *Application Layer*

Lapisan ini menyediakan data terproses kepada pengguna terakhir. Sehingga lapisan ini perlu lebih terjaga keamanannya.

1. *Authenticity*

Hal ini mendukung untuk menggunakan beberapa layanan autentikasi untuk mengautentikasi pengguna. Dapat menggunakan *blockchain* untuk mengautentikasi pengguna dengan memberikan data bernilai hash yang diperlukan untuk pengguna. Keamanan ini lebih baik daripada autentikasi berbasis *cloud* tersentralisasi.

2. *Intrusion Detection*

Lapisan ini mampu mengaplikasikan mekanisme deteksi intrusi untuk memantau aktivitas secara konstan dan mengirim *acknowledgement* kepada sistem jika terdapat aktivitas mencurigakan terpantau oleh sistem. Dapat menggunakan pendekatan penambahan data untuk mendeteksi intrusi.

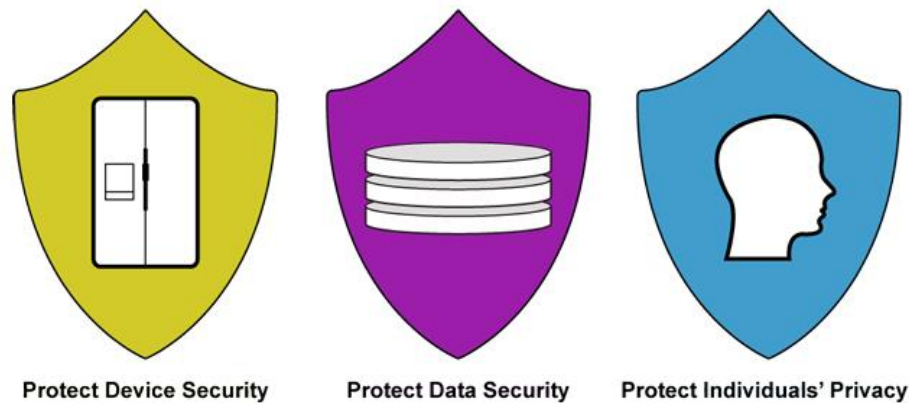
3. *Data Security*

Dapat menggunakan *firewall* untuk mencegah pencurian data.

Tujuan Keamanan berdasarkan Area Mitigasi Risiko

Pada literatur yang ditulis oleh Boeckl et al. (2019), mengemukakan area mitigasi risiko privasi, keamanan siber pada IoT, dan tujuan keamanannya. Risiko pada keamanan siber merupakan sebuah eksploitasi pada kerentanan oleh pelaku ancaman untuk membahayakan perangkat atau kerahasiaan, integritas, atau ketersediaan (CIA). Sementara risiko privasi adalah operasi yang memroses *Personal Identity Information* (PII) melalui siklus informasi untuk memenuhi misi atau kebutuhan bisnis dari sebuah organisasi atau proses PII yang berwenang dan, sebagai efek samping, menyebabkan individual untuk mengalami beberapa jenis masalah. Perangkat IoT umumnya mengalami hal yang sama seperti keamanan siber dan risiko privasi seperti halnya perangkat IT konvensional, meskipun prevalensi dan Tingkat keparahan risiko tersebut seringkali berbeda.

Risiko keamanan siber dan privasi untuk perangkat IoT dapat dikemukakan dalam tiga level tujuan mitigasi risiko:



Gambar 2.5 Tujuan Mitigasi Risiko

Sumber: Boeckl et al. (2019)

- a. Melindungi keamanan perangkat
 Dalam kata lain, mencegah digunakannya perangkat untuk melakukan serangan, termasuk berpartisipasi dalam serangan *Distributed Denial of Service* (DDoS) terhadap organisasi lain, dan menguping lalu lintas jaringan atau membahayakan perangkat lain pada segmen jaringan yang sama. Tujuan ini berlaku kepada semua perangkat IoT.
- b. Melindungi keamanan data
 Melindungi kerahasiaan, integritas, dan ketersediaan dari data yang dikumpulkan oleh, disimpan pada, diproses oleh, atau ditransmisikan kepada atau dari perangkat IoT. Tujuan ini berlaku pada setiap perangkat IoT kecuali mereka yang tidak memerlukan perlindungan data.
- c. Melindungi privasi individu
 Melindungi privasi individu terkena dampak oleh pemrosesan PII melampaui risiko yang dikelola melalui perlindungan keamanan perangkat dan data. Tujuan ini berlaku pada semua perangkat IoT yang memproses PII atau yang secara langsung atau tidak langsung berdampak pada individu.

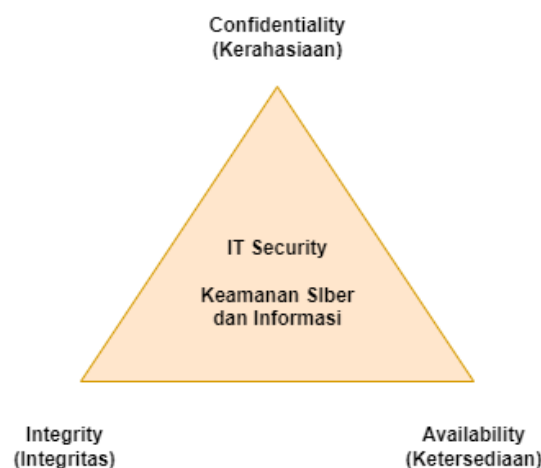
Pemenuhan setiap tujuan mitigasi risiko melibatkan penanganan serangkaian bidang mitigasi risiko. Setiap area mitigasi risiko mendefinisikan aspek mitigasi risiko keamanan siber atau privasi yang dianggap paling terkena dampak signifikan atau tidak terduga pada IoT.

- a. Area mitigasi risiko untuk tujuan 1, melindungi keamanan perangkat.
 1. Manajemen Aset
Mempertahankan arus, inventaris yang akurat pada semua perangkat IoT dan karakteristik mereka yang relevan selama siklus perangkat dengan tujuan untuk menggunakan informasi tersebut untuk kepentingan manajemen risiko keamanan siber dan privasi.
 2. Manajemen Kerentanan
Mengidentifikasi dan mengeliminasi kerentanan yang diketahui dalam *software* perangkat IoT dan *firmware* dengan tujuan untuk mengurangi kemungkinan dan kemudahan dari eksploitasi dan kompromi.
 3. Manajemen Akses
Mencegah akses yang tidak sah dan fisik yang tidak tepat dan logis untuk, penggunaan, dan administrasi perangkat IoT oleh orang-orang, proses, dan perangkatar komputasi lainnya.
 4. Deteksi Insiden keamanan perangkat
Memantau dan menganalisis aktivitas perangkat IoT untuk tanda-tanda dari kejadian yang melibatkan keamanan perangkat.
- b. Area mitigasi risiko untuk tujuan 2, melindungi keamanan data.
 1. Proteksi data
Cegah akses dan gangguan pada data yang disimpan atau dalam transit yang mungkin mengekspos informasi sensitif atau memungkinkan manipulasi atau gangguan pada operasi perangkat IoT.
 2. Deteksi insiden keamanan data
Memantau dan menganalisis aktivitas perangkat IoT untuk tanda-tanda dari kejadian yang melibatkan keamanan data.
- c. Area mitigasi risiko untuk tujuan 3, melindungi privasi individu.
 1. Manajemen arus informasi
Pertahankan peta siklus hidup informasi PII yang akurat dan terbaru, yang mencakup jenis tindakan data, komponen PII yang diproses oleh tindakan data, pihak yang melakukan pemrosesan, dan faktor kontekstual tambahan apa pun yang relevan dengan pemrosesan yang akan digunakan untuk tujuan privasi manajemen risiko.

2. Manajemen izin pemrosesan PII
Mengelola izin untuk pemrosesan PII untuk mencegah pemrosesan PII tidak diizinkan.
3. Pengambilan Keputusan yang terinformasi
Memungkinkan individu untuk memahami dampak pemrosesan dan interaksi PII dengan perangkat, berpartisipasi dalam pengambilan keputusan tentang pemrosesan atau interaksi PII, dan menyelesaikan masalah.
4. Manajemen data yang dipisahkan
Identifikasi pemrosesan PII yang sah dan menentukan bagaimana PII dapat diminimalisasi atau dipisahkan dari individu dan perangkat IoT.
5. Deteksi pelanggaran privasi
Memantau dan menganalisis aktivitas perangkat IoT untuk tanda-tanda dari kejadian yang melibatkan privasi individu.

2.2 Model Trias CIA (*Confidentiality, Integrity, Availability*)

Penelitian ini mengutamakan keamanan pada informasi data yang akan ditampilkan secara publik. Keamanan informasi merupakan hal yang krusial dan menjadi prioritas utama dalam penelitian ini. Ada tiga objektif pada keamanan informasi untuk memastikan keamanan pada data, yaitu *Confidentiality* atau Kerahasiaan, *Integrity* atau Integritas, dan *Availability* atau Ketersediaan.



Gambar 2.6 Model Trias CIA

Sumber: Rahma & Pratama (2020)

Menurut Rahma & Pratama (2020) pada Buku Keamanan Siber dan Informasi, menjelaskan bahwa model trias CIA dirancang untuk menjadi panduan dalam pembuatan kebijakan-kebijakan terkait keamanan siber dan informasi. Dalam keamanan informasi, aspek Kerahasiaan mencakup aturan yang diperlukan untuk membatasi akses ke informasi, aspek Integritas menjamin bahwa informasi yang disajikan akurat dan dapat dipercaya, dan aspek Ketersediaan menjamin bahwa informasi tersedia untuk semua pihak yang memiliki otoritas kapan pun informasi tersebut dibutuhkan.

Aspek kerahasiaan berkaitan erat dengan privasi. Secara umum, aspek kerahasiaan merupakan Upaya pencegahan dari penyerangan berupa akses dan penyalahgunaan informasi sensitif dari orang lain yang tidak memiliki otoritas. Metode-metode yang paling umum digunakan untuk menjaga aspek kerahasiaan ini adalah kriptografi dan autentikasi seperti menggunakan *username* dan *password* ataupun menggunakan biometrik seperti sidik jari, retina, atau pengenalan wajah.

Aspek integritas merupakan perhatian pada konsistensi, akurasi, dan kepercayaan pada data dan informasi selama siklus hidupnya, sejak data diciptakan atau dihasilkan, disimpan dan disajikan atau ditransmisikan, hingga dihilangkan atau dihapus. Metode-metode yang umum digunakan adalah kontrol pengguna (*access control*) dan verifikasi integritas dengan menggunakan Teknik *checksum*. Teknik kontrol pengguna mengatur entitas yang dapat melihat, menggunakan, dan melakukan modifikasi terhadap sumber daya di sebuah sistem.

Aspek ketersediaan merupakan pengelolaan komponen sistem siber, seperti perangkat lunak, perangkat keras, dan infrastruktur jaringan. Begitu juga dengan hal-hal terkait pencadangan, peningkatan, dan pembaharuan, terutama kewaspadaan terhadap isu keamanan yang harus menjadi prioritas utama dalam aspek ini. Di sisi lain, hal-hal seperti redundansi, pencadangan saat terjadi kegagalan, dan rencana pemulihan setelah bencana. Pada infrastruktur jaringan, kecepatan internet yang memadai juga perlu menjadi perhatian dalam mencegah terjadinya kemacetan. Pengamanan fisik dari tindak kejahatan ataupun bencana merupakan hal yang perlu dilakukan.

2.3 Teknik Keamanan Trias CIA berdasarkan Relevansinya dengan Lapisan IoT

Abdullah et al. (2019) dalam artikelnya mengemukakan beberapa teknik keamanan dalam IoT yang diperlukan untuk tindakan mitigasi atau penanggulangan pada kerentanan penyerangan. Akan tetapi, teknik keamanan yang digunakan mengacu pada perlindungan terhadap lapisan penyusun IoT seperti *Perception Layer*, *Network Layer*, dan *Application*

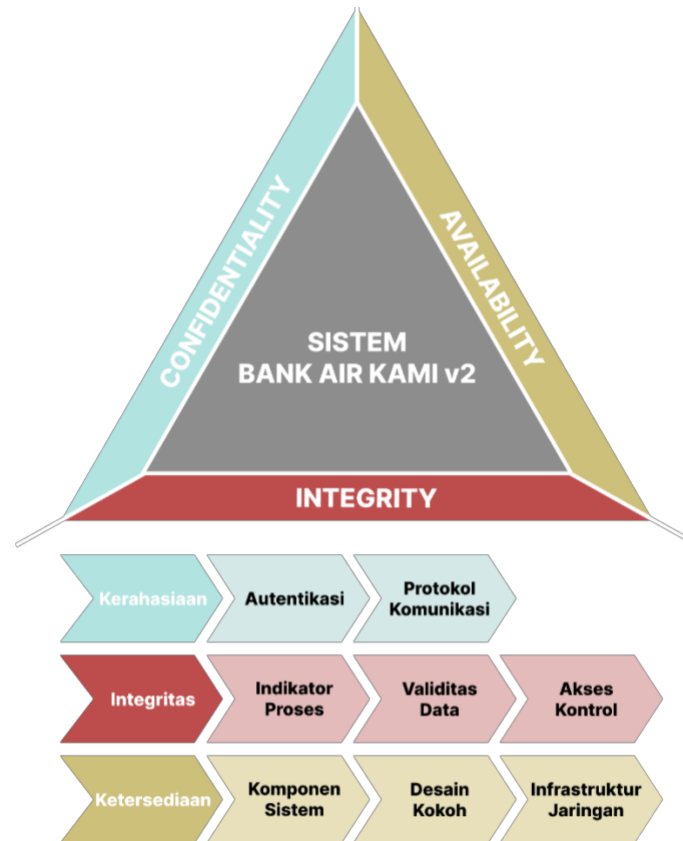
Layer. Namun, ketiga lapisan tersebut berkaitan erat dengan aspek-aspek trias CIA yang digunakan sebagai acuan utama pada penelitian ini. Hal ini dapat menjadi acuan dalam menentukan fitur-fitur untuk desain keamanan. Rincian mengenai teknik keamanan berdasarkan lapisan IoT dan relevansinya dengan trias CIA terdapat pada Tabel 2.1.

Tabel 2.1 Teknik Keamanan berdasarkan Lapisan IoT

Lapisan	Teknik Keamanan	Kerentanan Serangan	Relevansi
<i>Perception Layer</i>	- Algoritma <i>Hash</i> - Mekanisme Enkripsi - Asesmen Risiko - Autentikasi	- <i>Eavesdropping</i> - <i>Spoofing</i>	Aspek Kerahasiaan (<i>Confidentiality</i>)
<i>Network Layer</i>	- Integritas Data - Enkripsi <i>Point-to-Point</i> - Algoritma <i>Hash</i>	- Serangan <i>Denial of Service</i> (DoS) - Serangan <i>Man-in-the-Middle</i> - Serangan <i>Data Transit</i>	Aspek Integritas (<i>Integrity</i>)
<i>Application Layer</i>	- <i>Firewalls</i> - Deteksi Intrusi	- <i>Sniffing</i> - Serangan <i>Distributed Denial of Service</i> (DDoS)	Aspek Ketersediaan (<i>Availability</i>)

2.4 Domain Model CIA pada Keamanan Siber Bank Air Kami v2

Sistem Bank Air Kami v2 menggunakan pendekatan Trias CIA sebagai panduan untuk kebijakan-kebijakan terkait keamanan siber dan informasi. Memprioritaskan transmisi data yang optimal, terjaga keasliannya, dan ketersediaannya, serta kokoh secara keseluruhan dari segi tindakannya pada perangkat lunak, perangkat keras, dan telekomunikasi. Pada penelitian ini, komposisi dari setiap aspek akan menyesuaikan dengan tindakan atau upaya yang dibutuhkan dari setiap aspek. Di antaranya:



Gambar 2.7 Domain Trias CIA pada Bank Air Kami v2

2.4.1 Aspek Kerahasiaan (*Confidentiality*)

Bank Air Kami v2 menghasilkan informasi terkait kualitas atau kondisi air berupa suhu, keasaman (pH), kekeruhan air, dan konduktivitas air. Data-data tersebut bukan merupakan data sensitif atau rahasia bagi publik. Hal yang perlu dilakukan pada aspek ini adalah memastikan proses transmisi yang rahasia, sehingga mencegah adanya kemungkinan serangan melalui jaringan yang tersedia seperti *Eavesdropping* ataupun serangan yang mampu mengintervensi transmisi data. Beberapa upaya seperti menggunakan kredensial *username* dan *password*, *Secure Socket Layer* (SSL), dan *Transport Layer Security* (TLS) perlu dilengkapi pada protokol transmisi data.

2.4.2 Aspek Integritas (*Integrity*)

Aspek integritas pada konteks Bank Air Kami v2 adalah kepercayaan terhadap kondisi kualitas air untuk disajikan kepada masyarakat. Upaya untuk memastikan kepercayaan terhadap data adalah dengan menjaga integritas data. Salah satu cara untuk memastikan keamanan pada data yang berkaitan dengan integritas, autentikasi, dan non-repudiasi adalah dengan mengimplementasi kriptografi. Proses dalam mengimplementasi hal tersebut adalah

enkripsi, dan salah satu fungsi kriptografi adalah fungsi *hash* (Rachmawati et al., 2018). Untuk memastikan kualitas air yang disajikan pada masyarakat adalah kondisi yang sebenarnya dan dapat dipercaya, maka implementasi kriptografi dengan fungsi *hash* perlu dilakukan untuk memastikan kepercayaan terhadap data dan melakukan verifikasi terhadap keamanan, sehingga serangan seperti modifikasi data yang dilakukan oleh seseorang yang tidak berotoritas atau *Data Tampering* dapat dicegah.

Mengatur akses terhadap kontrol sistem memungkinkan untuk mempersempit ruang kerja, sehingga akan mengurangi kemungkinan entitas yang tidak berotoritas masuk kedalam dan mengintervensi sistem pada transmisi data ataupun kinerja sistem. Kontrol akses dapat berupa *platform* atau lingkungan untuk mengatur perangkat yang terhubung kepada entitas satu sama lain dengan kontroler yang hanya bisa diakses oleh pengembang atau *stakeholders* yang memiliki otoritas.

2.4.3 Aspek Kerahasiaan (*Availability*)

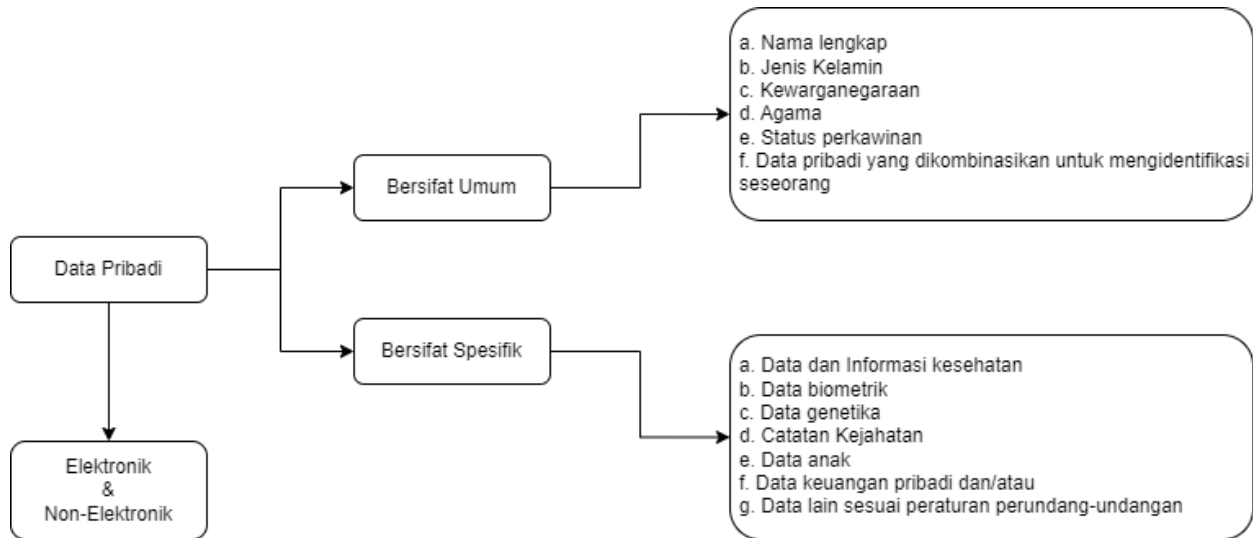
Dalam upaya memastikan ketersediaan terhadap data yang dihasilkan oleh sistem Bank Air Kami v2, perlu adanya tindakan seperti *Intrusion Detection System* (IDS) untuk mendeteksi adanya intrusi pada komputer yang dipandang sebagai tindakan yang berupaya membahayakan integritas, kerahasiaan, atau ketersediaan sumber daya (Bieerman et al., 2001). Dengan menerapkan *Intrusion Detection System* (IDS) pada Bank Air Kami v2, sistem mampu memantau adanya percobaan serangan atau tindakan seperti *Port Scanning* dan *Distributed Denial of Service* (DDoS) pada jaringan yang tersedia untuk dilakukan mitigasi atau penanganan terhadap serangan. Ketersediaan data kualitas air merupakan hal yang krusial dalam konteks pemantauan kondisi air.

Komponen-komponen untuk menunjang kinerja sistem diterapkan pada padanan yang tepat sesuai dengan fungsinya masing-masing. Akan tetapi, ketersediaan terhadap data belum terpenuhi pada beberapa faktor seperti pemantauan proses transmisi untuk dilakukan *restart* pada sistem, memilih *server* MQTT yang tepat untuk mencegah kelebihan beban, dan menguji MQTT dengan *load testing* untuk menguji pesan pada *broker*. Tidak hanya itu, pemulihan bencana seperti merencanakan pencadangan pada MQTT dan strategi pemulihan lainnya perlu dilakukan agar menjaga ketersediaan sistem.

2.5 Hukum Indonesia terhadap Perlindungan Data

Meningkatnya perkembangan teknologi pada sektor-sektor tertentu mampu menunjang aktivitas yang lebih efektif. Era digital membawa perubahan yang signifikan terhadap perilaku manusia terhadap segala hal, khususnya terhadap konsumsi informasi sebagai faktor utama untuk mengikuti perkembangan era. Namun, tidak luputnya kejahatan siber atau *cybercrime* yang terjadi sebagai sisi lain dari keuntungan era digital, terutama pada peretasan, penyalahgunaan, pengumpulan, dan penyebaran terhadap informasi sensitif yang mampu menyebabkan kerugian bagi beberapa aspek, hal ini disebut pelanggaran privasi pada data pribadi. Maka dari itu, pemerintah bersinergi dalam mengamankan data pribadi (Nisa et al., 2023).

Dewan Perwakilan Rakyat Republik Indonesia (DPR RI) mengesahkan Undang-Undang Perlindungan Data Pribadi (UU PDP) sebagai instrumen legal yang bertujuan untuk melindungi setiap data pribadi individu dari potensi penyalahgunaan data oleh pihak yang tidak memiliki otoritas (Bintarawati, 2024). UU PDP juga merupakan manifestasi dari UUD 1945 pasal 28G paragraf 1 tentang perlindungan hak asasi manusia. Ruang lingkup penerapan UU PDP bersifat ekstra teritorial mengacu pada lokasi kegiatan dan akibat dari kegiatan tersebut, selama berkaitan dengan wilayah Indonesia dan/atau warga negara Indonesia. Tujuan dari peran UU PDP itu adalah data pribadi, terminologi data pribadi didefinisikan sebagai data individu yang diidentifikasi atau dapat diidentifikasi secara individu atau digabungkan dengan informasi lain, baik secara langsung maupun tidak langsung melalui sistem elektronik atau non elektronik (Yuniarti, 2022). Pasal 3 UU PDP mencantumkan klasifikasi data pribadi yang bertujuan untuk jenis data yang dapat diproses hukum. Berikut klasifikasi data pribadi berdasarkan UU PDP pasal 3:



Gambar 2.8 Klasifikasi Data Pribadi Berdasarkan UU PDP Pasal 3

Sumber: Yuniarti (2022)

UU PDP merupakan hasil adaptasi dari regulasi Uni Eropa tentang *General Data Protection Regulation* (EU-GDPR) atau regulasi perlindungan data umum. Regulasi EU-GDPR disahkan oleh Uni Eropa yang bertujuan untuk melindungi data pribadi dan privasi individu yang tinggal di Uni Eropa. EU-GDPR juga berbasis prinsip hak asasi manusia dan dipertimbangkan sebagai tolok ukur internasional terkait perlindungan data pribadi (Simbolon & Juwono, 2022).

2.6 Regulasi terhadap Data Sensitif dan Relevansinya pada Bank Air Kami v2

Simbolon & Juwono (2022) mengemukakan perbedaan antara UU PDP dengan EU-GDPR dalam regulasi pemrosesan data pribadi. Secara garis besar perbedaan kedua regulasi terkait pandangan terhadap data pribadi atau sensitif adalah keabsenan proses asesmen terhadap determinasi risiko pada data pribadi, sehingga pada pemrosesan data pribadi tidak dibedakan perilaku penanganannya untuk semua data. Pada EU-GDPR terjadi proses asesmen yang dinamakan *Data Protection Impact Assessment* (DPIA) yang bertujuan untuk menentukan data pribadi yang bersifat sensitive atau tinggi risiko, yaitu data terkait preferensi politik seseorang, agama, etnis, ras, orientasi seksual, dan Kesehatan. UU PDP tidak mendefinisikan data pribadi berdasarkan risiko yang rendah, sedang, atau tinggi (sensitif).

Maka dari itu, EU-GDPR mendefinisikan data yang tinggi risiko atau data sensitif, yaitu data yang mengandung preferensi politik seseorang, agama, etnis, ras, orientasi seksual, dan Kesehatan, sementara UU PDP mendefinisikan data pribadi yang tertera pada Gambar 2.8

Klasifikasi Data Pribadi Berdasarkan UU PDP Pasal 3 sebagai data pribadi yang dilindungi secara hukum. Dalam konteks Bank Air Kami v2, data yang diproduksi dan dikelola merupakan data kualitas air pada sumber mata air di kawasan Masyarakat, sehingga data tersebut tidak termasuk ke dalam kategori data pribadi menurut UU PDP dan data sensitive menurut EU-GDPR. Akan tetapi, setiap nilai dari UU PDP ataupun EU-GDPR menjadi landasan utama dalam strategi keamanan siber yang memenuhi kebutuhan pemenuhan aspek-aspek trias CIA yang berfokus pada penyajian data.

Beberapa nilai yang terkandung pada pasal-pasal UU PDP yang relevan dengan penerapan trias CIA pada sistem Bank Air Kami V2 adalah sebagai berikut:

Tabel 2.2 Pasal-pasal UU PDP

Relevansi	Pasal
Penerapan Keamanan Siber secara umum dan Infrastruktur	Pasal 22 Menjelaskan tentang pemasangan alat pemroses data visual di tempat umum untuk tujuan keamanan atau pencegahan bencana, perlu adanya informasi yang ditampilkan bahwa tempat tersebut telah dipasang alat pemroses data, dan tidak digunakan untuk mengidentifikasi seseorang.
	Pasal 27 Menjelaskan tentang diperlukannya tindakan atau upaya untuk melindungi data pribadi yang diproses, dengan menetapkan langkah teknis operasional untuk melindungi data dan menentukan tingkat keamanan sesuai sifat dan risiko dari data pribadi yang dilindungi.
Kerahasiaan (<i>Confidentiality</i>)	Pasal 21 Menjelaskan tentang menjaga kerahasiaan pada pemrosesan data pribadi.
	Pasal 29 Menjelaskan tentang keharusan menjaga pemrosesan data pribadi agar tidak bisa diintervensi oleh orang lain yang tidak sah.
Integritas (<i>Integrity</i>)	Pasal 28 Menjelaskan tentang pengawasan data pribadi dari pihak yang menangani pemrosesan data.

	Pasal 35 Menjelaskan tentang diperlukannya verifikasi terhadap akurasi, kelengkapan, dan konsistensi data pribadi.
Ketersediaan (<i>Availability</i>)	Pasal 30 Menjelaskan tentang diperlukannya upaya seperti sistem elektronik yang andal untuk mencegah akses data pribadi yang tidak sah.

2.7 Protokol *Message Queueing Telemetry Transport* (MQTT)

Protokol komunikasi adalah hal yang sangat penting pada sistem heterogen, yang menentukan kerangka kerja yang dapat diterima untuk interaksi perangkat. Beberapa protokol pada lapisan aplikasi telah diciptakan untuk komunikasi antara sistem IoT dan *machine-to-machine* (M2M), yaitu *Message Queueing Telemetry Transport* (MQTT), *Advanced Message Queueing Protocol* (AMQP), dan *Constrained Application Protocol* (CoAP). Pemahaman yang dalam terkait sistem IoT dan kebutuhan komunikasi data merupakan hal yang penting untuk menentukan protokol yang tepat untuk sistem (Moraes et al., 2019). Maka dari itu, perlu adanya studi literatur untuk menentukan protokol yang cocok diterapkan pada sistem Bank Air Kami v2.

Moraes et al. (2019) pada penelitiannya menyajikan beberapa literatur yang membahas tentang asesmen terkait protokol yang mampu memenuhi kebutuhan mereka. Berikut komparasi studi yang disajikan:

Tabel 2.3 Komparasi Literatur pada Protokol Komunikasi

Penulis	Metode Penelitian	Hasil
Luzuriaga et al. (2015)	Evaluasi terhadap MQTT dan AMQP menggunakan eksperimen jaringan.	AMQP lebih terukur.
Chen & Kunz, (2016)	Melakukan komparasi kualitatif dengan menggunakan emulator jaringan, eksperimen terhadap MQTT, CoAP, <i>data distribution service</i> (DDS) dan UDP.	MQTT dan DDS menyediakan transmisi yang dapat diandalkan.
Collina et al. (2014)	Analisis kuantitatif untuk beberapa kondisi lalu lintas jaringan menggunakan emulator dengan protokol MQTT dan CoAP.	MQTT menyediakan <i>delay</i> terkecil, sementara CoAP

		lebih baik pada lalu lintas yang intens.
Yokotani & Sasaki (2016)	Evaluasi HTTP dan MQTT untuk sistem IoT yang terdapat pada jaringan <i>overload</i> .	MQTT lebih baik.
Moraes et al. (2019)	Eksperimen terhadap protokol MQTT, AMQP dan CoAP menggunakan beberapa metode kuantitatif pada lalu lintas jaringan.	CoAP adalah yang terbaik disusul oleh MQTT pada eksperimen. CoAP lebih baik pada sumber daya jaringan yang lebih sedikit daripada protokol lain.

Beberapa literatur mengatakan bahwa protokol MQTT menyediakan performa yang baik dan serbaguna. MQTT telah digunakan sebagai bagian dari banyak gawai IoT dan sistem pengiriman pesan instan karena dimaksudkan untuk bekerja pada mesin berdaya rendah sebagai protokol ringan (Soni & Makwana, 2017). MQTT diakui sebagai standar komunikasi pada IoT secara *de facto* dan mampu merealisasikan transmisi terenkripsi melalui *Secure Socket Layer* (SSL) atau *Transport Layer Security* (TLS) (Chen et al., 2020).

Atmoko et al. (2017) mengatakan mekanisme protokol MQTT menggunakan *publish/subscribe* dan berjalan di atas TCP/IP. Mekanisme *publish/subscribe* bekerja dengan cara pengirim data atau *publisher* dan penerima data atau *client* melalui kontrol perintah *subscribe* untuk menerima data tidak mengetahui satu sama lain karena terdapat *broker* di antara keduanya. Protokol ini secara spesifik didesain untuk komunikasi *machine-to-machine* (M2M). Memiliki kemampuan untuk transmisi berbagai macam tipe data seperti data biner, teks, *XML*, atau *JSON*. MQTT memiliki 14 tipe kontrol sinyal, tetapi hanya ada empat sinyal utama yang digunakan langsung oleh *client*, yaitu *publish*, *subscribe*, *unsubscribe*, dan *connect*. Sementara sinyal-sinyal lain hanya bagian dari mekanisme *publish* atau *subscribe*. Berikut penjelasan setiap tipe kontrol sinyal:

- Subscribe*, pesan yang digunakan *client* untuk berlangganan pada topik yang spesifik.
- Publish*, pesan yang merepresentasikan terbitan baru atau terpisah.
- Unsubscribe*, pesan yang digunakan *client* untuk berhenti berlangganan pada topik yang spesifik.
- Connect*, permintaan *client* untuk terhubung ke *server*.

Tipikal arsitektur MQTT dibagi menjadi dua komponen utama seperti tertera pada gambar. Berikut penjelasan setiap komponen:

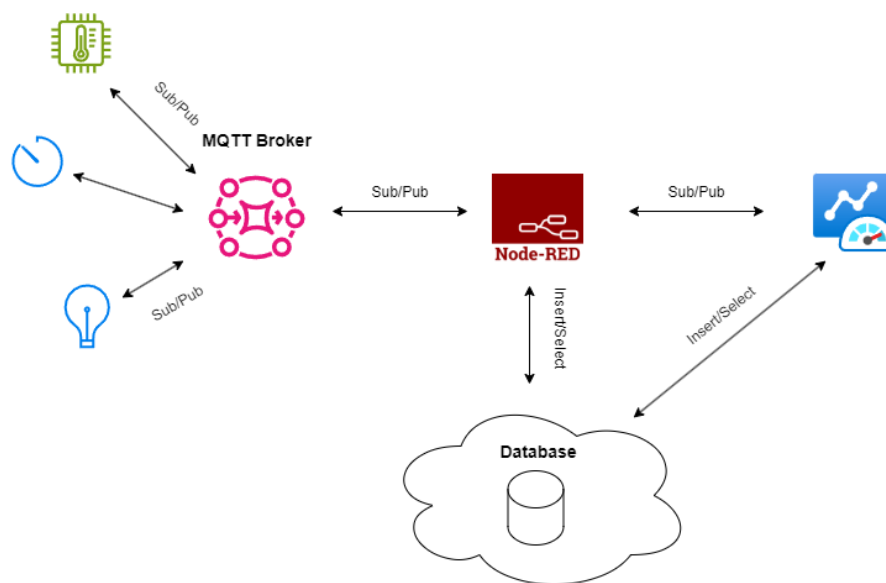
- a. *Client*: *Client* bisa berupa *publisher* atau *subscriber* dan selalu menetapkan koneksi jaringan kepada *server* atau *broker*. Ini bisa melakukan beberapa hal:
 1. *Publish* pesan untuk pengguna yang tertarik.
 2. *Subscribe* pada subjek yang tertarik untuk menerima pesan.
 3. *Unsubscribe* untuk mengekstrak dari subjek.
 4. Melepaskan diri dari *broker*.
- b. *Broker*: *Broker* mengontrol distribusi informasi dan terutama bertanggung jawab untuk menerima semua pesan dari *publisher*, menyaringnya, memutuskan siapa yang tertarik terhadap pesan tersebut dan mengirimnya kepada *client* yang telah berlangganan. *Broker* bisa melakukan beberapa hal:
 1. Menerima permintaan dari *client*
 2. Menerima pesan yang telah di-*publish* oleh pengguna
 3. Memroses beberapa permintaan seperti *subscribe* dan *unsubscribe* dari pengguna
 4. Setelah menerima pesan dari *publisher*, mengirimnya kepada pengguna yang tertarik.

MQTT dilengkapi dengan kemampuan untuk autentikasi menggunakan *username* dan *password*. Keamanan pada MQTT tergantung pada kebutuhan dan pemilihan *broker*. Beberapa *broker* menyediakan fitur keamanan berbasis TLS, tetapi tidak semua *broker* mampu menjaga performa yang stabil setelah menggunakannya (Soni & Makwana, 2017). Maka dari itu, perlu memerhatikan kebutuhan sistem secara menyeluruh untuk mengaplikasikan langkah yang tepat, apakah memprioritaskan pada keamanan yang berakibat pada performa yang berkurang atau memprioritaskan performa yang stabil untuk proses transmisi data dengan cepat yang tidak terlalu mementingkan fitur keamanan.

Mosquitto adalah salah satu *broker* MQTT yang dapat digunakan dalam semua situasi. Mosquitto dapat digunakan pada pengiriman pesan yang ringan, terutama pada perangkat yang memiliki keterbatasan terhadap daya. Dua literatur tentang penggunaan Mosquitto mengatakan bahwa *broker* tersebut mampu digunakan pada *Smart City Services*, dan pengembangan pada sistem pemantauan lingkungan (Light, 2017).

2.8 Platform Kontroler Node-RED

Menurut Ferencz et al. (2020), Node-RED adalah sebuah lingkungan untuk pengembangan sistem IoT berbasis *Node.js*, lebih spesifik lagi adalah sebuah lingkungan pemrograman virtual berbasis process-based yang menciptakan aliran data dari sensor kepada cloud dengan menghubungkan antara perangkat lunak dan perangkat keras. Node-RED dapat digunakan untuk membaca proses data, dengan demikian memfasilitasi pemrosesan data. Mampu mengatur algoritma data dan mengirimkannya kepada sistem yang lebih tinggi seperti *server SQL*, sistem manajemen perusahaan, pengumpul data pusat, dan layanan berbasis awan, lalu menampilkannya dalam waktu singkat. Node-RED mengandung nodes yang menyediakan berbagai fungsi seperti *MQTT broker*, *debug*, Raspberry Pi GPIO, TCP, dll. Menciptakan proses yang disimpan pada objek *JSON*. Node-RED terdiri dari 3 elemen dasar, yaitu panel *node*, panel aliran, dan *Info* atau *Debug*.



Gambar 2.9 Skema Node-RED

Sumber: Ferencz et al. (2020)

Secara garis besar Node-RED menyediakan lingkungan untuk pengembangan aliran data yang berasal dari *broker* sebagai sentralisasi data dari sensor, lalu kontrol data dilakukan sebelum mengirim data kepada *database* berbasis awan untuk ditampilkan pada *web dashboard*.

2.9 Intrusion Detection System (IDS)

Menurut Wagh et al. (2013), *Intrusion Detection System* adalah teknik keamanan yang mencoba mendeteksi berbagai serangan, mereka adalah serangkaian teknik yang digunakan untuk mendeteksi aktivitas mencurigakan. IDS dapat dilakukan oleh dua metode dasar seperti *Signature-Basesd* dan *Anomaly-Based*. Metode *Signature-Based* dikenal juga sebagai *misuse detection*, mereka menandai jenis serangan yang diketahui berdasarkan hasil identifikasi, mereka mampu mendeteksi banyak atau semua pola serangan yang diketahui. Akan tetapi, kelemahan *Signature-Based Detection System* adalah ketidakmampuan mengidentifikasi jenis serangan baru atau variasi serangan yang diketahui. Sementara menurut Ashoor & Gore (2010), *Anomaly-Based Detection System* bergantung pada klasifikasi jaringan dari normal ke ganjil, klasifikasi ini berdasarkan pada peraturan atau heuristik daripada pola atau tanda. Kemampuan dari deteksi *Anomaly-Based* tidak seperti *Signature-Based*, karena dia bisa mendeteksi ancaman yang belum diketahui sebelumnya.

IDS tidak hanya ditaksonomi oleh dua metode, tetapi dapat diklasifikasi berdasarkan tempat di mana dia dijalankan, seperti *Host based IDS* (HIDS), *Network based IDS* (NIDS), dan *Hybrid based IDS*.

a. *Host based IDS*

Tipe ini ditempatkan pada satu perangkat seperti *server* atau *workstation*. Data dianalisis secara lokal kepada mesin dan mengumpulkan data dari sumber yang berbeda. HIDS dapat digunakan oleh *Anomaly-based IDS* ataupun *Signature-based IDS*.

b. *Network based IDS*

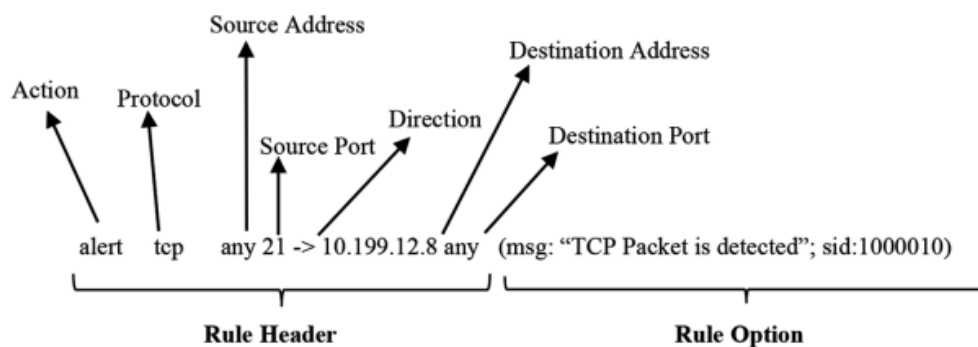
NIDS dikerahkan pada poin strategis pada infrastruktur jaringan. NIDS dapat menangkap dan menganalisis data untuk mendeteksi serangan yang diketahui dengan melakukan komparasi pola atau tanda dari basis data atau deteksi aktivitas ilegal dengan melakukan pemindaian lalu lintas untuk aktivitas ganjil. NIDS juga dikatakan sebagai *Packet-sniffers*, karena menangkap paket yang melewati medium komunikasi.

c. *Hybrid based IDS*

Manajemen dan peringatan dari perangkat *Intrusion Detection* jaringan dan *host*, memberikan efektivitas terhadap lapisan keamanan dan pelengkap logis untuk NIDS dan HIDS.

2.10 Snort

Salah satu contoh IDS berbasis tanda adalah Snort. Menurut (Kumar & Prakash Sangwan, 2012), Snort adalah *Network Intrusion Detection System* (NIDS) yang mampu menganalisis lalu lintas dan aliran data pada jaringan secara *real-time*. Snort mampu melakukan pengecekan pada protokol analisis dan mampu mendeteksi berbagai jenis serangan. Pada Snort yang bertindak sebagai NIDS, pada dasarnya mengecek paket terhadap peraturan yang ditulis oleh pengguna, dikenal sebagai *Snort Rules*. *Snort Rules* dapat ditulis pada bahasa apapun, strukturnya mudah dipahami dan dapat dimodifikasi. *Snort Rules* dirumuskan oleh pengguna dengan dua struktur, yaitu *rule header* dan *rule option*. Berikut contoh penulisan *Snort Rules* pada Gambar 2.10.



Gambar 2.10 Contoh Penulisan Snort Rules

Sumber: Chanthakoummene et al. (2016)

Snort memiliki tiga mode operasional utama, yaitu *Packet Sniffer*, *Packer Logger*, dan *Network Intrusion Detection System* (NIDS). Ketiga mode tersebut dapat dinotasikan oleh *Snort Rules* yang digunakan sesuai kebutuhan pengguna.

a. *Packet Sniffer*

Berfungsi untuk mengumpulkan data dan menampilkan lalu lintas jaringan. *Snort Rules* dalam mode *Packet Sniffer* dinotasikan pada Gambar 2.11

```
snort -d -e -v
# -v: menempatkan Snort pada mode Packet Sniffing (hanya header TCP)
# -d: memuat semua lapisan header jaringan (TCP, UDP, dan ICMP)
# -e: memuat header lapisan data link
```

Gambar 2.11 *Snort Rules* mode *Packet Sniffing*

b. *Packet Logger*

Berfungsi untuk mengumpulkan data sebagai *file*, menyortirnya pada direktori atau menyimpan data sebagai data biner. *Snort Rules* dalam mode *Packet Logger* dinotasikan pada Gambar 2.12.

```
snort -dev -l {logging-directory} -h {home-subnet-slash-notation}

#contoh menuliskan rule pada direktori:

snort -dev -l /var/adm/snort/logs -h 192.20.14.0/24
```

Gambar 2.12 *Snort Rules* mode *Packet Logging*

c. *Network Intrusion Detection System (NIDS)*

Berfungsi untuk menganalisis paket dan menyamakan jaringan terhadap tanda. *Snort Rules* dalam mode *Packet Logger* dinotasikan pada Gambar 2.13.

```
#untuk membuat Snort pada mode IDS, tambahkan satu hal pada rule
Packet Logging: file konfigurasi

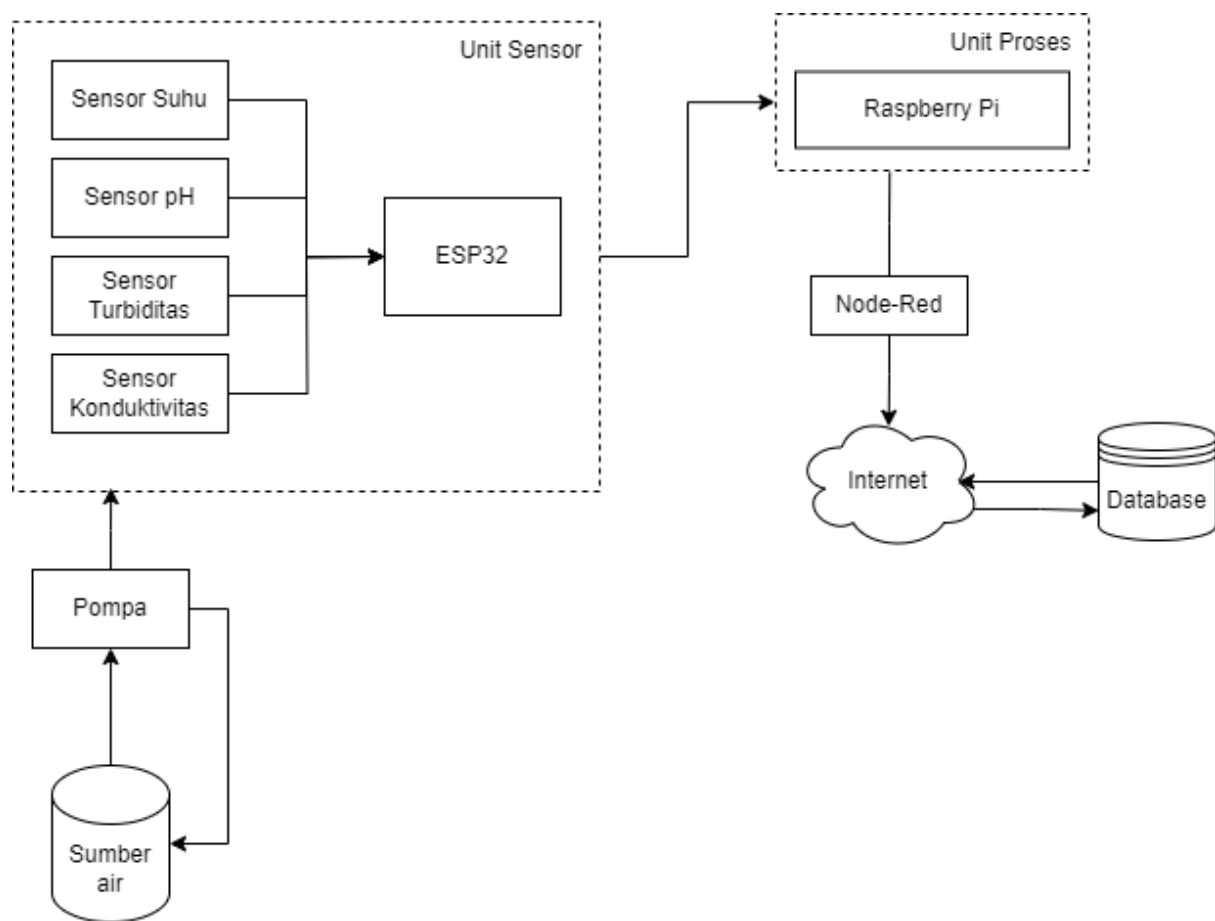
snort -dev -l /var/adm/snort/logs -h 192.20.14.0/24 -c
/root/mysnort.conf
```

Gambar 2.13 *Snort Rules* mode NIDS

BAB III METODOLOGI PENELITIAN

3.1 Identifikasi Kebutuhan Keamanan Bank Air Kami v2

Dalam menciptakan desain keamanan yang efektif pada Bank Air Kami v2, identifikasi kebutuhan keamanan diperlukan untuk kerentanan yang berpotensi menyerang sistem. Perencanaan arsitektur Bank Air Kami v2 yang dilakukan oleh Kevin Fernando Atalaric dengan menciptakan sistem yang kokoh dan fleksibel untuk dilakukan peningkatan.



Gambar 3.1 Alur komunikasi data Bank Air Kami v2

Transmisi data terjadi ketika sensor telah mengukur kualitas air dan mengirim data ke mikrokontroler ESP32, setelah itu Raspberry Pi akan menerima data dan meneruskannya dengan MQTT kepada Node-RED untuk dikoneksikan ke database. *Broker* MQTT berperan untuk komunikasi data antara ESP32 dengan Node-RED yang dikontrol oleh Raspberry Pi, sehingga peran Raspberry Pi adalah untuk sebagai *server* MQTT dan komponen keamanan.

Perangkat IoT rentan terhadap serangan jaringan, sehingga menjaga mekanisme autentikasi yang efisien adalah solusi untuk menjaga prinsip-prinsip trias CIA (*Confidentiality, Integrity, Availability*) pada jaringan IoT (Mangla et al., 2022). Jaringan yang digunakan adalah WiFi publik yang digunakan oleh masyarakat sekitar. Maka dari itu, strategi keamanan ditekankan pada pengamanan jaringan dari berbagai jenis serangan.

Jenis-jenis serangan yang berpotensi menyerang melalui jaringan dan mengintervensi komunikasi data berdasarkan aspek-aspek trias CIA adalah sebagai berikut:

- a. Kerahasiaan (*Confidentiality*)
 1. *Eavesdropping* atau *Packet Sniffing*: komunikasi pada *broker* MQTT pada dasarnya tidak dilengkapi keamanan berlapis pada komunikasinya, penyerang mampu memotong dan menguping komunikasi dan mendapatkan akses untuk data sensitif. Hal ini dapat terjadi juga pada komunikasi antara ESP32 dan Raspberry Pi.
 2. Serangan *Man-in-the-middle* (MiTM): penyerang mampu menempatkan diri antara ESP32 dan Raspberry Pi untuk memotong komunikasi dan memodifikasi lalu lintas data.
- b. Integritas (*Integrity*)
 1. *Replay Attacks*: penyerang dapat mendapatkan paket data pada komunikasi MQTT dan melakukan pengulangan pada pengiriman paket atau *replay* untuk memanipulasi sistem.
 2. *Data Tampering*: data kualitas air dapat dimodifikasi menjadi data yang tidak valid atau tidak dapat dipercaya melalui komunikasi antara ESP32 dan Raspberry Pi. Data yang tidak valid dapat dijadikan sebagai *hoax* untuk menggiring seseorang meyakini bahwa kualitas air di tempat tersebut tidak layak dikonsumsi.
- c. Ketersediaan (*Availability*)
 1. Serangan *Denial of Service* (DoS): hal ini berpotensi menyebabkan sistem tidak berfungsi dan berhenti mengirimkan data. Selain itu, Raspberry Pi yang digunakan pada Bank Air Kami v2 memiliki potensi untuk dijadikan *Zombie* yang dapat menyerang perangkat-perangkat yang terhubung dengan jaringan internet yang sama, sehingga tidak hanya merugikan sistem Bank Air Kami v2, tetapi juga dapat merugikan sekitar.
 2. Serangan fisik: hal-hal seperti pencurian komponen, kerusakan sensor, dan lain-lain, dapat terjadi ketika pada bilik sistem Bank Air Kami v2 tidak dilengkapi dengan kunci keamanan ganda.

Dengan identifikasi jenis-jenis serangan yang berpotensi mengganggu prinsip-prinsip trias CIA pada Bank Air Kami v2, desain keamanan akan dirancang untuk melakukan mitigasi atau pencegahan terhadap hal tersebut.

3.2 Rencana Desain Keamanan Data pada Sistem Bank Air Kami v2

Perancangan desain keamanan yang akan diimplementasi pada Bank Air Kami v2 berdasarkan dari identifikasi kebutuhan keamanan yang mengacu pada aspek-aspek trias CIA (*Confidentiality, Integrity, Availability*) untuk komunikasi data kualitas air yang akan ditampilkan pada web dashboard. Rencana desain terdiri dari parameter dan tindakan. Parameter ditentukan dari kebutuhan keamanan pada sistem berdasarkan infrastruktur dan sumber daya yang digunakan, sementara tindakan merupakan implementasi dari parameter yang ditentukan.

Tabel 3.1 Desain keamanan data pada Bank Air Kami v2

Aspek	Parameter	Tindakan
Kerahasiaan (<i>Confidentiality</i>)	Protokol komunikasi data dengan enkripsi dan autentikasi pengguna.	Menggunakan protokol MQTT dengan <i>Transport Layer Security</i> (TLS) sebagai standar protokol komunikasi data dengan <i>broker</i> Mosquitto yang dilengkapi dengan Sertifikat <i>Secure Socket Layer</i> (SSL).
Integritas (<i>Integrity</i>)	Akses kontrol pengelola sistem.	Menggunakan Node-RED sebagai platform untuk mengontrol perangkat atau entitas yang terhubung satu sama lain.
	Implementasi algoritma hash untuk setiap transmisi data	Menggunakan algoritma SHA-256 untuk melakukan hashing pada setiap pesan yang akan di- <i>publish</i> ke <i>broker</i>
Ketersediaan (<i>Availability</i>)	Implementasi <i>Network Intrusion Detection System</i> (NIDS) untuk mendeteksi intrusi pada jaringan.	Mengimplementasi <i>Network Intrusion Detection System</i> (NIDS) berbasis tanda menggunakan Snort

		untuk memantau adanya indikasi serangan.
	Mengimplementasi <i>Firewall</i> pada lalu lintas jaringan MQTT	Melindungi jaringan dengan ufw pada Raspberry Pi untuk mengelola dan menyaring lalu lintas jaringan untuk mencegah aktivitas jahat dan serangan <i>Distributed Denial of Service</i> (DDoS).

3.3 Tujuan Desain Keamanan

Dalam menciptakan desain keamanan, menentukan tujuan atau *goals* yang mengacu pada ketiga aspek trias CIA berdasarkan kebutuhan sistem seperti memantau proses transmisi data untuk memastikan sistem berjalan dengan baik, menentukan alur komunikasi yang efektif, dan merencanakan infrastruktur keamanan untuk menunjang sistem.

3.3.1 Tujuan Aspek Kerahasiaan (*Confidentiality*)

Kerahasiaan data bukan tujuan utama dalam Bank Air Kami v2, karena data yang dihasilkan bukan merupakan data sensitif yang berbahaya bagi sebagian pihak. Maka dari itu, aspek ini mengutamakan pada pengamanan komunikasi data pada MQTT *broker* dengan *Transport Layer Security* (TLS) untuk memastikan komunikasi terenkripsi dan tetap aman dari peretas. Berikut tujuan berdasarkan tindakan.

Tabel 3.2 Tujuan aspek kerahasiaan

Tindakan	Tujuan (<i>Goals</i>)
Menggunakan protokol MQTT dan <i>broker</i> Mosquitto yang dilengkapi dengan sertifikat <i>Secure Socket Layer</i> (SSL)	Memastikan komunikasi data antara <i>server</i> dan <i>client</i> terenkripsi untuk menjaga keamanan data dari peretas dengan metode <i>Packet Sniffing</i> atau <i>Eavesdropping</i> yang mampu melihat seluruh paket pada komunikasi jaringan.

3.3.2 Tujuan Aspek Integritas (*Integrity*)

Memastikan kepercayaan terhadap data merupakan hal yang penting. Maka pada aspek ini, difokuskan untuk melakukan akses kontrol pada pengelola sistem Bank Air Kami v2, sedangkan implementasi deteksi intrusi pada jaringan untuk mendeteksi adanya serangan yang dapat mengganggu kepercayaan data, hal ini berisikan dengan aspek ketersediaan. Berikut tujuan berdasarkan tindakan.

Tabel 3.3 Tujuan aspek integritas

Tindakan	Tujuan (<i>Goals</i>)
Platform Node-RED untuk kontrol data dan visualisasi yang dilengkapi dengan kontrol akses berupa autentikasi	Mempersempit ruang kerja bagi pengelola, sehingga mengurangi kemungkinan celah atau kerentanan pada sistem yang mampu merubah alur komunikasi pada Node-RED.
Implementasi algoritma hash SHA-256	Mengurangi kemungkinan adanya modifikasi data pada Bank Air Kami v2 untuk memastikan kepercayaan atas data.

3.3.3 Tujuan Aspek Ketersediaan (*Availability*)

Untuk mendeteksi serangan *Distributed Denial of Service* (DDoS) pada jaringan, maka implementasi deteksi intrusi merupakan hal yang penting. Aspek ini juga memfokuskan pada pemantauan *broker*, jaringan, dan komponen untuk memastikan sistem berjalan dan menghasilkan data. Berikut tujuan berdasarkan tindakan.

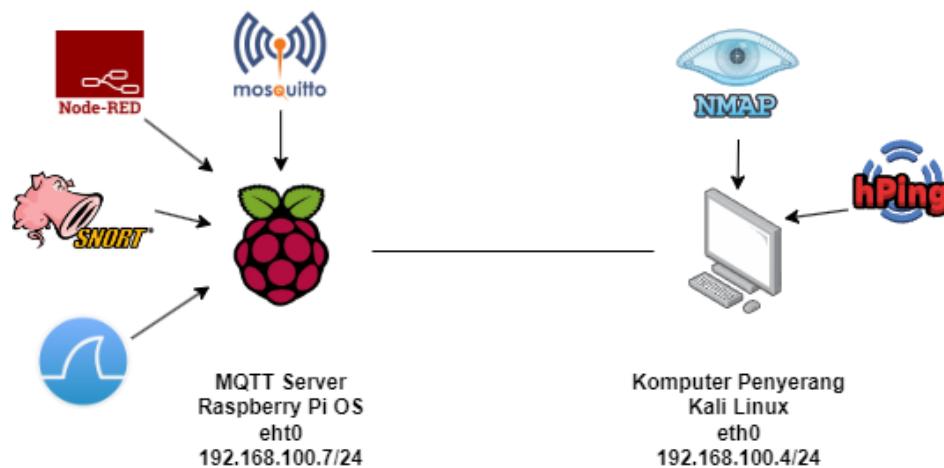
Tabel 3.4 Tujuan aspek ketersediaan

Tindakan	Tujuan (<i>Goals</i>)
Implementasi <i>Network Intrusion Detection System</i> (NIDS) untuk mendeteksi intrusi pada jaringan.	Mengimplementasi <i>Network Intrusion Detection System</i> (NIDS) berbasis tanda menggunakan Snort untuk memantau adanya indikasi serangan seperti <i>Distributed Denial of Service</i> (DDoS), dan <i>Port Scanning</i>
Implementasi <i>Firewall</i> pada lalu lintas jaringan MQTT	Mengantisipasi adanya serangan pada lalu lintas jaringan dengan mengimplementasi

	<i>rules</i> pada <i>ufw</i> untuk menyaring dan mengelola jaringan perangkat.
--	--

3.4 Tahapan Implementasi Desain Keamanan

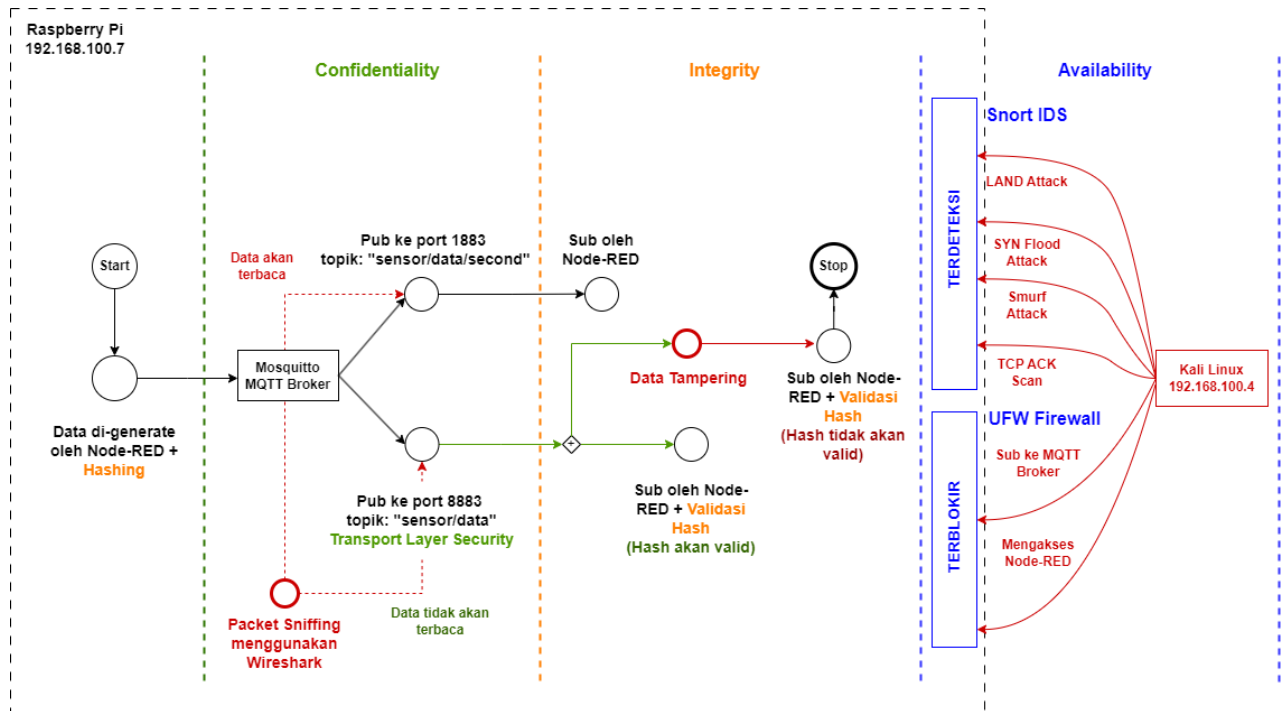
Implementasi bertujuan untuk menguji fungsionalitas dari keamanan desain yang telah dirancang sebelum benar-benar diimplementasi pada produk akhir. Implementasi desain keamanan akan dilakukan pada Oracle VirtualBox Machine GUI versi 7.0.18 untuk melakukan simulasi komunikasi data dan pengujian keamanan pada prinsip-prinsip trias CIA. *Virtual Machine* yang digunakan untuk implementasi protokol MQTT, Node-RED, dan Snort adalah Raspberry Pi OS Bullseye dengan Raspbian versi 11 (64-bit) sesuai dengan infrastruktur yang digunakan pada arsitektur Bank Air Kami v2, sedangkan untuk replikasi serangan menggunakan *Virtual Machine* Kali Linux dengan menyerang jaringan pada alamat jaringan yang digunakan pada Raspberry Pi OS.



Gambar 3.2 Topologi pengujian pada VirtualBox Machine

Gambar 3.2 menunjukkan *machine* yang digunakan pada topologi pengujian, Kali Linux dengan alamat IP 192.168.100.4 dengan *interface* eth0, dan Raspberry Pi OS dengan alamat IP 192.168.100.7 dengan *interface* eth0. Pengujian yang dilakukan pada VirtualBox Machine akan menggunakan jaringan *NAT Network* yang ditentukan untuk melakukan pengujian dengan nama “simulasibankairkami”. *Subnet* Alamat IP yang ditentukan adalah 192.168.100.0/24 untuk setiap *machine* yang digunakan untuk pengujian, sehingga seluruh *machine* mampu berkomunikasi dan melakukan simulasi serangan. Raspberry Pi OS akan dijadikan sebagai *server* MQTT menggunakan *broker* Mosquitto. Raspberry Pi OS akan dilengkapi dengan

Node-RED untuk melakukan *generate* data dan melakukan *hash*, Snort IDS untuk mendeteksi serangan dari jaringan, *Firewall* untuk membatasi jaringan, dan Wireshark sebagai alat untuk melihat komunikasi yang terjadi pada jaringan Raspberry Pi OS.



Gambar 3.3 Arsitektur Desain Keamanan

Gambar 3.3 menunjukkan arsitektur desain keamanan berupa alur komunikasi data, implementasi fitur-fitur keamanan berdasarkan trias CIA, dan pengujian serangan pada setiap aspek. Implementasi desain keamanan dan pengujian pada aspek kerahasiaan dan aspek integritas dilakukan di Raspberry Pi OS, sedangkan pengujian pada aspek ketersediaan dilakukan oleh Kali Linux yang menyerang Raspberry Pi OS. Skenario keamanan akan terjadi ketika transmisi data dimulai, data diperoleh dari hasil *generate* menggunakan Node-RED, lalu dikirimkan atau di-*publish* ke *broker* Mosquitto, kemudian diterima atau di-*subscribe* kembali oleh Node-RED dengan tujuan untuk melakukan simulasi transmisi data yang sesungguhnya. Berikut penjelasan mengenai rencana implementasi dan pengujian:

3.4.1 Rencana Implementasi MQTT

Raspberry Pi OS akan berperan sebagai penyedia layanan atau *server* MQTT untuk membangun jaringan komunikasi data antara sensor dengan Raspberry Pi. Untuk melakukan hal tersebut, dibutuhkan *broker* yang akan menjadi perantara kedua perangkat saling

berkomunikasi. *Broker* yang digunakan adalah Mosquitto dengan menentukan *port* yang akan digunakan. Berikut rincian mengenai *port* yang akan digunakan dan perannya pada penelitian ini:

- a. *Port* 1883. Merupakan *port* standar yang digunakan untuk komunikasi MQTT tanpa adanya fitur keamanan. *Port* 1883 sebagai pembanding antara *port* ini dengan *port* lainnya pada pengujian TLS menggunakan Wireshark. Pada dasarnya semua komunikasi yang terjadi pada *port* ini akan terekspos seperti isi pesan, paket, permintaan, dan lain-lain.
- b. *Port* 8883. Merupakan *port* yang akan dilengkapi dengan fitur keamanan TLS untuk mengenkripsi setiap komunikasi yang terjadi. Untuk menguji TLS pada *port* 8883, maka akan dilakukan pengujian TLS menggunakan Wireshark untuk melihat komunikasi yang terenkripsi, lalu melihat perbandingannya dengan *port* 1883. Hasil pengujian akan menunjukkan bahwa *port* 8883 tidak mengekspos isi pesan atau paket dari komunikasi terjadi, karena komunikasi telah dienkripsi atau dirahasiakan.

3.4.2 Rencana Implementasi Node-RED

Node-RED digunakan untuk melakukan simulasi komunikasi data dari sensor ke Raspberry Pi. Karena penelitian ini sepenuhnya dilakukan menggunakan *Virtual Machine*, sehingga tidak ada data sesungguhnya yang diperoleh dari sensor. Maka dari itu, perolehan data akan disiasati dengan melakukan *generate* data oleh Node-RED. Selain itu, Node-RED digunakan untuk melakukan *hashing* dan validasi *hash*.

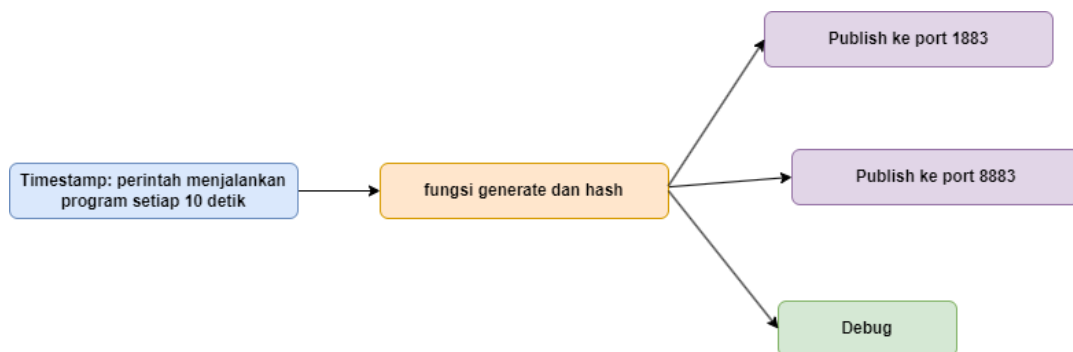
Tabel 3.5 Data-data untuk dilakukan simulasi

Sensor	Parameter Pengukuran	Satuan
Sensor pH	Mengukur jumlah alkalinitas dan keasaman air.	Skala 0 - 14 pH
Sensor Suhu (DHT11)	Mengukur suhu dan kelembapan air.	Skala 0 - 30°C
Sensor Turbiditas	Mengukur kekeruhan air dengan menghitung partikel yang terdapat pada air menggunakan cahaya.	NTU (<i>Nephelometric Turbidity Units</i>), dengan skala angka 0 - 100

Sensor Konduktivitas (TDS)	Mengukur konduktivitas air dengan menghitung <i>Total Dissolved Solids</i> (TDS)	ppm (<i>parts per million</i>), dengan skala 0 - 5000
----------------------------	--	---

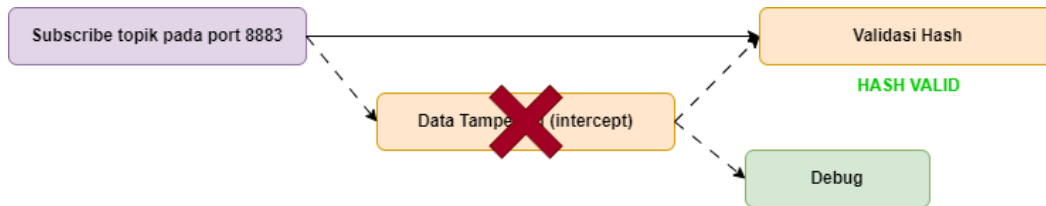
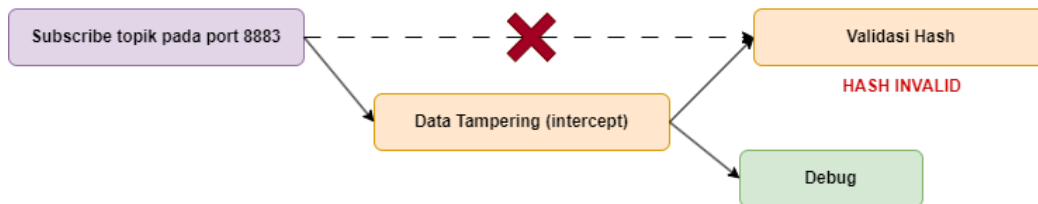
Tabel 3.5 menunjukkan parameter kualitas air yang dihasilkan dari sensor-sensor Bank Air Kami v2. Data-data tersebut akan di-*generate* oleh Node-RED dan melakukan *hashing*. Menciptakan program pada Node-RED adalah dengan cara membuat *flow* yang terdiri dari blok-blok fungsi atau *node*. Pada penelitian ini, dua *flow* akan dibuat dengan rincian sebagai berikut:

- a. *Flow* 1, untuk melakukan *generate* data setiap 10 detik, lalu mengirimkannya atau *publish* ke *broker* Mosquitto pada *port* 1883 dan 8883. Hal ini dilakukan sebagai simulasi perolehan data dari sensor yang diteruskan ke Raspberry Pi melalui MQTT. Sebelum data dikirim ke *broker* Mosquitto dengan topik yang ditentukan, data dilengkapi dengan kode *hash* sebagai upaya untuk meningkatkan kepercayaan data. Rancangan *flow* 1 adalah seperti pada Gambar 3.4.



Gambar 3.4 *Flow* 1 Node-RED

- b. *Flow* 2, untuk melakukan *subscribe* pada topik yang telah dibuat dan melakukan validasi *hash* pada *port* 8883. Selain itu, pengujian aspek integritas dilakukan pada *flow* ini dengan melakukan simulasi *data tampering* sebagai upaya memotong komunikasi dan mengubah data untuk menguji validasi *hash* ketika adanya perubahan data. Ketika terdapat perubahan data, maka *hash* tidak valid sebagai indikator adanya perubahan data. Rancangan *flow* 2 pada skenario normal adalah seperti pada Gambar 3.5, sementara pada skenario terjadi serangan adalah seperti pada Gambar 3.6.

Gambar 3.5 *Flow 2* pada skenario normalGambar 3.6 *Flow 2* pada skenario terjadi serangan

3.4.3 Rencana Implementasi Snort

Snort akan diimplementasi pada Raspberry Pi OS sebagai *Network Intrusion Detection System* (NIDS). Snort bekerja dengan cara mengeksekusi perintah-perintah yang dituliskan atau *Snort rules* untuk mendeteksi serangan-serangan yang dilakukan oleh Kali Linux. Snort akan melakukan deteksi pada seluruh perangkat pada alamat IP 192.168.100.0/24. Berikut beberapa serangan yang akan dideteksi oleh Snort IDS:

- a. *LAND Attack*. Serangan *Distributed Denial of Service* (DDoS) yang bergerak pada lapisan *Transport Control Protocol* (TCP) untuk menyerang lalu lintas jaringan perangkat berskala kecil.
- b. *SYN Flood Attack*: Serangan *Distributed Denial of Service* (DDoS) yang bergerak pada lapisan *Transport Control Protocol* (TCP) untuk menyerang jaringan dan proses kerja perangkat berskala besar.
- c. *Smurf Attack*: Serangan *Distributed Denial of Service* (DDoS) yang bergerak pada lapisan *Internet Control Message Protocol* (ICMP) untuk mengirimkan pesan berskala besar kepada perangkat.
- d. *Port Scanning*: Merupakan upaya penelusuran untuk melihat kelemahan atau celah pada jaringan perangkat, sehingga berpotensi dilakukannya serangan yang lebih besar melalui celah yang terekspos.

3.4.4 Rencana Implementasi UFW Firewall

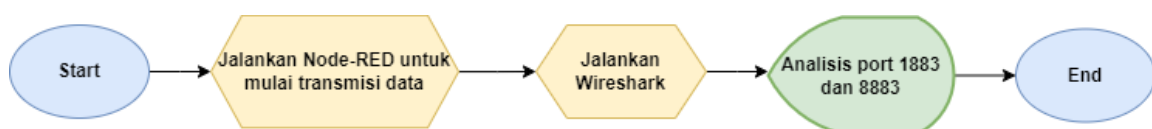
Firewall yang digunakan pada penelitian ini adalah *UFW*. *Firewall* bekerja dengan menuliskan peraturan atau *rules* yang berisi perizinan pada entitas yang akan berkomunikasi dengan Raspberry Pi. Tujuan utama menerapkan *Firewall* pada penelitian ini adalah sebagai pelapis keamanan tambahan bagi perangkat untuk memblokir beberapa jaringan yang tidak diinginkan, berikut rincian *port* jaringan yang dituliskan peraturannya:

- Port* 22. Merupakan *port* untuk melakukan komunikasi *Secure Shell* (SSH) dengan perangkat lain. *Port* 22 akan dilakukan limitasi, sehingga hanya perangkat yang diizinkan dapat berkomunikasi melalui SSH dengan Raspberry Pi.
- Port* 80. Mengizinkan *port* 80 atau HTTP untuk berkomunikasi dengan Raspberry Pi.
- Port* 443. Mengizinkan *port* 443 atau HTTPS untuk berkomunikasi dengan Raspberry Pi.
- Port* 1883. Tidak mengizinkan *port* 1883 atau MQTT untuk dapat diakses oleh perangkat lain.

3.5 Rencana Pengujian pada Desain Keamanan

Pengujian akan dilakukan pada Raspberry Pi OS untuk Aspek Kerahasiaan dan Integritas, sementara Aspek Ketersediaan akan menggunakan Kali Linux untuk meluncurkan serangan ke Raspberry Pi OS. Tujuan melakukan pengujian adalah untuk menguji fungsionalitas desain keamanan yang telah dirancang, lalu diimplementasikan pada Virtual Machine sebagai simulasi alur komunikasi data pada Bank Air Kami v2. Pengujian ini terdiri dari 3 tahapan, yaitu pengujian pada Aspek Kerahasiaan (*Confidentiality*), pengujian pada Aspek Integritas (*Integrity*), dan pengujian pada Aspek Ketersediaan (*Availability*).

3.5.1 Aspek Kerahasiaan (*Confidentiality*)

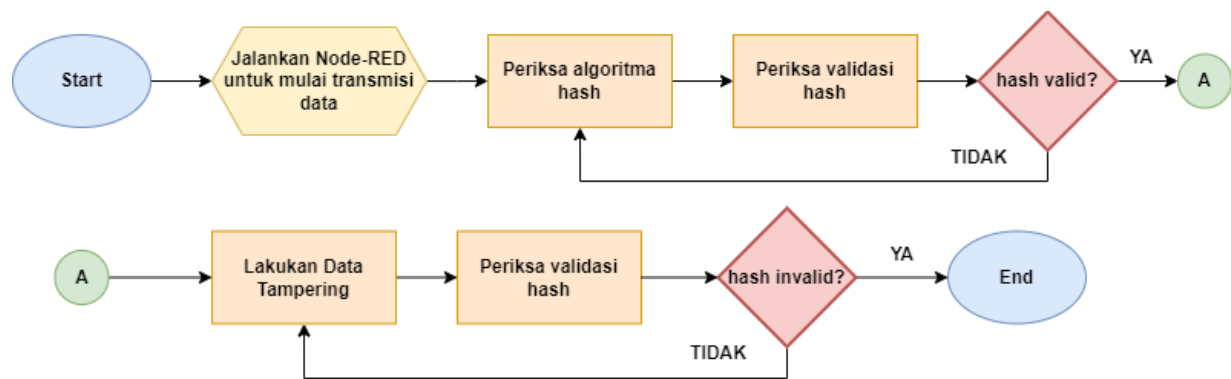


Gambar 3.7 Flowchart pengujian aspek kerahasiaan

Fitur keamanan yang diterapkan pada aspek ini adalah melengkapi *broker* MQTT dengan *Transport Layer Security* (TLS) menggunakan sertifikat *Secure Socket Layer* (SSL). Pengujian

akan dilakukan menggunakan Wireshark untuk menangkap paket dan Node-RED untuk *generate data*. Pengujian dilakukan pada port 1883 dan 8883 untuk melihat komunikasi *broker* MQTT pada kedua *port*. Melakukan analisis paket bertujuan untuk melihat apakah komunikasi terenkripsi atau tidak, *port* yang dilengkapi dengan TLS tidak akan mengekspos paket yang terjadi pada komunikasinya. Sedangkan *port* yang tidak dilengkapi dengan TLS tidak akan terenkripsi. Hal ini dapat diketahui melalui paket yang tertangkap oleh Wireshark, menandakan bahwa komunikasi yang tidak terenkripsi.

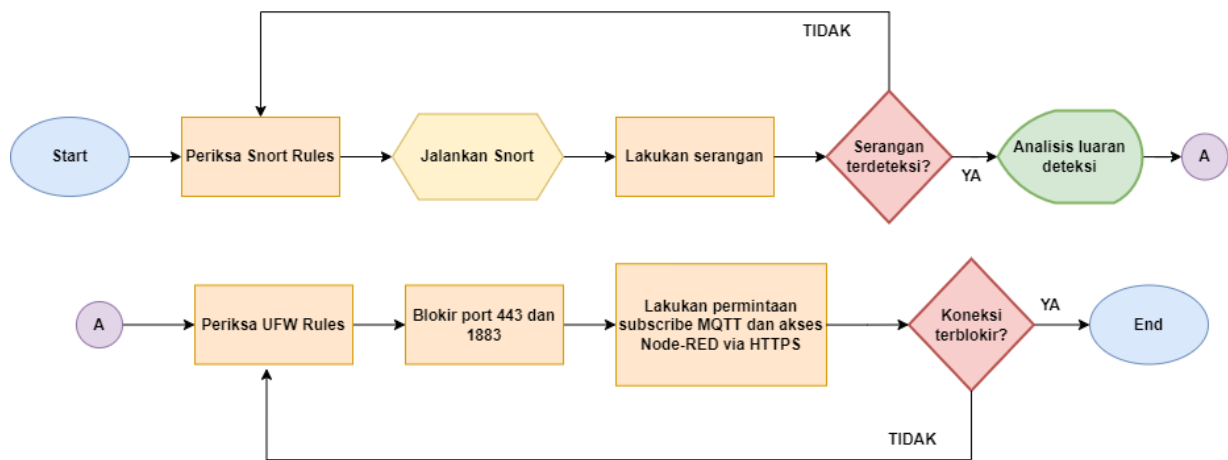
3.5.2 Aspek Integritas (*Integrity*)



Gambar 3.8 Flowchart pengujian aspek integritas

Fitur keamanan yang diterapkan pada aspek ini adalah mengimplementasi *hash* data pada Node-RED menggunakan *library* “CryptoJS” dengan algoritma *hash* “SHA-256” pada setiap pesan untuk memberikan *hash* dan melakukan validasi. Hal ini bertujuan untuk mitigasi adanya modifikasi atau perubahan data yang terjadi ketika transmisi data. *Generate* data dan *hashing* dilakukan pada *flow* 1 dan validasi pada *flow* 2. Pengujian dilakukan dengan melakukan *Data Tampering* pada *flow* 2 dengan menjalankan program untuk menambah nilai dari data, sehingga validasi *hash* akan menunjukkan pesan *hash invalid*.

3.5.3 Aspek Ketersediaan (*Availability*)



Gambar 3.9 *Flowchart* pengujian pada aspek ketersediaan

Fitur keamanan yang diterapkan pada aspek ini adalah mengimplementasi *Network Intrusion Detection System* (NIDS) berbasis tanda menggunakan Snort yang diimplementasi pada Raspberry Pi OS untuk mendeteksi adanya serangan pada jaringan. Mitigasi utama dari aspek ini adalah serangan *Distributed Denial of Service* (DDoS) yang dapat menyebabkan kegagalan pada sistem. Selain itu, Snort akan dilengkapi dengan *rule* yang dapat mendeteksi serangan *Port Scanning*, di mana serangan tersebut dapat mengisiasi serangan berikutnya oleh penyerang. Pengujian aspek ini akan melakukan simulasi DDoS dengan tipe serangan *LAND Attack*, *SYN Flood Attack*, dan *Smurf Attack* menggunakan *tool* Hping3 pada Kali Linux. Setelah menguji DDoS, akan dilanjutkan dengan menguji *Port Scanning* berupa *TCP ACK Scan* menggunakan *tool* nmap.

Selain itu, implementasi keamanan pada jaringan dilakukan dengan cara menerapkan *Firewall* pada jaringan menggunakan *ufw* pada Raspberry Pi OS. Konfigurasi *ufw* dengan cara menuliskan *rules* pada *Terminal* untuk melakukan perintah seperti mengizinkan koneksi terhadap Alamat IP tertentu dan membatasi koneksi pada perangkat atau jaringan tertentu. Pada pengujian ini, Kali Linux akan mengirim permintaan untuk *subscribe* ke *port* 1883 dan melakukan akses Node-RED melalui HTTPS. Kedua permintaan tersebut tidak akan terjadi ketika *Firewall* tidak mengizinkan jaringan untuk berkomunikasi dengan Raspberry Pi OS.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

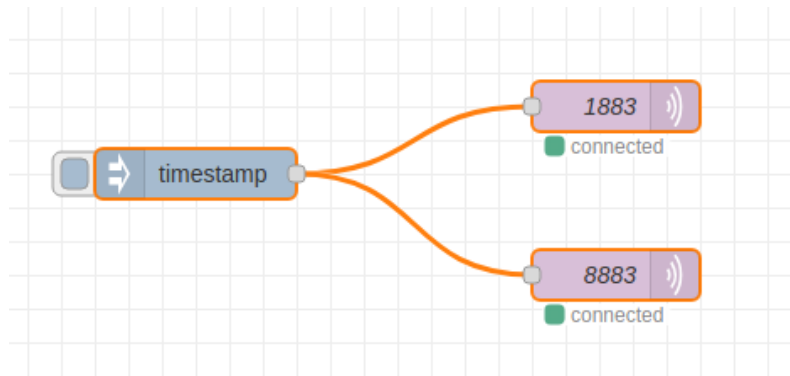
4.1 Implementasi Desain Keamanan

Bagian ini membahas tentang implementasi desain keamanan yang telah dirancang pada tahap sebelumnya mengenai rencana desain keamanan dan rencana pengujian yang bertujuan untuk melakukan simulasi penerapan fitur-fitur pada sistem Bank Air Kami v2, sehingga implementasi secara langsung pada sistem dapat dilakukan dengan efektif. Bagian ini memuat implementasi yang dilakukan melalui pendekatan trias CIA (*Confidentiality, Integrity, Availability*). Pada aspek *Confidentiality*, implementasi MQTT dengan TLS dilakukan, aspek *Integrity* implementasi Node-RED dengan memberikan fungsi hash pada setiap pesan, dan aspek *Availability* implementasi Snort IDS untuk sebagai deteksi serangan yang mungkin terjadi pada Bank Air Kami v2 dan melengkapi jaringan dengan Firewall pada Raspberry Pi untuk mengantisipasi penyerangan jaringan. Setelah implementasi dilakukan dan fitur-fitur keamanan pada desain keamanan telah diterapkan, maka dilakukan pengujian dengan simulasi serangan untuk menguji ketahanan pada desain keamanan dan melihat celah keamanan yang masih rentan terserang oleh penyerang.

4.1.1 Implementasi Aspek Kerahasiaan (*Confidentiality*)

Konfigurasi *Broker Mosquitto*

Untuk mengimplementasi komunikasi terenkripsi pada MQTT menggunakan TLS, maka diharuskan untuk menggunakan *port* 8883 sebagai media komunikasi utama yang dilengkapi autentikasi SSL, sementara *port* 1883 akan digunakan sebagai komparasi pada pengujian menggunakan Wireshark antara *port* terenkripsi dengan port yang tidak terenkripsi. Untuk melihat apakah *broker* telah berjalan dan terkoneksi, peneliti akan menggunakan Node-RED untuk koneksi *port* 1883 dan 8883.



Gambar 4.1 Alur koneksi *broker* MQTT

Gambar 4.1 menunjukkan alur pada ruang kerja Node-RED untuk melihat apakah *broker* telah berjalan dan dapat terkoneksi dengan Node-RED. *Node timestamp* berfungsi untuk melakukan injeksi data pada *node mqtt* yang disambungkan, *node mqtt* dengan nama 1883 menunjukkan *port* 1883 dan 8883 menunjukkan *port* 8883. Konfigurasi pada Node-RED untuk *port* 1883 dengan memasukkan alamat *server* yaitu 192.168.100.7:1883, sedangkan untuk *port* 8883 yaitu 192.168.100.7:8883 dengan melakukan konfigurasi TLS pada menu *security*. Sementara untuk mengadakan atau konfigurasi *port* dilakukan pada *file* lokal dengan direktori `/etc/mosquitto/mosquitto.conf`.

Implementasi TLS pada *broker*

Transport Layer Security (TLS) bekerja dengan cara melakukan pengamanan pada *port*, sehingga *port* hanya dapat diakses oleh entitas yang memiliki kunci dan sertifikat untuk berkomunikasi. Sertifikat dan kunci akan disimpan pada direktori lokal, sehingga ketika entitas ingin mengakses *port*, maka perlu menyertakan *file* kunci dan sertifikat.

Konfigurasi TLS dilakukan dalam beberapa langkah. Di antaranya membuat direktori untuk menyimpan *file* kunci dan sertifikat, melakukan *generate private key* untuk *Certificate Authority* (CA), melakukan *generate* sertifikat untuk CA, melakukan *generate private key* untuk *server*, melakukan *generate Certificate Signing Request* (CSR) menggunakan *server key*, lengkapi sertifikat *server* dengan sertifikat CA, konfigurasi pada *file* `mosquitto.conf`, dan terakhir memberikan izin kepada *file* sertifikat dan kunci untuk dapat diproses oleh *mosquitto*.

Langkah pertama adalah membuat direktori lokal untuk menyimpan *file*, direktori yang digunakan adalah `/etc/mosquitto/certs` menggunakan perintah `cd` pada terminal. Setelah berada pada direktori yang ditentukan, selanjutnya melakukan *generate private key* dan sertifikat untuk CA menggunakan *openssl*.

```
# Melakukan Generate private key
openssl genpkey -algorithm RSA -out ca.key
```

Gambar 4.2 Sintaks perintah melakukan *generate private key*

```
# Melakukan Generate sertifikat untuk CA
openssl req -x509 -new -nodes -key ca.key -sha256 -days 1024 -out
ca.crt
```

Gambar 4.3 Sintaks perintah melakukan *generate* sertifikat untuk CA

Setelah menghasilkan *private key* dan sertifikat untuk CA, maka akan dimintai informasi mengenai negara, nama organisasi, dan lain-lain, untuk disertakan pada sertifikat. Langkah selanjutnya adalah *server key* dan *Certificate Signing Request* (CSR) dengan melakukan perintah pada terminal.

```
# Melakukan Generate server key
openssl genpkey -algorithm RSA -out server.key
```

Gambar 4.4 Sintaks perintah melakukan *generate server key*

```
# Melakukan Generate CSR menggunakan server key
openssl req -new -key server.key -out server.csr
```

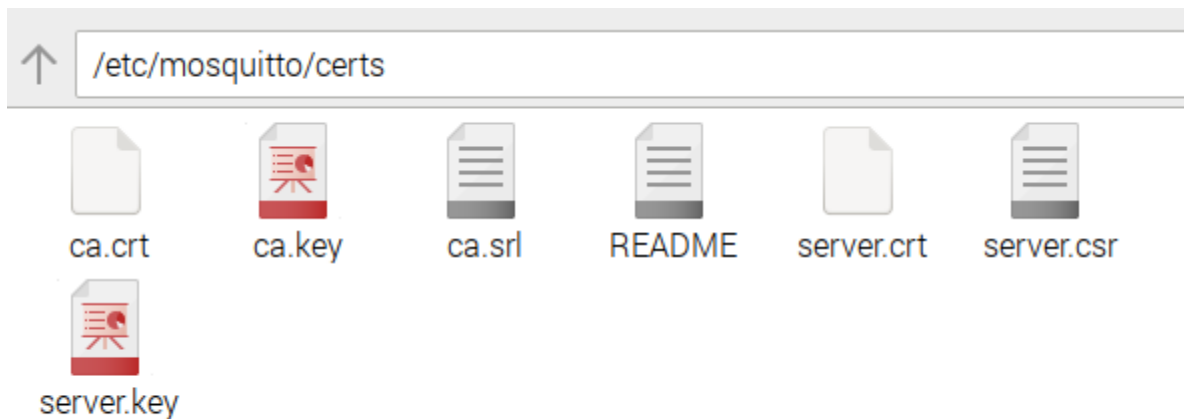
Gambar 4.5 Sintaks perintah melakukan generate CSR

Menjalankan perintah pada Gambar 4.4 dan Gambar 4.5 akan dimintai kembali informasi yang sebelumnya telah diisikan, informasi ini akan disertakan juga dalam CSR. Langkah selanjutnya adalah melengkapi sertifikat *server* dengan sertifikat CA dengan melakukan perintah pada Gambar 4.6.

```
# Melengkapi sertifikat server dengan sertifikat CA
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -
CAcreateserial -out server.crt -days 500 -sha256
```

Gambar 4.6 Sintaks perintah melengkapi sertifikat *server* dengan sertifikat CA

Setelah melakukan langkah-langkah untuk melakukan *generate* pada sertifikat, maka pada direktori yang telah ditentukan akan muncul beberapa *file* seperti yang tertera pada Gambar 4.7.



Gambar 4.7 Direktori *file* sertifikat dan kunci

Terdapat beberapa file yang ditambahkan setelah melakukan generate sebelumnya, berikut rincian setiap file:

- a. `ca.crt`: sertifikat CA
- b. `ca.key`: CA *private key*
- c. `server.crt`: sertifikat *server*
- d. `server.csr`: *server Certificate Signing Request* (CSR)
- e. `server.key`: *server private key*

File `ca.crt`, `server.crt`, dan `server.key` akan digunakan untuk disertakan pada autentikasi ketika *broker* akan dihubungkan kepada Node-RED atau melakukan *publish* dan *subscribe* pada *broker*. Langkah selanjutnya adalah melakukan konfigurasi pada `mosquitto.conf` untuk melengkapi *port* dengan sertifikat dan kunci pada *port* 8883.

```
listener 1883
allow_anonymous true

listener 8883
cafile /etc/mosquitto/certs/ca.crt
certfile /etc/mosquitto/certs/server.crt
keyfile /etc/mosquitto/certs/server.key
```

Gambar 4.8 Konfigurasi TLS pada `mosquitto.conf`

Pada Gambar 4.8 menunjukkan konfigurasi pada `mosquitto.conf`. Untuk menginisiasi *port* menggunakan perintah *listener* yang diikuti dengan nomor *port* secara spesifik. Untuk *port* 1883 yang tidak dilengkapi dengan enkripsi ataupun autentikasi, hanya diberikan perintah untuk mengizinkan siapapun untuk mengakses *port*. Sementara pada *port* 8883 dilengkapi dengan sertifikat CA, sertifikat *server*, dan *server* kunci yang mengarah pada direktori *file* pada Gambar 4.7. Langkah terakhir adalah memastikan Mosquitto dapat memproses sertifikat dan kunci pada direktori dengan memberikan perintah.

```
# mengubah perizinan pada file ca.crt dan server.crt

sudo chmod 644 /etc/mosquitto/certs/ca.crt
/etc/mosquitto/certs/server.crt /etc/mosquitto/certs/server.key
```

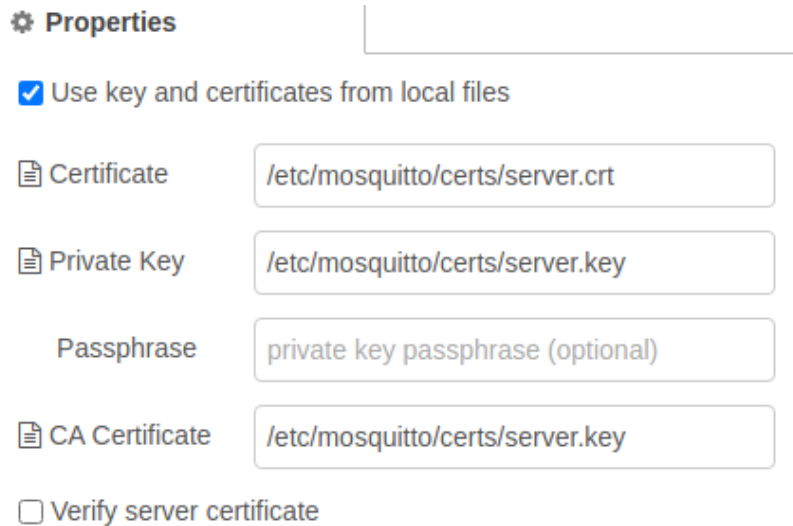
Gambar 4.9 Mengubah perizinan pada *file*

```
# memberikan hak kepemilikan pada mosquitto

sudo chown mosquitto:mosquitto /etc/mosquitto/certs/ca.crt
/etc/mosquitto/certs/server.crt /etc/mosquitto/certs/server.key
```

Gambar 4.10 Memberikan hak kepemilikan pada Mosquitto

Pada Gambar 4.9 merupakan sintaks perintah untuk mengubah perizinan pada *file* `ca.crt` dan `server.crt` menggunakan perintah `chmod`. Sementara pada Gambar 4.10 merupakan sintaks perintah untuk memberikan hak kepemilikan *file* sertifikat dan kunci pada Mosquitto untuk dapat diproses. Untuk menghubungkan *broker* pada Node-RED, sertifikat dan kunci disertakan pada properti *node mqtt* dengan inisiasi konfigurasi TLS.



The image shows the 'Properties' panel in Node-RED for a TLS broker. It includes a checked checkbox 'Use key and certificates from local files'. Below this are four input fields: 'Certificate' with the path '/etc/mosquitto/certs/server.crt', 'Private Key' with the path '/etc/mosquitto/certs/server.key', 'Passphrase' with the text 'private key passphrase (optional)', and 'CA Certificate' with the path '/etc/mosquitto/certs/server.key'. At the bottom is an unchecked checkbox 'Verify server certificate'.

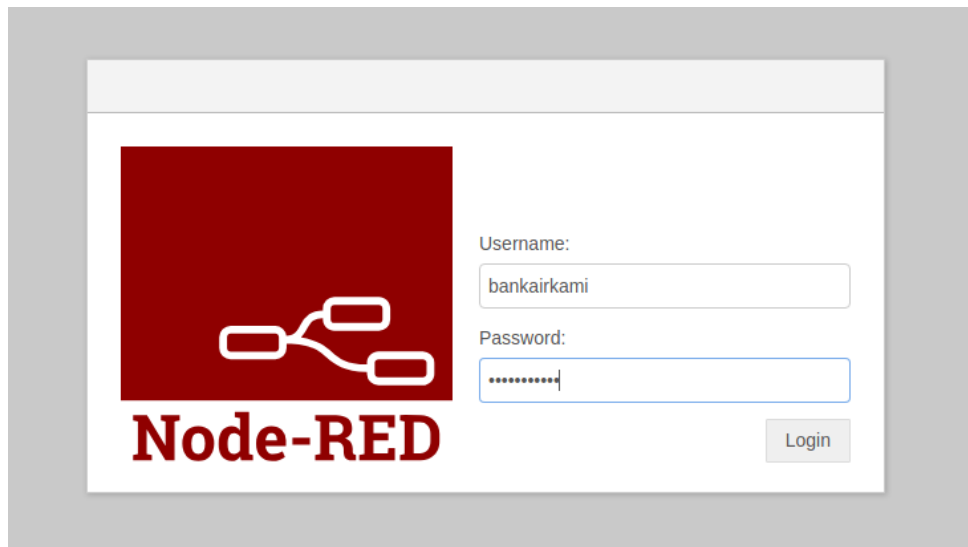
Gambar 4.11 Konfigurasi TLS *broker* pada Node-RED

Gambar 4.1 menunjukkan menu konfigurasi TLS pada Node-RED. *Broker* yang dilengkapi dengan TLS perlu dikonfigurasi melalui Node-RED dengan menyertakan *file* sertifikat dan kunci berdasarkan direktori lokal. Setelah terkoneksi, maka implementasi aspek integritas dapat dilakukan.

4.1.2 Implementasi Aspek Integritas (*Integrity*)

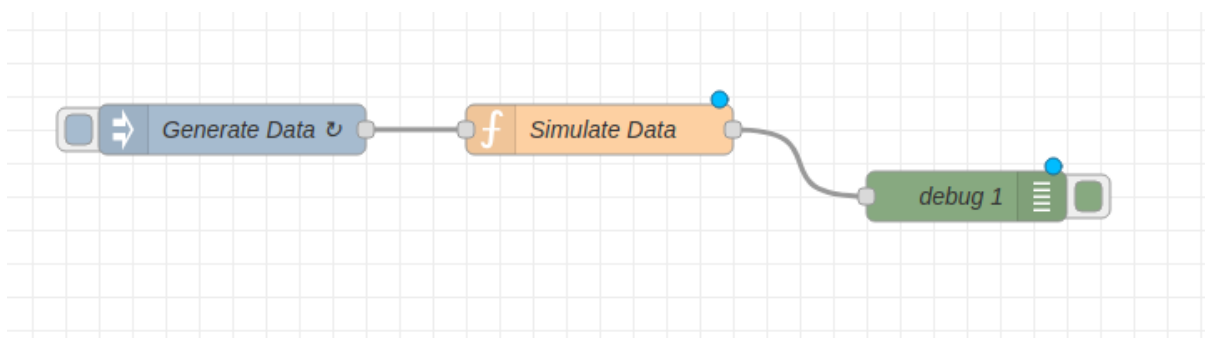
Mengatur Alur Transmisi data pada Node-RED

Simulasi transmisi data Bank Air Kami v2 dapat dilakukan pada Node-RED dengan melakukan *generate data* menggunakan fungsi yang dilengkapi dengan kode *Javascript*. Data yang di-*generate* adalah data kualitas air berupa suhu, pH, turbiditas, dan konduktivitas. Sebelum melakukan konfigurasi pada Node-RED, pengguna diharuskan untuk mengisi autentikasi untuk dapat mengakses *flow* pada Node-RED, hal ini merupakan upaya untuk mencegah pengguna tidak berotoritas masuk dan mengubah *flow*.



Gambar 4.12 Autentikasi pengguna pada Node-RED

Untuk mengakses Node-RED, dilakukan perintah `node-red-start` pada terminal dan mengakses “127.0.0.1:1880” pada browser. *Username* yang digunakan adalah “bankairkami” dan *password* “bankairkami” untuk melakukan konfigurasi pada *flow*. Untuk konfigurasi *flow* yang dapat melakukan *generate* data, menggunakan *node inject* untuk menginisiasi *generate* dengan interval 10 detik, function untuk menyimpan kode *generate* data, dan *debug* untuk melihat apakah data berhasil di-*generate*.



Gambar 4.13 Flow untuk melakukan *generate* data

Dengan menghubungkan ketiga *node* untuk memulai *generate* data. Maka data akan di-*generate* yang bisa dipantau melalui menu *debug* pada Node-RED. Kode *Javascript* yang digunakan untuk melakukan *generate* pada *node function* adalah seperti yang tertera pada Gambar 4.14.

```

var sensorData = {
  temperature: (Math.random() * 30).toFixed(2),
  pH: (Math.random() * 14).toFixed(2),
  turbidity: (Math.random() * 100).toFixed(2),
  conductivity: (Math.random() * 5000).toFixed(2)
};
msg.payload = sensorData;
return msg;

```

Gambar 4.14 kode *Javascript* untuk melakukan *generate* data

Untuk data suhu dilakukan *generate* angka desimal antara 0.00 sampai 30.00 celsius dan menggunakan fungsi `toFixed` untuk memastikan nilai memiliki hanya 2 angka desimal. Untuk data pH dilakukan *generate* angka 0.00 sampai 14.00, data turbiditas dengan *generate* angka 0.00 sampai 100.00 satuan *Nephelometric Turbidity Units* (NTU), dan data konduktivitas dengan *generate* angka 0.00 sampai 5000.00 satuan $\mu\text{S}/\text{cm}$ (*microsiemens per centimeter*). Variabel `sensorData` akan dimasukkan pada pesan objek `msg.payload` dan `return msg;` untuk mengembalikan pesan yang telah dimodifikasi dengan *generate* data. Maka akan muncul pada menu *debug* seperti yang tertera pada Gambar 4.15.



```

5/22/2024, 1:44:42 PM node: debug 1
msg.payload : Object
  { temperature: "3.38", pH: "3.25", turbidity: "8.69", conductivity: "1639.00" }

5/22/2024, 1:44:52 PM node: debug 1
msg.payload : Object
  { temperature: "4.04", pH: "12.84", turbidity: "5.09", conductivity: "435.15" }

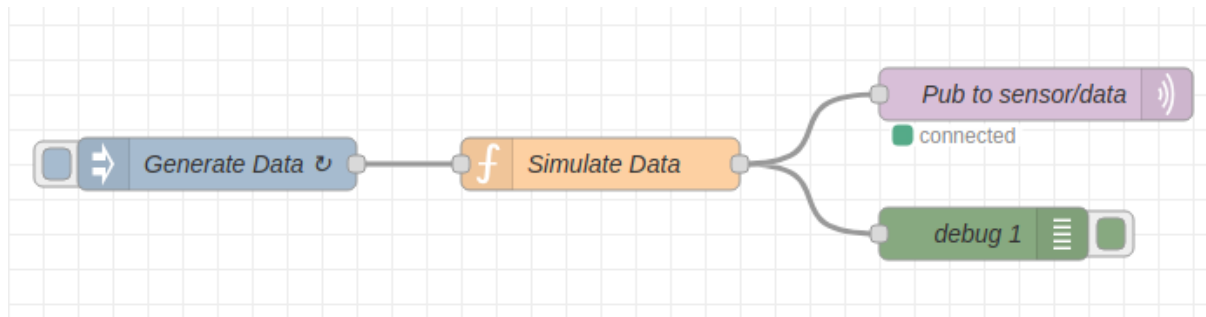
5/22/2024, 1:45:02 PM node: debug 1
msg.payload : Object
  { temperature: "10.24", pH: "9.48", turbidity: "13.22", conductivity: "1996.81" }

```

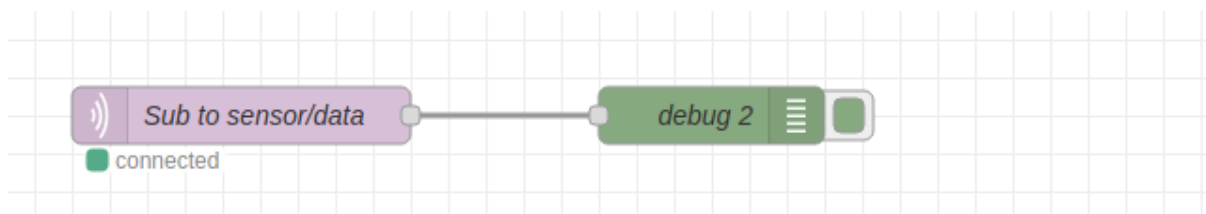
Gambar 4.15 *Debug* pada *flow* *generate* data

Gambar 4.15 menunjukkan 3 pesan terbaru yang berhasil di-*generate* dengan keterangan tanggal dan waktu. *Generate* data dilakukan setiap 10 detik seperti yang terlihat pada keterangan waktu, pesan yang terkirim memuat keempat data kondisi air berdasarkan hasil

generate. Setelah *flow* berhasil melakukan *generate*, maka *flow* akan melakukan *publish* ke *broker* MQTT *port* 8883 menggunakan *node mqtt* dengan topik *sensor/data* untuk mengirim pesan berupa objek, lalu menyiapkan *flow* yang berbeda untuk melakukan *subscribe* pada topik yang sama.

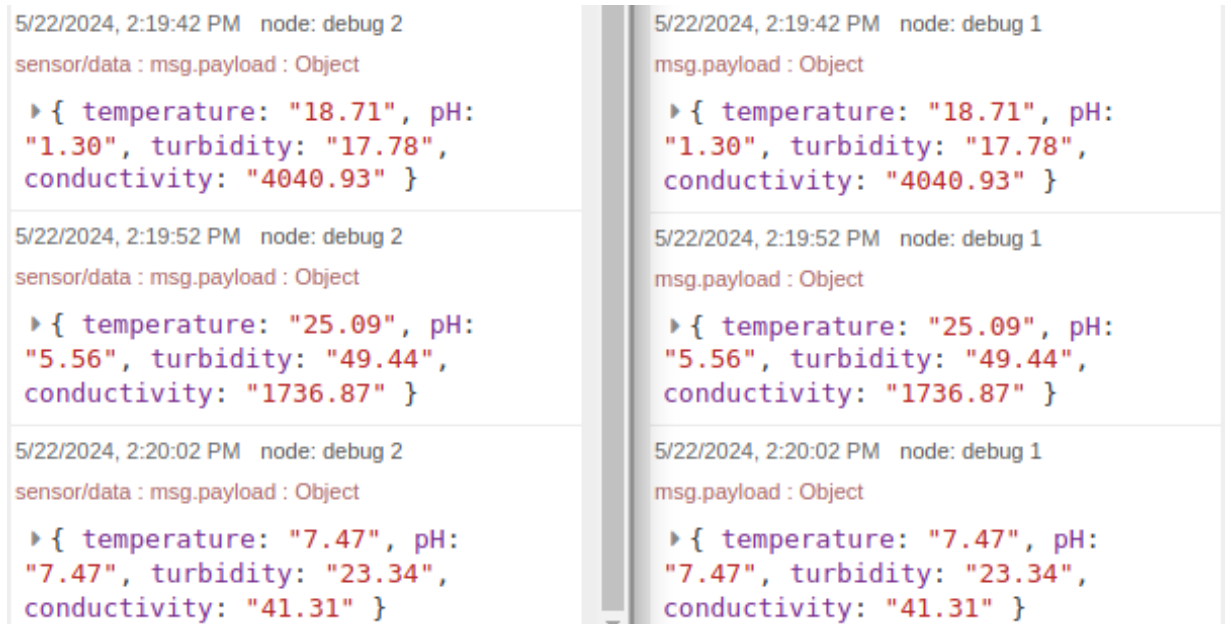


Gambar 4.16 *Flow* melakukan *publish* ke topik *sensor/data*



Gambar 4.17 *Flow* melakukan *subscribe* topik *sensor/data*

Gambar 4.16 menunjukkan *flow* untuk melakukan *publish* kepada topik *sensor/data*, sementara Gambar 4.17 menunjukkan *flow* untuk melakukan *subscribe* pada topik *sensor/data* yang dilengkapi dengan *node debug* untuk melihat apakah pesan diterima.



Gambar 4.18 Debug pada *flow publish* dan *subscribe*

Gambar 4.18 menunjukkan *debug* pada *flow publish* dan *subscribe*, *debug 1* merepresentasikan *flow publish* dan *debug 2* merepresentasikan *flow subscribe*. Pesan yang dikirim dan diterima sama persis pada *flow publish* dan *flow subscribe* pada waktu yang sama, menandakan bahwa komunikasi pada *broker* berhasil dan pesan yang terkirim dan diterima valid.

Implementasi Algoritma Hash

Implementasi algoritma *hash* dilakukan pada pesan yang didapat dari topik *sensor/data*, sehingga fungsi *hash* akan diimplementasi pada *flow 1*. Algoritma *hash* menggunakan *library* “CryptoJS” yang diinstal melalui fungsi *npm* dan disimpan pada direktori Node-RED. Instalasi “CryptoJS” pada direktori lokal Node-RED dengan perintah `cd` yang mengarah pada direktori `/node-red`, lalu menginstal dengan perintah `npm install crypto-js`. Setelah melakukan instalasi, perlu dilakukan konfigurasi pada *file settings.js* untuk menambahkan “CryptoJS” pada “*functionGlobalContext*”, tetapi sebelum melakukan konfigurasi perlu adanya perubahan izin dan kepemilikan pada *file settings.js* dengan menggunakan perintah yang terlihat pada Gambar 4.19.

```
# merubah kepemilikan file kepada user
sudo chown nindyarayhan:nindyarayhan ~/.node-red/settings.js
```

```
# merubah perizinan pada file settings.js
sudo chmod 666 ~/.node-red/settings.js
```

Gambar 4.19 Merubah kepemilikan dan perizinan pada file `settings.js`

Setelah merubah kepemilikan dan perizinan pada *file* `settings.js`, lalu melakukan konfigurasi pada *file* menggunakan perintah `sudo nano ~/.node-red/settings.js` dengan menuliskan kode seperti yang tertera pada Gambar 4.20.

```
# menambahkan kode pada settings.js

functionGlobalContext: {
  CryptoJS: require('crypto-js')}

```

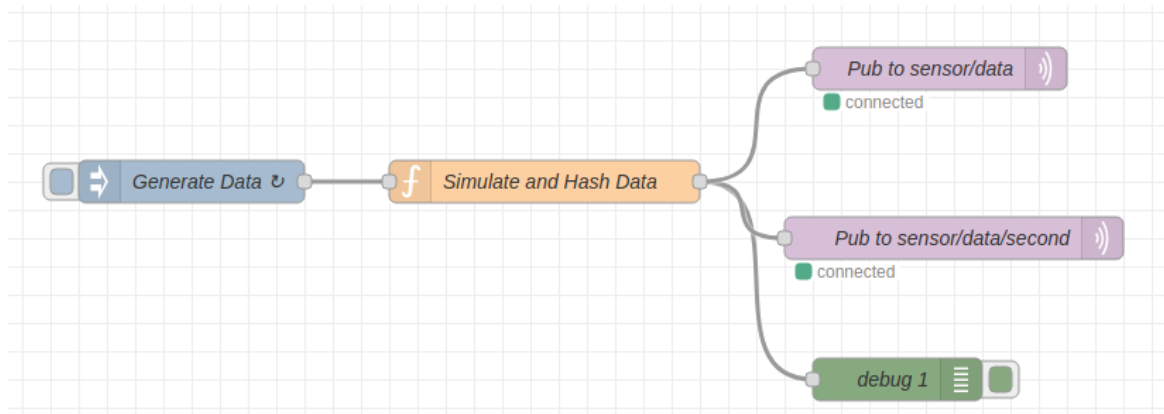
Gambar 4.20 Kode yang ditambahkan pada `settings.js`

Setelah menambahkan kode pada `settings.js`, perizinan file kembali diubah dengan perintah `sudo chmod 644 ~/.node-red/settings.js` untuk memastikan *file* lebih aman. Langkah terakhir adalah mengimplementasi *node function* pada *flow subscribe* yang dilengkapi dengan kode *Javascript* untuk melakukan *hashing*. Fungsi *hash* dilakukan dengan kode *Javascript* seperti yang tertera pada Gambar 4.21.

```
var CryptoJS = global.get('CryptoJS');
var hash =
CryptoJS.SHA256(JSON.stringify(msg.payload)).toString(CryptoJS.enc.Hex);
msg.hash = hash;
return msg;
```

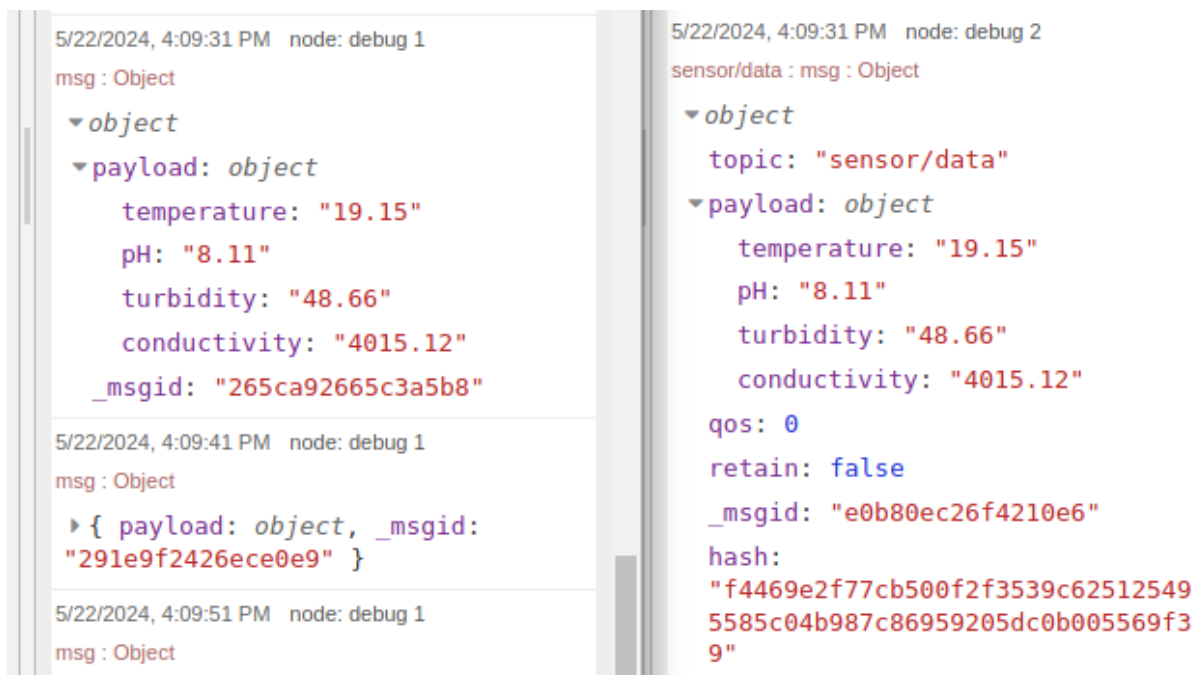
Gambar 4.21 Kode *Javascript* untuk melakukan *hashing*

Algoritma *hash* yang digunakan adalah algoritma “SHA-256” yang berasal dari *library* “CryptoJS”. Pesan yang diterima adalah berupa objek, maka dilakukan konversi kepada pesan menjadi format *JSON String* untuk dilakukan *hash*. Hasil dari *hashing* akan dikonversi menjadi *hexadecimal string*. Proses *hash* disimpan pada variabel *hash* yang akan disertakan pada pesan. Saat ini, setiap memiliki kode *hash* yang dapat dilihat melalui *node debug*.



Gambar 4.22 Flow subscribe yang ditambahkan fungsi hash

Gambar 4.22 menunjukkan *flow 1* yang telah dilengkapi dengan fungsi *hashing* yang menyatu dengan fungsi *generate*. Setiap data yang di-*generate* akan dilengkapi dengan kode *hash* sebagai elemen integritas data, hal ini dinamakan sebagai *Digital Signature*, bertujuan untuk memastikan kepercayaan atas data.



Gambar 4.23 Perbandingan data sebelum dan sesudah *hashing*

Gambar 4.23 menunjukkan hasil perbandingan dari data ketika sebelum dan sesudah *hashing*. Data yang telah dilakukan *hashing* akan disertakan dengan kode *hash* sebagai identitas keaslian data. Ketika data terjadi perubahan, kode *hash* tidak akan berubah.

4.1.3 Implementasi Aspek Ketersediaan (*Availability*)

Instalasi dan Konfigurasi Snort

Megimplementasi Snort sebagai IDS diperlukan konfigurasi seperti penulisan aturan-aturan atau *rules* untuk melakukan deteksi. Untuk mendeteksi jenis serangan secara spesifik, penulisan *rules* ditentukan berdasarkan parameter jenis serangan dan jenis pesan yang akan diperingatkan pada pengguna. Implementasi Snort pada Raspberry Pi dilakukan dalam beberapa langkah, di antaranya instalasi Snort pada terminal, dan menuliskan *rules* untuk mendeteksi jenis serangan. Instalasi Snort pada terminal hanya memasukkan perintah `sudo apt-get install snort`, lalu sistem akan meminta alamat IP dan *interface* jaringan yang digunakan untuk melakukan pemantauan. Pada penelitian ini, alamat IP yang digunakan adalah 192.168.100.0/24 dan *interface* `eth0`. Versi Snort yang digunakan adalah versi 2.9.20. Selanjutnya, dilakukan validasi bahwa Snort telah berjalan dengan cara menambahkan *rule* untuk mendeteksi *Internet Control Message Protocol (ICMP) Ping* pada Raspberry Pi, sehingga *Ping Attack* yang dilakukan oleh Kali Linux dapat terdeteksi oleh Snort. *Rule* yang dituliskan pada direktori `/etc/snort/rules/local.rules` adalah yang tertera pada Gambar 4.24.

```
#rule untuk mendeteksi ICMP Ping
alert icmp any any -> $HOME_NET any (msg:"ICMP Ping Detected";
sid:100001; rev:1;)
```

Gambar 4.24 *Rule* mendeteksi ICMP Ping

Variabel `$HOME_NET` berisi alamat IP dari jaringan yang digunakan. Snort yang dilengkapi dengan *rule* untuk mendeteksi ICMP Ping ketika dijalankan oleh perintah `sudo snort -q -l /var/log/snort -i eth0 -A console -c /etc/snort/snort.conf` dan melakukan *Ping Attack* oleh Kali Linux pada alamat IP target, maka akan muncul pesan pada terminal seperti pada Gambar 4.26.

```

nindyarayhan@nindyarayhan: ~
File Actions Edit View Help

(nindyarayhan@nindyarayhan)-[~]
$ ping 192.168.100.7
PING 192.168.100.7 (192.168.100.7) 56(84) bytes of data.
64 bytes from 192.168.100.7: icmp_seq=1 ttl=64 time=0.259 ms
64 bytes from 192.168.100.7: icmp_seq=2 ttl=64 time=0.326 ms
64 bytes from 192.168.100.7: icmp_seq=3 ttl=64 time=0.288 ms
64 bytes from 192.168.100.7: icmp_seq=4 ttl=64 time=0.267 ms
64 bytes from 192.168.100.7: icmp_seq=5 ttl=64 time=0.432 ms
64 bytes from 192.168.100.7: icmp_seq=6 ttl=64 time=0.987 ms
64 bytes from 192.168.100.7: icmp_seq=7 ttl=64 time=0.398 ms
64 bytes from 192.168.100.7: icmp_seq=8 ttl=64 time=0.291 ms
64 bytes from 192.168.100.7: icmp_seq=9 ttl=64 time=0.340 ms
64 bytes from 192.168.100.7: icmp_seq=10 ttl=64 time=0.342 ms
64 bytes from 192.168.100.7: icmp_seq=11 ttl=64 time=0.314 ms
64 bytes from 192.168.100.7: icmp_seq=12 ttl=64 time=0.245 ms
64 bytes from 192.168.100.7: icmp_seq=13 ttl=64 time=0.261 ms
64 bytes from 192.168.100.7: icmp_seq=14 ttl=64 time=0.260 ms
64 bytes from 192.168.100.7: icmp_seq=15 ttl=64 time=0.296 ms
64 bytes from 192.168.100.7: icmp_seq=16 ttl=64 time=0.564 ms
64 bytes from 192.168.100.7: icmp_seq=17 ttl=64 time=0.417 ms
64 bytes from 192.168.100.7: icmp_seq=18 ttl=64 time=0.405 ms
64 bytes from 192.168.100.7: icmp_seq=19 ttl=64 time=0.432 ms
64 bytes from 192.168.100.7: icmp_seq=20 ttl=64 time=1.54 ms
64 bytes from 192.168.100.7: icmp_seq=21 ttl=64 time=0.304 ms
64 bytes from 192.168.100.7: icmp_seq=22 ttl=64 time=0.367 ms

```

Gambar 4.25 Kali Linux melakukan *Ping*

```

05/25-00:41:32.768583 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.4 -> 192.168.100.7
05/25-00:41:33.768588 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.4
05/25-00:41:34.844601 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.4 -> 192.168.100.7
05/25-00:41:34.844630 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.4
05/25-00:41:36.122122 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.4 -> 192.168.100.7
05/25-00:41:36.122148 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.4
05/25-00:41:37.177710 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.4 -> 192.168.100.7
05/25-00:41:37.177741 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.4
05/25-00:41:38.210859 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.4 -> 192.168.100.7
05/25-00:41:38.210886 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.4
05/25-00:41:39.233083 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.4 -> 192.168.100.7
05/25-00:41:39.233109 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.4
05/25-00:41:40.253103 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.4 -> 192.168.100.7
05/25-00:41:40.253131 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.4
05/25-00:41:41.279624 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.4 -> 192.168.100.7
05/25-00:41:41.279645 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.4
05/25-00:41:42.303152 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.4 -> 192.168.100.7
05/25-00:41:42.303175 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.4
05/25-00:41:43.322568 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.4 -> 192.168.100.7
05/25-00:41:43.322607 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.4
05/25-00:41:44.346062 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.4 -> 192.168.100.7
05/25-00:41:44.346087 [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.4

```

Gambar 4.26 Hasil deteksi Snort pada ICMP *Ping*

Gambar 4.25 menunjukkan perintah untuk melakukan *Ping* kepada alamat IP 192.168.100.7, maka Snort mendeteksi Ping tersebut dengan pesan yang telah ditentukan pada *rule* seperti pada Gambar 4.26. Maka dari itu, Snort IDS berhasil dijalankan.

Selanjutnya, Snort dilengkapi dengan *rules* untuk mendeteksi serangan *LAND Attack*, *SYN Flood Attack*, *Smurf Attack*, dan *TCP ACK Scan*. Semua *rule* dituliskan pada file `local.rules` pada direktori `etc/snort/rules` yang dijalankan ketika pengujian. *Rules* yang dituliskan adalah seperti tertera pada Gambar 4.27.

```
# LAND Attack

alert tcp any any -> any any (msg:"LAND Attack Detected"; flags:S;
sameip; sid:1000003; rev:1;)

# SYN Flood Attack

alert tcp any any -> any any (msg:"SYN Flood Attack Detected";
flags:S; detection_filter:track by_dst, count 100, seconds 1;
sid:1000004; rev:1;)

# Smurf Attack

alert icmp any any -> any any (msg:"Smurf Flooding Attack Detected";
threshold: type threshold, track by_src, count 20, seconds 60;
sid:100000023; rev:1;)

# TCP ACK Scan

alert tcp any any -> any any (msg:"TCP ACK Scan Detected"; flags:A;
detection_filter:track by_dst, count 100, seconds 1; sid:1000005;
rev:1;)
```

Gambar 4.27 *Rules* pada Snort untuk deteksi serangan

Simulasi serangan dilakukan menggunakan Kali Linux untuk menyerang alamat IP Raspberry Pi yang dilengkapi dengan IDS. Dengan menuliskan *rules* untuk mendeteksi jenis serangan secara spesifik, maka pengujian dapat dilakukan dan Snort dapat memberikan peringatan kepada pengguna.

Implementasi Firewall

Implementasi *Firewall* pada Raspberry Pi bertujuan untuk memastikan hanya perangkat atau *port* yang diizinkan, hal ini mengurangi kemungkinan penyerang untuk melakukan serangan pada jaringan ataupun sistem. Jenis *firewall* yang digunakan adalah *ufw*. Setelah melakukan instalasi, *ufw* perlu diberikan kebijakan dalam menjalankan *firewall*, memberikan kebijakan untuk menolak semua hal yang masuk menggunakan perintah `sudo ufw default deny incoming` dan mengizinkan segala hal untuk keluar menggunakan

perintah `sudo ufw default allow outgoing`. Konfigurasi yang digunakan adalah melakukan limitasi pada *port* 22 yaitu *Secure Powershell* (SSH), tidak mengizinkan *port* 1883 dan 8883 untuk dapat berkomunikasi dengan perangkat lain, dan memblokir alamat IP tertentu.

```
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To Action From
--
22/tcp ALLOW IN Anywhere
80/tcp ALLOW IN Anywhere
443/tcp ALLOW IN Anywhere
1883/tcp DENY IN Anywhere
8883/tcp DENY IN Anywhere
22/tcp (v6) ALLOW IN Anywhere (v6)
80/tcp (v6) ALLOW IN Anywhere (v6)
443/tcp (v6) ALLOW IN Anywhere (v6)
1883/tcp (v6) DENY IN Anywhere (v6)
8883/tcp (v6) DENY IN Anywhere (v6)
```

Gambar 4.28 Konfigurasi *Firewall ufw* pada Raspberry Pi

Gambar 4.28 menunjukkan konfigurasi *ufw* dalam daftar *port* yang diizinkan untuk menerima segala hal dari luar. Hal yang paling utama untuk diterapkan konfigurasi *firewall* adalah komunikasi pada *broker* MQTT. Selain untuk mencegah penyerang melakukan modifikasi pada data yang dapat menyerang aspek integritas, *firewall* juga dapat mencegah serangan seperti serangan *Distributed Denial of Service* (DDoS) dan *Man-in-the-Middle* (MiTM) dalam skala tertentu.

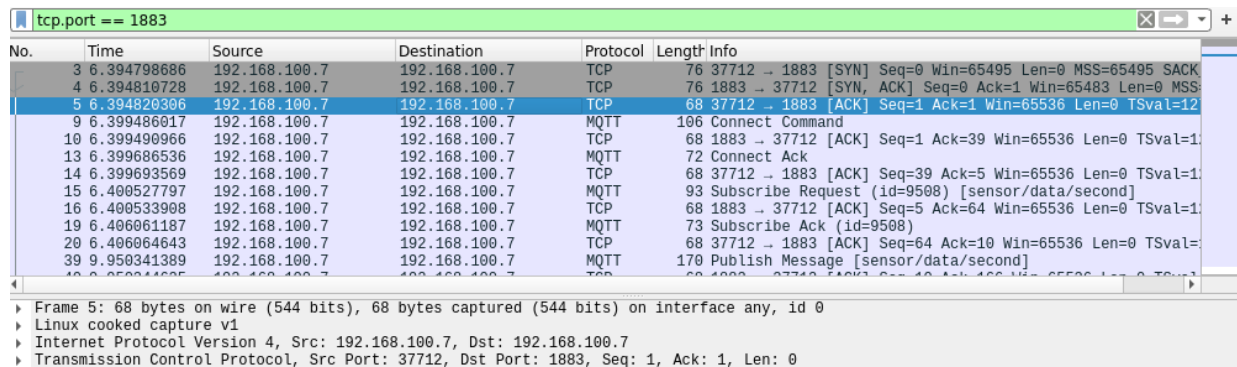
4.2 Pengujian Desain Keamanan

Pengujian dilakukan untuk melihat apakah desain telah terimplementasi dengan baik. Pengujian dilakukan berdasarkan aspek trias CIA yang telah ditentukan sebelumnya, seperti analisis lalu lintas jaringan pada *broker* MQTT, melakukan simulasi serangan menggunakan Kali Linux, dan melakukan deteksi menggunakan Snort IDS dengan *rule* yang telah dituliskan.

4.2.1 Pengujian pada Implementasi Aspek Kerahasiaan (*Confidentiality*)

Untuk menguji implementasi TLS yang telah dilakukan, maka dilakukan analisis TLS melalui analisis komunikasi MQTT pada simulasi Bank Air Kami v2. Analisis melibatkan

broker port 1883 tanpa enkripsi dan *port* 8883 dengan enkripsi TLS. Analisis dilakukan menggunakan Wireshark pada Raspberry Pi dengan menangkap paket dari *interface* eth0. Sebelum dilakukan analisis, *generate data* akan di-publish dengan topik *sensor/data/second* pada *port* 1883 untuk melihat mekanisme komunikasi paket yang dilakukan.

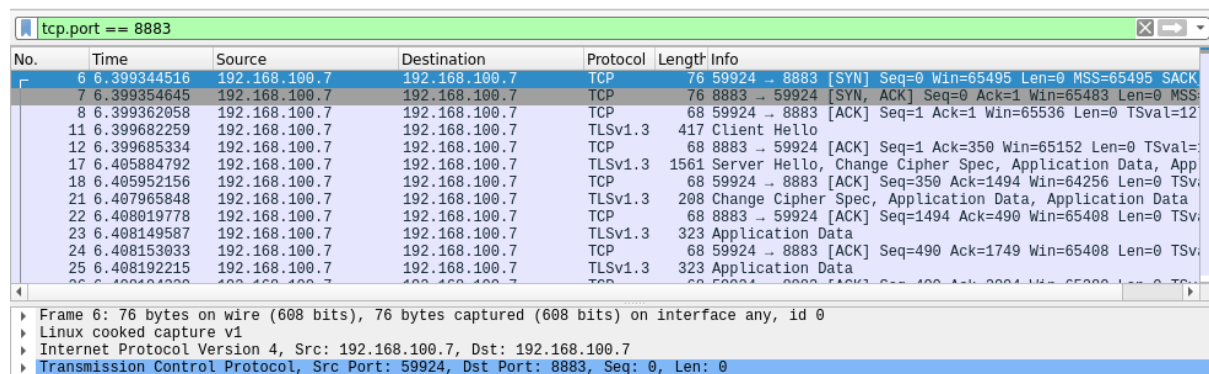


tcp.port == 1883

No.	Time	Source	Destination	Protocol	Length	Info
3	6.394798686	192.168.100.7	192.168.100.7	TCP	76	37712 → 1883 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK
4	6.394810728	192.168.100.7	192.168.100.7	TCP	76	1883 → 37712 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS
5	6.394820306	192.168.100.7	192.168.100.7	TCP	68	37712 → 1883 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=12
9	6.399486017	192.168.100.7	192.168.100.7	MQTT	106	Connect Command
10	6.399490966	192.168.100.7	192.168.100.7	TCP	68	1883 → 37712 [ACK] Seq=1 Ack=39 Win=65536 Len=0 TSval=1
13	6.399686536	192.168.100.7	192.168.100.7	MQTT	72	Connect Ack
14	6.399693569	192.168.100.7	192.168.100.7	TCP	68	37712 → 1883 [ACK] Seq=39 Ack=5 Win=65536 Len=0 TSval=1
15	6.400527797	192.168.100.7	192.168.100.7	MQTT	93	Subscribe Request (id=9508) [sensor/data/second]
16	6.400533908	192.168.100.7	192.168.100.7	TCP	68	1883 → 37712 [ACK] Seq=5 Ack=64 Win=65536 Len=0 TSval=1
19	6.400601187	192.168.100.7	192.168.100.7	MQTT	73	Subscribe Ack (id=9508)
20	6.400606463	192.168.100.7	192.168.100.7	TCP	68	37712 → 1883 [ACK] Seq=64 Ack=10 Win=65536 Len=0 TSval=
39	9.950341389	192.168.100.7	192.168.100.7	MQTT	170	Publish Message [sensor/data/second]

Frame 5: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0
 Linux cooked capture v1
 Internet Protocol Version 4, Src: 192.168.100.7, Dst: 192.168.100.7
 Transmission Control Protocol, Src Port: 37712, Dst Port: 1883, Seq: 1, Ack: 1, Len: 0

Gambar 4.29 Tangkapan paket pada *port* 1883



tcp.port == 8883

No.	Time	Source	Destination	Protocol	Length	Info
6	6.399344516	192.168.100.7	192.168.100.7	TCP	76	59924 → 8883 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK
7	6.399354645	192.168.100.7	192.168.100.7	TCP	76	8883 → 59924 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS
8	6.399362058	192.168.100.7	192.168.100.7	TCP	68	59924 → 8883 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=12
11	6.399682259	192.168.100.7	192.168.100.7	TLSv1.3	417	Client Hello
12	6.399685334	192.168.100.7	192.168.100.7	TCP	68	8883 → 59924 [ACK] Seq=1 Ack=350 Win=65152 Len=0 TSval=
17	6.405884792	192.168.100.7	192.168.100.7	TLSv1.3	1561	Server Hello, Change Cipher Spec, Application Data, App
18	6.405952156	192.168.100.7	192.168.100.7	TCP	68	59924 → 8883 [ACK] Seq=350 Ack=1494 Win=64256 Len=0 TSv
21	6.407965848	192.168.100.7	192.168.100.7	TLSv1.3	208	Change Cipher Spec, Application Data, Application Data
22	6.408019778	192.168.100.7	192.168.100.7	TCP	68	8883 → 59924 [ACK] Seq=1494 Ack=490 Win=65408 Len=0 TSv
23	6.408149587	192.168.100.7	192.168.100.7	TLSv1.3	323	Application Data
24	6.408153033	192.168.100.7	192.168.100.7	TCP	68	59924 → 8883 [ACK] Seq=490 Ack=1749 Win=65408 Len=0 TSv
25	6.408192215	192.168.100.7	192.168.100.7	TLSv1.3	323	Application Data

Frame 6: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface any, id 0
 Linux cooked capture v1
 Internet Protocol Version 4, Src: 192.168.100.7, Dst: 192.168.100.7
 Transmission Control Protocol, Src Port: 59924, Dst Port: 8883, Seq: 0, Len: 0

Gambar 4.30 Tangkapan paket pada *port* 8883

Gambar 4.29 menunjukkan hasil tangkapan paket dari port 1883 dan Gambar 4.30 menunjukkan hasil tangkapan paket dari port 8883. Terlihat bahwa port 1883 menggunakan protokol MQTT pada komunikasinya, sementara port 8883 menggunakan protokol TLS dengan versi 1.3. Paket ditangkap ketika *broker* Mosquitto dijalankan, sehingga terlihat permintaan *Connect* kepada *broker* pada port 1883. Terdapat perbedaan antara *broker* yang menggunakan protokol TLS dengan yang hanya menggunakan MQTT. Pada TLS, komunikasi terenkripsi, sehingga Wireshark tidak mampu memperlihatkan muatan pesan karena tidak

mampu melakukan dekripsi pada muatan pesan, sementara pada *port* 1883 terlihat kontrol paket seperti *Subscribe*, *Publish*, *Connect*, dan lain-lain.

Protocol	Length	Info
TCP	76	37712 → 1883 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK
TCP	76	1883 → 37712 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=
TCP	68	37712 → 1883 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=12
MQTT	106	Connect Command
TCP	68	1883 → 37712 [ACK] Seq=1 Ack=39 Win=65536 Len=0 TSval=1
MQTT	72	Connect Ack
TCP	68	37712 → 1883 [ACK] Seq=39 Ack=5 Win=65536 Len=0 TSval=1
MQTT	93	Subscribe Request (id=9508) [sensor/data/second]
TCP	68	1883 → 37712 [ACK] Seq=5 Ack=64 Win=65536 Len=0 TSval=1
MQTT	73	Subscribe Ack (id=9508)
TCP	68	37712 → 1883 [ACK] Seq=64 Ack=10 Win=65536 Len=0 TSval=
MQTT	170	Publish Message [sensor/data/second]

Gambar 4.31 Muatan pesan pada *port* 1883

Port 1883 mengekspos muatan pesannya dengan *plaintext* seperti yang tertera pada Gambar 4.31. Terlihat bahwa komunikasi *Publish* dan *Subscribe* tertangkap pada paket, bahkan permintaan *subscribe* dan keterangan *publish* pesan pada topik *sensor/data/second* yang digunakan dalam komunikasi. Tidak hanya itu, konten pesan yang dikirim pada *broker* dapat terbaca oleh Wireshark seperti yang tertera pada Gambar 4.32.

192.168.100.7	MQTT	170 Publish Message [sensor/data/second]
192.168.100.7	TCP	68 1883 → 37712 [ACK] Seq=10 Ack=166 Win=
, 170 bytes captured (1360 bits) on interface any, id 0		
0 00 00 08 00		
3 c0 a8 64 07	E.....@..@..c..d..	
b 26 5d 3e 3c	..d..P.[...&]><	
a 4c 01 cd b3	I... ..L...	
3 6f 72 2f 64	L...0d.. sensor/d	
2 74 65 6d 70	ata/seco nd{"temp	
e 32 32 22 2c	erature" : "0.22",	
2 2c 22 74 75	"pH" : "13 .22", "tu	
1 2e 34 38 22	rbidity" : "31.48"	
9 74 79 22 3a	,"conduc tivity":	
	"2276.82 "}	

Gambar 4.32 Konten pesan yang terbaca oleh Wireshark

Berbeda dengan *port* 8883 dengan TLS, muatan pesannya terenkripsi pada setiap komunikasi yang dilakukan sehingga tidak dapat dibaca ataupun ditangkap oleh Wireshark. Hal ini merupakan kerentanan pada keamanan protokol standar MQTT, paket dapat ditangkap

oleh penyerang dan mengambil tindakan pada informasi yang didapat, sehingga penting untuk menerapkan TLS pada setiap *broker* MQTT untuk menjaga kerahasiaan pada proses transmisi data.

4.2.2 Pengujian pada Implementasi Aspek Integritas (*Integrity*)

Untuk menguji validasi pada algoritma *hash* yang dilakukan, maka dilakukan pengujian dengan cara menambah *node* “Hash Validation” untuk melakukan validasi pada pesan yang telah dilakukan *hashing*, kemudian menguji *hash* dengan melakukan *Data Tampering* ketika komunikasi pada *flow* 2 terjadi. *Node function* dilengkapi dengan kode *Javascript* untuk melakukan validasi seperti yang tertera pada Gambar 4.33.

```
var CryptoJS = global.get('CryptoJS');
var receivedData = msg.payload.data;
var receivedHash = msg.payload.hash;
var calculatedHash =
CryptoJS.SHA256(JSON.stringify(receivedData)).toString(CryptoJS.enc.
Hex);

if (receivedHash === calculatedHash) {
  node.status({fill:"green",shape:"dot",text:"Hash valid"});
  msg.valid = true;
} else {
  node.status({fill:"red",shape:"ring",text:"Hash invalid"});
  msg.valid = false;
}
return msg;
```

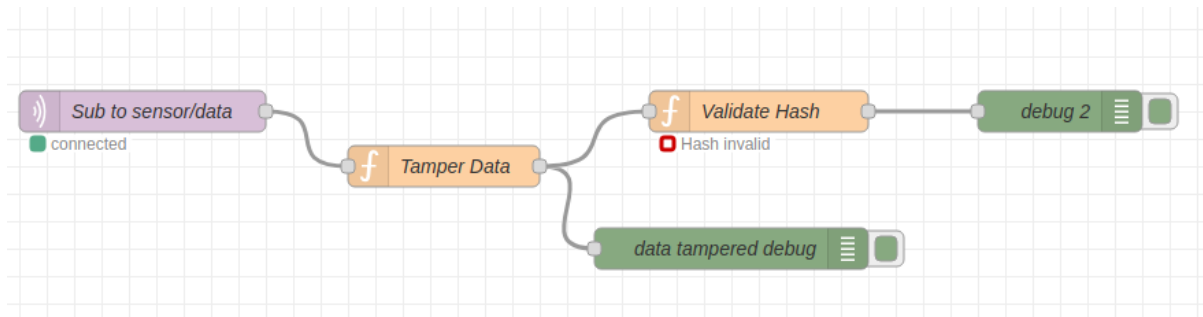
Gambar 4.33 Kode Javascript untuk validasi *hash*

Kode pada Gambar 4.33 berfungsi untuk melakukan komparasi antara *hash* yang diterima dan *hashing* ulang pada pesan. Ketika *hash* yang diterima sesuai dengan *hash* ulang, maka *hash* tersebut valid dan data tidak berubah, menghasilkan peringatan bahwa *hash* valid. Ketika *hash* yang diterima tidak sesuai dengan *hash* ulang, maka *hash* tersebut tidak valid yang berarti data telah diubah atau dimodifikasi, menghasilkan peringatan bahwa *hash* tidak valid.

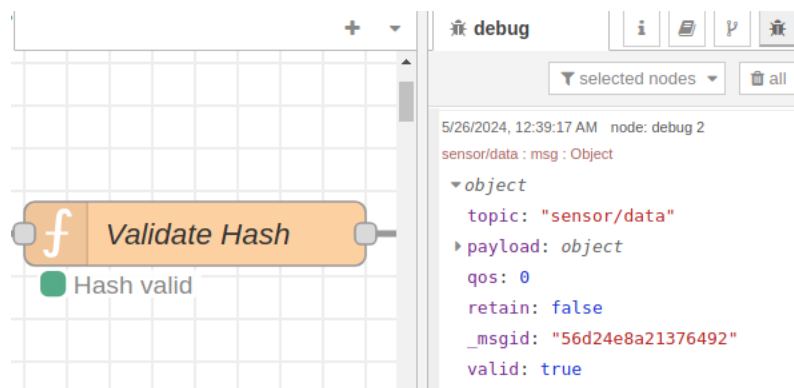
```
var tamperedData = Object.assign({}, msg.payload.data);
tamperedData.temperature = (tamperedData.temperature *
1.5).toFixed(2);
msg.payload.data = tamperedData;
return msg;
```

Gambar 4.34 Kode Javascript untuk melakukan *Data Tampering*

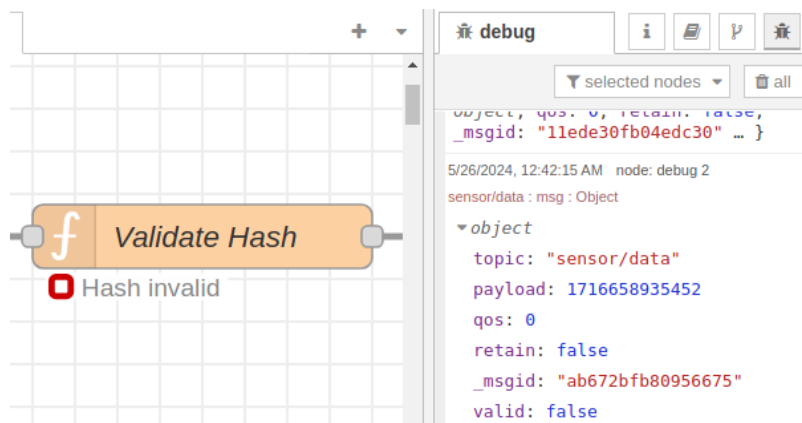
Kode pada Gambar 4.34 berfungsi untuk melakukan modifikasi data secara paksa dengan mengakses objek temperature dari variabel sensorData, sehingga setiap data akan dikalikan dengan 1.5. Hal ini bertujuan untuk menguji validasi hash ketika ada perubahan data. Bentuk *flow 2* untuk melakukan *Data Tampering* adalah seperti yang tertera pada Gambar 4.35.



Gambar 4.35 *Flow 2* ketika melakukan *Data Tampering*



Gambar 4.36 *Debug* ketika *hash* valid



Gambar 4.37 *Debug* ketika *hash* tidak valid

Gambar 4.36 menunjukkan ketika *hash* valid, menandakan bahwa data tidak dirubah atau dimodifikasi ketika data ditransmisikan di antara proses *hash* dan validasi. Tetapi ketika data terindikasi telah diubah atau dimodifikasi, akan muncul peringatan seperti pada Gambar 4.37 yang menunjukkan bahwa *hash* tidak valid. Hal ini terjadi ketika data berubah dari semula proses *hash* dilakukan, sekecil apapun perubahan, akan terdeteksi ketika proses validasi *hash*. Proses validasi ini dapat menjadi indikator bahwa data dapat dipercaya atau tidak validitasnya, dan juga dapat menjadi deteksi ketika ada anomali pada perubahan data, sehingga dapat dilakukan perbaikan dan evaluasi.

4.2.3 Pengujian pada Implementasi Aspek Ketersediaan (*Availability*)

Untuk menguji implementasi aspek kerahasiaan, Snort akan dijalankan untuk melakukan deteksi pada beberapa jenis serangan yang telah ditentukan, bertujuan untuk menguji keandalan Snort dalam mendeteksi serangan untuk mencegah terjadinya kerusakan pada sistem Bank Air Kami v2 karena terjadinya anomali atau serangan yang tidak terdeteksi.

LAND Attack

Serangan *Local Area Network Denial* atau *LAND Attack* merupakan salah satu tipe serangan *Distributed Denial of Service* (DDoS) yang dapat dilakukan oleh penyerang untuk merusak sistem. Pengujian *LAND Attacks* dilakukan dengan *tool* Hping3 pada Kali Linux dengan melakukan perintah pada terminal seperti yang tertera pada Gambar 4.38.

```
# perintah melakukan LAND Attack
sudo hping3 -S -a 192.168.100.7 -p 1883 192.168.100.7 -d 1000
```

Gambar 4.38 Perintah melakukan *LAND Attack*

Pengujian ini akan menargetkan *port* 1883 untuk dikirim pesan sebesar 1000 *bytes* pada alamat IP Raspberry Pi secara terus menerus dengan alamat IP pengirim dan destinasi diatur sama, sehingga ketika mesin mencoba untuk membalas akan memasuki perputaran, secara terus menerus mengirimkan balasan yang pada akhirnya menyebabkan kegagalan sistem atau *crash*.

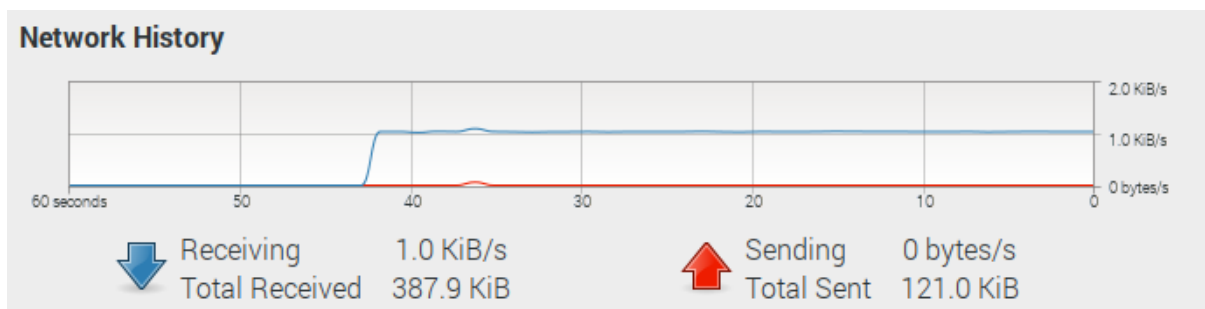
No.	Time	Source	Destination	Protocol	Length	Info
170	681.609345708	192.168.100.7	192.168.100.7	MQTT	1056	Publish Received (id=22616)
171	682.623115515	192.168.100.7	192.168.100.7	MQTT	1056	Publish Received (id=22616)
172	683.632471405	192.168.100.7	192.168.100.7	MQTT	1056	Publish Received (id=22616)
173	684.634804694	192.168.100.7	192.168.100.7	MQTT	1056	Publish Received (id=22616)
174	685.640276454	192.168.100.7	192.168.100.7	MQTT	1056	Publish Received (id=22616)
175	686.651954248	192.168.100.7	192.168.100.7	MQTT	1056	Publish Received (id=22616)
176	687.654319590	192.168.100.7	192.168.100.7	MQTT	1056	Publish Received (id=22616)
177	688.654307336	192.168.100.7	192.168.100.7	MQTT	1056	Publish Received (id=22616)
178	689.654783598	192.168.100.7	192.168.100.7	MQTT	1056	Publish Received (id=22616)
179	690.659150618	192.168.100.7	192.168.100.7	MQTT	1056	Publish Received (id=22616)
180	691.665870227	192.168.100.7	192.168.100.7	MQTT	1056	Publish Received (id=22616)
181	692.665596220	192.168.100.7	192.168.100.7	MQTT	1056	Publish Received (id=22616)

Frame 177: 1056 bytes on wire (8448 bits), 1056 bytes captured (8448 bits) on interface any, id 0									
Linux cooked capture v1									
Internet Protocol Version 4, Src: 192.168.100.7, Dst: 192.168.100.7									
Transmission Control Protocol, Src Port: 2061, Dst Port: 1883, Seq: 0, Len: 1000									
MQ Telemetry Transport Protocol, Publish Received									
MQ Telemetry Transport Protocol, Publish Received									
MQ Telemetry Transport Protocol, Publish Received									
MQ Telemetry Transport Protocol, Publish Received									
MQ Telemetry Transport Protocol, Publish Received									
MQ Telemetry Transport Protocol, Publish Received									

0060	58 58 58 58 58 58 58 58	58 58 58 58 58 58 58 58	XXXXXXXX XXXXXXXX
0070	58 58 58 58 58 58 58 58	58 58 58 58 58 58 58 58	XXXXXXXX XXXXXXXX
0080	58 58 58 58 58 58 58 58	58 58 58 58 58 58 58 58	XXXXXXXX XXXXXXXX
0090	58 58 58 58 58 58 58 58	58 58 58 58 58 58 58 58	XXXXXXXX XXXXXXXX
00a0	58 58 58 58 58 58 58 58	58 58 58 58 58 58 58 58	XXXXXXXX XXXXXXXX
00b0	58 58 58 58 58 58 58 58	58 58 58 58 58 58 58 58	XXXXXXXX XXXXXXXX
00c0	58 58 58 58 58 58 58 58	58 58 58 58 58 58 58 58	XXXXXXXX XXXXXXXX
00d0	58 58 58 58 58 58 58 58	58 58 58 58 58 58 58 58	XXXXXXXX XXXXXXXX
00e0	58 58 58 58 58 58 58 58	58 58 58 58 58 58 58 58	XXXXXXXX XXXXXXXX
00f0	58 58 58 58 58 58 58 58	58 58 58 58 58 58 58 58	XXXXXXXX XXXXXXXX
0100	58 58 58 58 58 58 58 58	58 58 58 58 58 58 58 58	XXXXXXXX XXXXXXXX
0110	58 58 58 58 58 58 58 58	58 58 58 58 58 58 58 58	XXXXXXXX XXXXXXXX
0120	58 58 58 58 58 58 58 58	58 58 58 58 58 58 58 58	XXXXXXXX XXXXXXXX
0130	58 58 58 58 58 58 58 58	58 58 58 58 58 58 58 58	XXXXXXXX XXXXXXXX
0140	58 58 58 58 58 58 58 58	58 58 58 58 58 58 58 58	XXXXXXXX XXXXXXXX

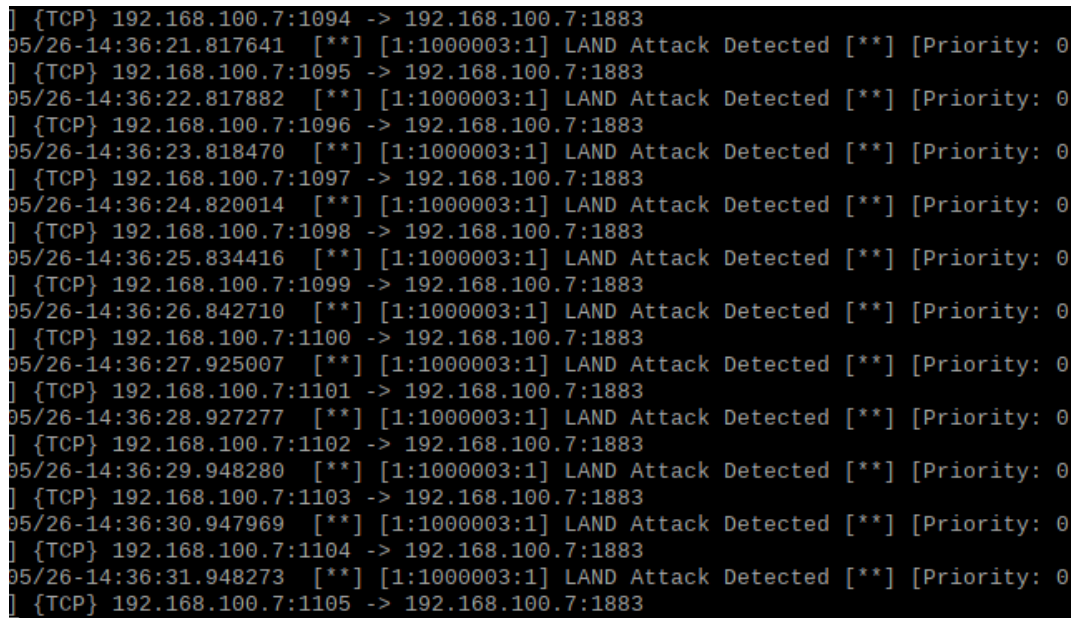
Gambar 4.39 Tangkapan paket ketika pengujian *LAND Attack*

Gambar 4.39 menunjukkan tangkapan paket oleh Wireshark ketika *LAND Attack* dilakukan, terlihat bahwa pesan yang diterima sepanjang 1000 *bytes* pada satu kali pengiriman dan mesin akan membalas pesan tersebut secara terus menerus.

Gambar 4.40 Lalu lintas jaringan ketika pengujian *LAND Attack*

Gambar 4.40 menunjukkan monitor sistem Raspberry Pi, terlihat bahwa ketika *LAND Attack* dilakukan terjadi lonjakan pada lalu lintas jaringan yang signifikan jika dibandingkan sebelum *LAND Attack* dilakukan. Ketika penyerang mengirim serangan dengan ukuran yang jauh lebih besar, maka hal ini dapat menyebabkan sistem kewalahan dengan lalu lintas jaringan yang padat, sehingga mempengaruhi performa sistem secara keseluruhan dan menyebabkan

kegagalan pada skenario terburuk yang mungkin terjadi. Maka dari itu, peran Snort untuk mendeteksi adanya *LAND Attack* diperlukan pada kasus ini.



```

[ {TCP} 192.168.100.7:1094 -> 192.168.100.7:1883
05/26-14:36:21.817641  [**] [1:1000003:1] LAND Attack Detected [**] [Priority: 0
[ {TCP} 192.168.100.7:1095 -> 192.168.100.7:1883
05/26-14:36:22.817882  [**] [1:1000003:1] LAND Attack Detected [**] [Priority: 0
[ {TCP} 192.168.100.7:1096 -> 192.168.100.7:1883
05/26-14:36:23.818470  [**] [1:1000003:1] LAND Attack Detected [**] [Priority: 0
[ {TCP} 192.168.100.7:1097 -> 192.168.100.7:1883
05/26-14:36:24.820014  [**] [1:1000003:1] LAND Attack Detected [**] [Priority: 0
[ {TCP} 192.168.100.7:1098 -> 192.168.100.7:1883
05/26-14:36:25.834416  [**] [1:1000003:1] LAND Attack Detected [**] [Priority: 0
[ {TCP} 192.168.100.7:1099 -> 192.168.100.7:1883
05/26-14:36:26.842710  [**] [1:1000003:1] LAND Attack Detected [**] [Priority: 0
[ {TCP} 192.168.100.7:1100 -> 192.168.100.7:1883
05/26-14:36:27.925007  [**] [1:1000003:1] LAND Attack Detected [**] [Priority: 0
[ {TCP} 192.168.100.7:1101 -> 192.168.100.7:1883
05/26-14:36:28.927277  [**] [1:1000003:1] LAND Attack Detected [**] [Priority: 0
[ {TCP} 192.168.100.7:1102 -> 192.168.100.7:1883
05/26-14:36:29.948280  [**] [1:1000003:1] LAND Attack Detected [**] [Priority: 0
[ {TCP} 192.168.100.7:1103 -> 192.168.100.7:1883
05/26-14:36:30.947969  [**] [1:1000003:1] LAND Attack Detected [**] [Priority: 0
[ {TCP} 192.168.100.7:1104 -> 192.168.100.7:1883
05/26-14:36:31.948273  [**] [1:1000003:1] LAND Attack Detected [**] [Priority: 0
[ {TCP} 192.168.100.7:1105 -> 192.168.100.7:1883

```

Gambar 4.41 Deteksi Snort pada *LAND Attack*

Deteksi Snort pada *LAND Attack* tertera pada Gambar 4.41. Snort mendeteksi adanya *LAND Attack* secara cepat ketika terdapat anomali pada pesan yang diterima oleh sistem, terlihat bahwa penyerang menargetkan *port* 1883 untuk melakukan serangan yang terdeteksi oleh Snort. Hal ini dapat menjadi indikator bahwa penyerang mencoba untuk menggagalkan sistem dengan *LAND Attack*, sehingga pengelola dapat langsung melakukan tindakan untuk mengatasi atau mencegah hal tersebut.

SYN Flood Attack

Pengujian *Distributed Denial of Service* (DDoS) dengan *SYN Flood Attack* dilakukan menggunakan *tool* Hping3 pada Kali Linux dengan menargetkan alamat IP Raspberry Pi pada *port* 1883. Perintah yang dilakukan pada terminal adalah seperti yang tertera pada Gambar 4.42.

```

# perintah melakukan SYN Flood Attack

sudo hping3 --flood -S -p 1883 192.168.100.7

```

Gambar 4.42 Perintah melakukan *SYN Flood Attack*

Untuk melihat pesan yang diterima oleh sistem dan performa sistem setelah dilakukan serangan, maka penangkapan paket dan monitor sistem pada Raspberry Pi dilakukan.

No.	Time	Source	Destination	Protocol	Length	Info
4470874	222.743585032	192.168.100.4	192.168.100.7	TCP	62	[TCP Port numbers reused] 41371 → 1883 [SYN] Seq=0 Win=51
4470875	222.743585073	192.168.100.4	192.168.100.7	TCP	62	36367 → 1883 [RST] Seq=1 Win=0 Len=0
4470876	222.743585093	192.168.100.4	192.168.100.7	TCP	62	36368 → 1883 [RST] Seq=1 Win=0 Len=0
4470877	222.743585123	192.168.100.4	192.168.100.7	TCP	62	36369 → 1883 [RST] Seq=1 Win=0 Len=0
4470878	222.743585143	192.168.100.4	192.168.100.7	TCP	62	[TCP Port numbers reused] 41372 → 1883 [SYN] Seq=0 Win=51
4470879	222.743585173	192.168.100.4	192.168.100.7	TCP	62	[TCP Port numbers reused] 41373 → 1883 [SYN] Seq=0 Win=51
4470880	222.743585203	192.168.100.4	192.168.100.7	TCP	62	36371 → 1883 [RST] Seq=1 Win=0 Len=0
4470881	222.743593449	192.168.100.7	192.168.100.4	TCP	60	1883 → 41371 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1
4470882	222.743602367	192.168.100.7	192.168.100.4	TCP	60	1883 → 41372 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1
4470883	222.743609651	192.168.100.7	192.168.100.4	TCP	60	1883 → 41373 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1
4470884	222.743623519	192.168.100.4	192.168.100.7	TCP	62	36372 → 1883 [RST] Seq=1 Win=0 Len=0
4470885	222.743623539	192.168.100.4	192.168.100.7	TCP	62	36373 → 1883 [RST] Seq=1 Win=0 Len=0
4470886	222.743623569	192.168.100.4	192.168.100.7	TCP	62	36374 → 1883 [RST] Seq=1 Win=0 Len=0
4470887	222.743623589	192.168.100.4	192.168.100.7	TCP	62	36375 → 1883 [RST] Seq=1 Win=0 Len=0
4470888	222.743623619	192.168.100.4	192.168.100.7	TCP	62	36376 → 1883 [RST] Seq=1 Win=0 Len=0

Gambar 4.43 Tangkapan paket ketika pengujian *SYN Flood Attack*Gambar 4.44 Sistem monitor ketika pengujian *SYN Flood Attack*

Tangkapan paket ketika pengujian *SYN Flood Attack* tertera pada Gambar 4.43, penyerang mengirim pesan secara cepat dan terus menerus, terlihat juga bahwa ada sekitar 4

juta paket tertangkap oleh Wireshark dalam hanya beberapa saat. Performa *CPU* dan lalu lintas jaringan mengalami kenaikan yang signifikan seperti yang tertera pada Gambar 4.44. Terlihat bahwa kinerja ketiga *CPU* mengalami kenaikan hingga 100% hanya dalam 5 detik dan terus bekerja selama pengujian berlangsung. Peran Snort untuk mendeteksi *SYN Flood Attack* secara cepat diperlukan pada kasus ini.

```

ty: 0] {TCP} 192.168.100.4:20721 -> 192.168.100.7:1883
05/26-16:12:23.672535  **] [1:1000004:1] SYN Flood Attack Detected  **] [Priori
ty: 0] {TCP} 192.168.100.4:20722 -> 192.168.100.7:1883
05/26-16:12:23.672535  **] [1:1000004:1] SYN Flood Attack Detected  **] [Priori
ty: 0] {TCP} 192.168.100.4:20723 -> 192.168.100.7:1883
05/26-16:12:23.672535  **] [1:1000004:1] SYN Flood Attack Detected  **] [Priori
ty: 0] {TCP} 192.168.100.4:20724 -> 192.168.100.7:1883
05/26-16:12:23.672535  **] [1:1000004:1] SYN Flood Attack Detected  **] [Priori
ty: 0] {TCP} 192.168.100.4:20725 -> 192.168.100.7:1883
05/26-16:12:23.672535  **] [1:1000004:1] SYN Flood Attack Detected  **] [Priori
ty: 0] {TCP} 192.168.100.4:20726 -> 192.168.100.7:1883
05/26-16:12:23.672535  **] [1:1000004:1] SYN Flood Attack Detected  **] [Priori
ty: 0] {TCP} 192.168.100.4:20727 -> 192.168.100.7:1883
05/26-16:12:23.672588  **] [1:1000004:1] SYN Flood Attack Detected  **] [Priori
ty: 0] {TCP} 192.168.100.4:20728 -> 192.168.100.7:1883
05/26-16:12:23.672588  **] [1:1000004:1] SYN Flood Attack Detected  **] [Priori
ty: 0] {TCP} 192.168.100.4:20729 -> 192.168.100.7:1883
05/26-16:12:23.672588  **] [1:1000004:1] SYN Flood Attack Detected  **] [Priori
ty: 0] {TCP} 192.168.100.4:20730 -> 192.168.100.7:1883
05/26-16:12:23.672643  **] [1:1000004:1] SYN Flood Attack Detected  **] [Priori
ty: 0] {TCP} 192.168.100.4:20731 -> 192.168.100.7:1883
05/26-16:12:23.672759  **] [1:1000004:1] SYN Flood Attack Detected  **] [Priori
ty: 0] {TCP} 192.168.100.4:20732 -> 192.168.100.7:1883

```

Gambar 4.45 Deteksi Snort pada *SYN Flood Attack*

Gambar 4.45 menunjukkan deteksi Snort pada *SYN Flood Attack*. Snort mendeteksi serangan dengan cepat dengan frekuensi waktu yang sangat cepat, menandakan bahwa *SYN Flood Attack* mengirim pesan dengan sangat cepat. Hal ini dapat dengan cepat menyebabkan kegagalan pada sistem atau *crash*. Maka dari itu, deteksi Snort dapat menjadi peringatan untuk segera melakukan tindakan pada jenis serangan tersebut.

Smurf Attack

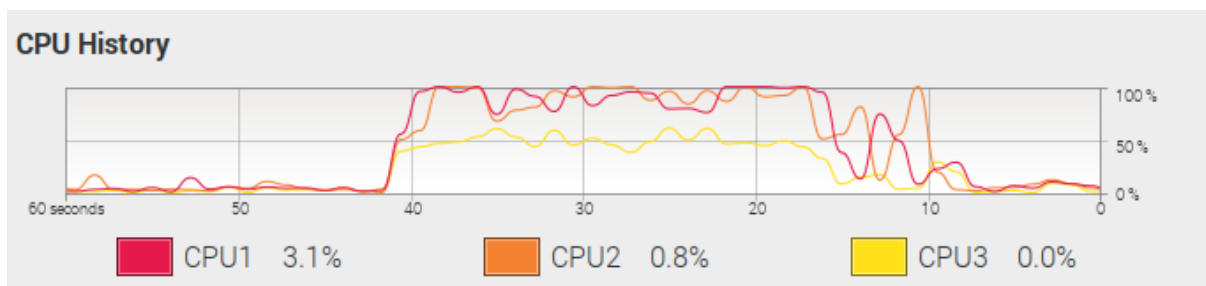
Smurf Attack merupakan salah satu jenis serangan *Distributed Denial of Service* (DDoS) yang identik dengan *SYN Flood Attack*. Smurf Attack menyerang melalui protokol ICMP untuk menyerang sistem dengan mengirimkan permintaan ping berskala besar. Pengujian dilakukan menggunakan *tool* Hping3 dengan memberikan perintah pada Kali Linux seperti yang tertera pada Gambar 4.46.

```
# perintah melakukan Smurf Attack
sudo hping3 --icmp --flood 192.168.100.7 -a 192.168.100.7
```

Gambar 4.46 Perintah melakukan *Smurf Attack*

Untuk melihat pesan yang diterima oleh sistem dan performa sistem setelah dilakukan serangan, maka penangkapan paket dan monitor sistem pada Raspberry Pi dilakukan.

3315357	351.357609087	192.168.100.7	192.168.100.7	ICMP	62 Echo (ping) request
3315358	351.357664652	192.168.100.7	192.168.100.7	ICMP	62 Echo (ping) request
3315359	351.357664682	192.168.100.7	192.168.100.7	ICMP	62 Echo (ping) request
3315360	351.357664703	192.168.100.7	192.168.100.7	ICMP	62 Echo (ping) request
3315361	351.357735396	192.168.100.7	192.168.100.7	ICMP	62 Echo (ping) request
3315362	351.357735416	192.168.100.7	192.168.100.7	ICMP	62 Echo (ping) request
3315363	351.357735446	192.168.100.7	192.168.100.7	ICMP	62 Echo (ping) request
3315364	351.357735476	192.168.100.7	192.168.100.7	ICMP	62 Echo (ping) request
3315365	351.357735496	192.168.100.7	192.168.100.7	ICMP	62 Echo (ping) request
3315366	351.357735526	192.168.100.7	192.168.100.7	ICMP	62 Echo (ping) request
3315367	351.357813023	192.168.100.7	192.168.100.7	ICMP	62 Echo (ping) request
3315368	351.357813072	192.168.100.7	192.168.100.7	ICMP	62 Echo (ping) request

Gambar 4.47 Tangkapan paket ketika pengujian *Smurf Attack*Gambar 4.48 Sistem monitor ketika pengujian *Smurf Attack*

Sama halnya *SYN Flood Attack*, *Smurf Attack* mengirim paket dengan skala yang besar dengan waktu yang singkat seperti pada Gambar 4.47 dan juga memberikan dampak yang sama pada sistem seperti pada Gambar 4.48. Jenis serangan tersebut mampu menyebabkan kegagalan sistem yang berdampak pada berhentinya transmisi data karena sistem yang diberi beban terlalu banyak dan lalu lintas jaringan yang padat.

```

05/27-16:52:44.558288  [**] [1:100000023:1] Smurf Flooding Attack Detected [**]
[Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.7
05/27-16:52:44.558573  [**] [1:100000023:1] Smurf Flooding Attack Detected [**]
[Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.7
05/27-16:52:44.558834  [**] [1:100000023:1] Smurf Flooding Attack Detected [**]
[Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.7
05/27-16:52:44.559108  [**] [1:100000023:1] Smurf Flooding Attack Detected [**]
[Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.7
05/27-16:52:44.559331  [**] [1:100000023:1] Smurf Flooding Attack Detected [**]
[Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.7
05/27-16:52:44.559617  [**] [1:100000023:1] Smurf Flooding Attack Detected [**]
[Priority: 0] {ICMP} 192.168.100.7 -> 192.168.100.7

```

Gambar 4.49 Deteksi Snort pada *Smurf Attack*

Deteksi Snort pada *Smurf Attack* tertera pada Gambar 4.49, terlihat bahwa karakteristik *Smurf Attack* adalah menjadikan alamat IP target sebagai pengirim dan penerima, hal yang sama terjadi pada *LAND Attack*. Snort mendeteksi serangan dengan frekuensi waktu yang banyak, menandakan bahwa pesan yang dikirim berskala besar dalam waktu singkat.

TCP ACK Scan

TCP ACK Scan merupakan jenis serangan *Port Scanning* untuk melakukan pemindaian terhadap *port* yang terbuka pada alamat IP Raspberry Pi, dan juga merupakan bagian dari jenis serangan yang dapat menyerang aspek kerahasiaan. Pengujian dilakukan menggunakan *tool nmap* pada Kali Linux untuk melakukan pemindaian. Perintah untuk melakukan pemindaian pada terminal seperti pada Gambar 4.50.

```

# perintah melakukan TCP ACK Scan
sudo nmap -sA 192.168.100.7

```

Gambar 4.50 Perintah melakukan *TCP ACK Scan*

Penyerang akan mendapatkan informasi mengenai *port* yang tersaring oleh *firewall* dan tidak, dan *port* yang terbuka dan tidak terbuka. Hal ini dapat dijadikan titik kerentanan bagi penyerang, sehingga penyerang dapat melancarkan serangan yang lainnya seperti DDoS dan MiTM. *TCP ACK Scan* bekerja tidak terlalu agresif, sehingga IDS akan kesulitan dalam mendeteksi adanya pemindaian. Maka dari itu, Snort dilengkapi dengan *rule* untuk mendeteksi *TCP ACK Scan* untuk mencegah penyerang mengumpulkan informasi yang lebih detail terhadap jaringan.

```

05/27-22:27:09.523502  [**] [1:1000005:1] TCP ACK Scan Detected [**] [Priority:
0] {TCP} 192.168.100.4:47344 -> 192.168.100.7:1580
05/27-22:27:09.523502  [**] [1:1000005:1] TCP ACK Scan Detected [**] [Priority:
0] {TCP} 192.168.100.4:47344 -> 192.168.100.7:31038
05/27-22:27:09.523576  [**] [1:1000005:1] TCP ACK Scan Detected [**] [Priority:
0] {TCP} 192.168.100.4:47344 -> 192.168.100.7:5988
05/27-22:27:09.523577  [**] [1:1000005:1] TCP ACK Scan Detected [**] [Priority:
0] {TCP} 192.168.100.4:47344 -> 192.168.100.7:4224
05/27-22:27:09.523577  [**] [1:1000005:1] TCP ACK Scan Detected [**] [Priority:
0] {TCP} 192.168.100.4:47344 -> 192.168.100.7:668
05/27-22:27:09.523577  [**] [1:1000005:1] TCP ACK Scan Detected [**] [Priority:
0] {TCP} 192.168.100.4:47344 -> 192.168.100.7:1002
05/27-22:27:09.523577  [**] [1:1000005:1] TCP ACK Scan Detected [**] [Priority:
0] {TCP} 192.168.100.4:47344 -> 192.168.100.7:62078
05/27-22:27:09.523578  [**] [1:1000005:1] TCP ACK Scan Detected [**] [Priority:
0] {TCP} 192.168.100.4:47344 -> 192.168.100.7:15660

```

Gambar 4.51 Deteksi Snort pada *TCP ACK Scan*

Deteksi Snort pada *TCP ACK Scan* tertera pada Gambar 4.51, terlihat bahwa alamat IP penyerang mencoba melakukan pemindaian terhadap alamat IP Raspberry Pi. Hal ini dapat menjadi indikator untuk menerapkan *firewall* pada sistem dan *port* untuk mencegah penyerang menggali lebih banyak informasi dan meluncurkan serangan yang lainnya.

Pengujian *Firewall*

Pengujian *Firewall* berfokus pada penolakan perangkat untuk masuk dan berkomunikasi dengan Raspberry Pi. Bertujuan untuk menguji konfigurasi *ufw* pada komunikasi terhadap *port* 1883 sebagai *broker* MQTT. Pengujian dilakukan dengan percobaan permintaan *subscribe* pada topik tertentu menggunakan Kali Linux.

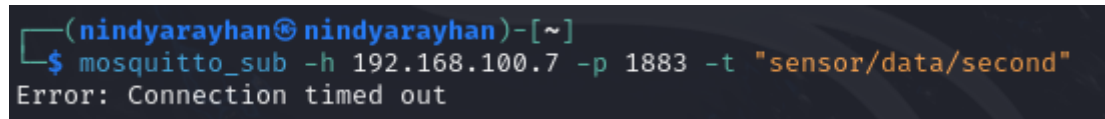
```

(nindyarayhan@nindyarayhan)-[~]
$ mosquitto_sub -h 192.168.100.7 -p 1883 -t "sensor/data/second"
{"data":{"temperature":"17.70","pH":"0.04","turbidity":"75.27","conductivity":
"1783.20"},"hash":"ac874afed41b8af5cd2cf387c271de213f04e473f337b39bc3dbfee29
6849628"}
{"data":{"temperature":"24.71","pH":"2.48","turbidity":"98.65","conductivity"
:"4976.87"},"hash":"e5089e0e726f6d0c37c46b8330de85b99632f169aefc073e9a3d6b81b
6b00e42"}
{"data":{"temperature":"23.19","pH":"4.39","turbidity":"28.29","conductivity"
:"4712.28"},"hash":"395f1f543b5992195503e6c411460e70f58c2616e2d019a6a20c34690
80a4121"}
{"data":{"temperature":"2.02","pH":"1.61","turbidity":"86.31","conductivity":
"1472.89"},"hash":"a33b91192a20f96db7b10c50da6b3faadfe6d0f8bc7383419b6cacb3a8
45cfd8"}

```

Gambar 4.52 Permintaan *subscribe* pada topik

Gambar 4.52 menunjukkan permintaan *subscribe* pada topik “*sensor/data/second*” pada *port* 1883 ketika *firewall* tidak dikonfigurasi untuk menolak masuk komunikasi pada *port* 1883, terlihat bahwa Kali Linux dapat melihat transmisi data yang di-*publish* oleh Raspberry Pi.



```
(nindyarayhan@nindyarayhan)-[~]
$ mosquitto_sub -h 192.168.100.7 -p 1883 -t "sensor/data/second"
Error: Connection timed out
```

Gambar 4.53 Permintaan *subscribe* tidak diterima

Ketika *firewall* dikonfigurasi untuk menolak masuk komunikasi pada *port* 1883, maka permintaan *subscribe* tidak dapat dilakukan pada Kali Linux, sehingga muncul pesan *error* seperti yang tertera pada Gambar 4.53. Dengan menerapkan *firewall* untuk tidak mengizinkan perangkat dari luar berkomunikasi dengan sistem, maka kemungkinan penyerang untuk menyerang akan semakin kecil.

4.3 Hasil Implementasi dan Pengujian

Setelah melakukan implementasi pada desain keamanan dan melakukan pengujian untuk menguji fungsionalitas desain keamanan, maka akan dituliskan hasil dari implementasi pada tabel berupa *checklist* pada desain yang telah terimplementasi. Parameter yang telah berhasil terimplementasi dan dilakukan pengujian akan ditandai dengan keterangan diterima.

Tabel 4.1 Hasil Implementasi

Aspek	Parameter	Hasil	Keterangan
Kerahasiaan (<i>Confidentiality</i>)	Menerapkan TLS pada <i>broker</i> MQTT	TLS diterapkan pada <i>broker</i> MQTT pada <i>port</i> 8883 dan setiap pesan pada komunikasi terenkripsi.	Diterima
Integritas (<i>Integrity</i>)	Menggunakan Node-RED untuk kontrol data	Dapat melakukan <i>generate</i> data dan komunikasi dengan <i>broker</i> MQTT.	Diterima
	Menerapkan algoritma <i>hash</i> pada pesan	Diterapkan pada data yang akan di- <i>publish</i> dan melakukan validasi <i>hash</i>	Diterima

		yang dilakukan dengan Node-RED.	
Ketersediaan (<i>Availability</i>)	Mengimplementasi Snort IDS pada Raspberry Pi	Dapat mendeteksi serangan DDoS berupa <i>LAND Attack</i> , <i>SYN Flood Attack</i> , <i>Smurf Attack</i> , dan <i>Port Scanning</i> berupa <i>TCP ACK Scan</i> .	Diterima
	Implementasi <i>Firewall</i> pada Raspberry Pi	Dapat memblokir perangkat lain untuk berkomunikasi dengan <i>broker</i> MQTT.	Diterima

4.4 Kekurangan Sistem

Implementasi dan pengujian desain keamanan telah dilakukan. Akan tetapi, beberapa kelemahan teridentifikasi setelah memperoleh hasil implementasi dan pengujian. Berikut kelemahan-kelemahan yang terdapat pada desain keamanan dalam implementasi dan pengujiannya:

Tabel 4.2 Kekurangan Sistem

Implementasi	Pengujian	Kelemahan
Integrasi <i>hashing</i> dengan data (Aspek Integritas)	Melakukan validasi <i>hash</i>	Data yang ditransmisikan tidak berasal dari sensor sesungguhnya, melainkan data hasil <i>generate</i> oleh Node-RED yang berasal dari Raspberry Pi OS, sehingga implementasi <i>hashing</i> kurang akurat karena data sesungguhnya diperoleh dari ESP32.
Implementasi Snort IDS	Simulasi serangan untuk pengujian deteksi	Snort IDS hanya akan memberi peringatan ketika ada serangan, sehingga tindakan tidak dilakukan secara otomatis.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Desain keamanan telah berhasil diimplementasi dan dilakukan pengujian untuk menguji fungsionalitas dan keandalan dalam mengatasi serangan yang mungkin terjadi pada sistem Bank Air Kami v2. Dengan berlandaskan trias CIA (*Confidentiality, Integrity, Availability*) dalam menciptakan fitur-fitur keamanan, desain keamanan yang telah dirancang mampu menjangkau berbagai aspek keamanan pada rencana pengamanan dan kerentanan yang terdapat pada sistem. Implementasi Aspek Kerahasiaan (*Confidentiality*) dengan mengimplementasi TLS pada *broker* MQTT yang mampu menjaga kerahasiaan komunikasi data, implementasi Aspek Integritas (*Integrity*) dengan mengimplementasi algoritma *hash* pada pesan dan validasi untuk memastikan kepercayaan atas data, dan implementasi Aspek Ketersediaan (*Availability*) dengan mengimplementasi deteksi serangan jaringan dengan Snort IDS yang mampu mendeteksi serangan DDoS dan konfigurasi *Firewall* untuk melindungi perangkat dari serangan. Maka dari itu, acuan terhadap ketiga elemen trias CIA (*Confidentiality, Integrity, Availability*) mampu menciptakan solusi untuk permasalahan keamanan yang terdapat pada sistem *Internet of Things* (IoT) ataupun sistem aplikasi dan organisasi.

5.2 Saran

Dalam menciptakan desain keamanan dan implementasi serta pengujiannya, peneliti masih mendapati beberapa kekurangan untuk dilakukan pengembangan lebih lanjut. Beberapa hal yang dapat dikembangkan pada desain keamanan dan implementasinya adalah sebagai berikut:

- a. Implementasi dan pengujian desain keamanan diharapkan untuk dilakukan pada sistem Bank Air Kami secara langsung untuk menguji keakuratan fitur-fitur keamanan, salah satunya adalah penerapan *hashing* yang dilakukan ketika data langsung diperoleh untuk meningkatkan integritas data.
- b. Diharapkan untuk mengimplementasi *Intrusion Prevention System* (IPS) menggunakan Snort ataupun alat lain untuk memperkuat fitur keamanan pada sistem Bank Air Kami, sehingga ketika serangan terjadi, sistem akan langsung mengambil tindakan pencegahan.

DAFTAR PUSTAKA

- Abdullah, A., Hamad, R., Abdulrahman, M., Moala, H., & Elkhediri, S. (2019). CyberSecurity: A Review of Internet of Things (IoT) Security Issues, Challenges and Techniques. *CCAIS 2019 : 2nd International Conference on Computer Applications & Information Security*, 533.
- Andrea, I., Chrysostomou, C., & Hadjiscrhistofi, G. (2015). Internet of Things: Security Vulnerabilities and Challenges. *The 3rd IEEE ISCC 2015 International Workshop on Smart City and Ubiquitous Computing Applications*, 180–187.
- Ashoor, A. S., & Gore, S. (2010). Importance of Intrusion Detection System (IDS). *International Journal of Scientific and Engineering Research*, 2(1). <http://www.ijser.org>
- Atmoko, R. A., Riantini, R., & Hasin, M. K. (2017). IoT real time data acquisition using MQTT protocol. *Journal of Physics: Conference Series*, 853(1). <https://doi.org/10.1088/1742-6596/853/1/012003>
- Bierman, E., Cloete, E., & Venter, L. M. (2001). A Comparison of Intrusion Detection System. *Computers & Security*, 676–683.
- Bintarawati, F. (2024). *ANAYASA (Journal of Legal Studies) THE INFLUENCE OF THE PERSONAL DATA PROTECTION LAW (UU PDP) ON LAW ENFORCEMENT IN THE DIGITAL ERA. 1(2)*.
- Boeckl, K., Fagan, M., Fisher, W., Lefkovitz, N., Megas, K. N., Nadeau, E., O'Rourke, D. G., Piccarreta, B., & Scarfone, K. (2019). *Considerations for managing Internet of Things (IoT) cybersecurity and privacy risks*. <https://doi.org/10.6028/NIST.IR.8228>
- Butun, I., Österberg, P., & Song, H. (2019). *Security of the Internet of Things: Vulnerabilities, Attacks and Countermeasures*. <https://doi.org/10.1109/COMST.2019.2953364>
- Chanthakoummane, Y., Saiyod, S., Benjamas, N., Khamphakdee, N., & Chaichawananit, J. (2016). Improving Intrusion Detection on Snort Rules for Botnet Detection. *Convergence Security, 1*, 1–20. <https://doi.org/10.13140/RG.2.1.1418.9040>
- Chen, F., Huo, Y., Zhu, J., & Fan, D. (2020). A Review on the Study on MQTT Security Challenge. *Proceedings - 2020 IEEE International Conference on Smart Cloud, SmartCloud 2020*, 128–133. <https://doi.org/10.1109/SmartCloud49737.2020.00032>
- Craigen, D., Diakun-Thibault, N., & Purse, R. (2014). *Technology Innovation Management Review Defining Cybersecurity*. www.timreview.ca

- Ferencz, K., Domokos, J., Jubileumi, X., Konferencia, K., & Domokos, J. (2020). *Using Node-RED platform in an industrial environment*. <https://www.researchgate.net/publication/339596157>
- Hoffman, L., & Zahadat, N. (2018). Securing Democracy: A Comparative Look at Modern and Future US Voting Systems Through the Lens of the CIA Triad. In *Journal of Information Assurance and Security* (Vol. 13). www.mirlabs.net/jias/index.html
- Khandare, S. V., Herlekar, V. P., Hanwate, V. S., Shirale, G. M., & Sirbhate, D. D. (2023). A Global Overview of Data Security, Safety, Corporate Data Privacy, and Data Protection. *Intelligent Computing and Networking*, 699 LNNS, 437–451. https://doi.org/10.1007/978-981-99-3177-4_6
- Kornita, S. E. (2020). Strategi Pemenuhan Kebutuhan Masyarakat terhadap Air Bersih di Kabupaten Bengkalis. *Jurnal Samudra Ekonomi Dan Bisnis*, 11(2), 166–181. <https://doi.org/10.33059/jseb.v11i2.1883>
- Kumar, V., & Prakash Sangwan, O. (2012). Signature Based Intrusion Detection System Using SNORT. In *International Journal of Computer Applications & Information Technology: Vol. I*. www.ijcait.com
- Kurniawati, R. D., Kraar, M. H., Amalia, V. N., & Kusaeri, M. T. (2020). PENINGKATAN AKSES AIR BERSIH MELALUI SOSIALISASI DAN PENYARINGAN AIR SEDERHANA DESA HAURPUGUR. *Jurnal Pengabdian Dan Peningkatan Mutu Masyarakat (JANAYU)*, 1(2). <https://doi.org/10.22219/janayu.v1i2.11784>
- Light, R. A. (2017). Mosquitto: server and client implementation of the MQTT protocol. *The Journal of Open Source Software*, 2(13), 265. <https://doi.org/10.21105/joss.00265>
- Lomi, R. A., Messakh, J. J., & Tamelan, P. G. (2021). PEMANFAATAN AIR BERSIH UNTUK KEBUTUHAN RUMAH TANGGA DARI MATA AIR OELNAISANAM DI KELURAHAN BAKUNASE II, KOTA KUPANG UTILIZATION OF CLEAN WATER FOR HOUSEHOLD NEEDS FROM THE OELNAISANAM SPRING IN BAKUNASE II VILLAGE, KUPANG CITY. In *Jurnal Batakarang* (Vol. 2, Issue 1).
- Mangla, M., Ambarkar, S., Akhare, R., Deokar, S., Mohanty, S. N., & Satpathy, S. (2022). A Proposed Framework to Achieve CIA in IoT Networks. *Lecture Notes in Electrical Engineering*, 837, 19–30. https://doi.org/10.1007/978-981-16-8546-0_3
- Meneghello, F., Calore, M., Zucchetto, D., Polese, M., & Zanella, A. (2019). *IoT: Internet of Threats? A survey of practical security vulnerabilities in real IoT devices*.

- Moraes, T., Nogueira, B., Lira, V., & Tavares, E. (2019). Performance comparison of iot communication protocols. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics, 2019-October*, 3249–3254. <https://doi.org/10.1109/SMC.2019.8914552>
- Nisa, K., Hayat, D., Malang, I., Tata, J., Perumahan Galaxy, S., Dinoyo, R., Lowokwaru, K., Malang, K., & Timur, J. (2023). Urgensi Pengesahan Undang-Undang Perlindungan Data Pribadi oleh DPR RI. In *Realism: Law Review* (Vol. 1).
- Olaniyi, O. O., Okunleye, O. J., Olabanji, S. O., Asonze, C. U., & Ajayi, S. A. (2023). IoT Security in the Era of Ubiquitous Computing: A Multidisciplinary Approach to Addressing Vulnerabilities and Promoting Resilience. *Asian Journal of Research in Computer Science*, 16(4), 354–371. <https://doi.org/10.9734/ajrcos/2023/v16i4397>
- Patel, R. M. (2020). *Cyber Security in Domain of IoT: A Review Threats and Security*. <https://doi.org/10.13140/RG.2.2.20037.47841>
- Purwanto, E. W. (2020). Pembangunan Akses Air Bersih Pasca Krisis Covid-19. *The Indonesian Journal of Development Planning*, IV(2), 2020.
- Rachmawati, D., Tarigan, J. T., & Ginting, A. B. C. (2018). A comparative study of Message Digest 5(MD5) and SHA256 algorithm. *Journal of Physics: Conference Series*, 978(1). <https://doi.org/10.1088/1742-6596/978/1/012116>
- Rahma, F., & Pratama, A. R. (2020). *Keamanan Siber dan Informasi: Prinsip Dasar dan Ancaman Terkini*.
- Reshan, M. S. Al. (2021). IoT-based Application of Information Security Triad. *International Journal of Interactive Mobile Technologies*, 15(24), 61–76. <https://doi.org/10.3991/IJIM.V15I24.27333>
- Simbolon, V. A., & Juwono, V. (2022). Comparative Review of Personal Data Protection Policy in Indonesia and The European Union General Data Protection Regulation. *Publik (Jurnal Ilmu Administrasi)*, 11(2), 178. <https://doi.org/10.31314/pjia.11.2.178-190.2022>
- Soni, D., & Makwana, A. (2017). *A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT)*. <https://www.researchgate.net/publication/316018571>
- Syahputra, B., Sujarwo, A., & Maharika, I. (2019). Bank Air Kami: Terban waterscape information system. *IOP Conference Series: Materials Science and Engineering*, 482(1). <https://doi.org/10.1088/1757-899X/482/1/012047>

- Vojkovic, G., & Milenkovic, M. (2019). *IoT and Smart Home Data Breach Risks from the Perspective of Croatian Data Protection and Information Security Law*.
<https://www.researchgate.net/publication/335828354>
- Wagh, S. K., Pachghare, V. K., & Kolhe, S. R. (2013). Survey on Intrusion Detection System using Machine Learning Techniques. In *International Journal of Computer Applications* (Vol. 78, Issue 16).
- Yuniarti, S. (2022). PROTECTION OF INDONESIA'S PERSONAL DATA AFTER THE RATIFICATION OF THE DRAFT PERSONAL DATA PROTECTION LAW. *Progressive In Law*, 4(2).

LAMPIRAN