RANCANG BANGUN SISTIM KEAMANAN PERANGKAT IoT DENGAN METODE AUTENTIKASI MENGGUNAKAN JSON WEB TOKEN PADA PROTOKOL MQTT

SKRIPSI

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

Celvin Arya Mangkurat

1807422023

PROGRAM STUDI TEKNIK MULTIMEDIA DAN JARINGAN

JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER

POLITEKNIK NEGERI JAKARTA

2022

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertanda tangan dibawah ini:

Nama	: Celvin Arya Mangkurat
NIM	: 1807422023
Jurusan/Program Studi	: Teknik Informatika dan Komputer/Teknik Multimedia dan jaringan
Judul Skripsi	: Rancang Bangun Sistim Keamanan Perangkat IoT dengan Metode Autentikasi Menggunakan JSON Web Token pada Protokol MQTT

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

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

Depok, 27 Juni 2022

Yang membuat pernyataan

Celvin Arya Mangkurat

NIM. 1807422023

HALAMAN PENGESAHAN

Skripsi diajukan oleh :

Nama : Celvin Arya Mangkurat

NIM : 1807422023

Program Studi : Teknik Multimedia dan Jaringan

Judul Skripsi : Rancang Bangun Sistim Keamanan Perangkat IoT dengan Metode Autentikasi Menggunakan JSON Web Token pada Protokol MQTT

Telah diuji oleh tim penguji dalam Sidang Skripsi pada hari Jumat, Tanggal 8, Bulan Juli, Tahun 2022 dan dinyatakan **LULUS**.

Disahkan oleh

Pembimbing : Ayu Rosida Zain, S.ST., M.T. ( )

Penguji I : Nur Fauzi Soelaiman, S.T., M.Kom. ( )

Penguji II : Maria Agustin, S.Kom., M.Kom. ( )

Penguji III : Fachroni Arbi Murad, S.Kom., M.Kom. ( )

Mengetahui :

Ketua Jurusan Teknik Informatika dan Komputer

Mauldy Laya, S.Kom., M.Kom.

NIP 197802112009121003

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan laporan skripsi ini. Penulisan laporan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Terapan di Politeknik Negeri Jakarta. Fokus penelitian ini adalah pembuatan sistem keamanan perangkat IoT dengan metode autentikasi menggunakan JSON Web Token pada protokol MQTT. Penulis menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dan masa perkuliahan sampai pada penyusunan laporan skripsi, sangatlah sulit bagi penulis untuk menyelesaikan skripsi ini. Oleh karena itu, penulis mengucapkan terima kasih kepada semua pihak yang telah membantu terutama kepada:

1. Bapak Mauldy Laya, S.Kom., M.Kom., selaku ketua jurusan Teknik Informatika dan Komputer Politeknik Negeri Jakarta.
2. Bapak Defiana Arnaldy, S.Tp., M.Si., selaku kepala program studi Teknik Multimedia dan Jaringan jurusan Teknik Informatika dan Komputer Politeknik Negeri Jakarta.
3. Ayu Rosida Zain, S.ST, M.T. selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan penulis dalam penyusunan laporan skripsi ini.
4. Orang tua dan keluarga, selaku pihak yang telah memberikan dukungan dan bimbingan moral dan material.

Penulis berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu dan semoga laporan skripsi ini dapat memberikan manfaat bagi pengembangan ilmu.

SURAT PERNYATAAN PERSETUJUAN PUBLIKASI
SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Politeknik Negeri Jakarta, saya yang bertanda tangan di bawah ini:

Nama	: Celvin Arya Mangkurat
NIM	: 1807422023
Jurusan	: Teknik Informatika dan Komputer
Program Studi	: Teknik Multimedia dan Jaringan

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

Rancang Bangun Sistim Keamanan Perangkat IoT dengan Metode Autentikasi Menggunakan JSON Web Token pada Protokol MQTT

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

Depok, 27 Juni 2022

Yang membuat pernyataan

Celvin Arya Mangkurat

NIM. 1807422023

ABSTRAK

IoT merupakan suatu konsep yang bertujuan untuk menghubungkan perangkat-perangkat yang ada disekitar melalui internet sehingga dapat menciptakan sebuah lingkungan yang cerdas dengan melakukan proses pengumpulan dan pertukaran data antar *node*. Dalam mendukung proses pertukaran data pada perangkat IoT agar cepat dan ringan maka dibutuhkan sebuah protokol yang dapat bekerja dengan energi dan media penyimpanan yang kecil. Protokol MQTT adalah protokol pesan ringan berbasis *publish-subscribe* dengan ukuran paket data yang kecil dan juga konsumsi catu daya kecil. Oleh karena itu, protokol MQTT cocok untuk diterapkan pada perangkat IoT. Walaupun demikian, protokol tersebut masih rentan terhadap serangan siber. Berdasarkan survey yang dilakukan oleh shodan, terdapat hampir 67.000 server MQTT yang beredar di internet dengan sebagian besar tidak memiliki autentikasi. Oleh karena itu, penelitian ini dilakukan untuk mengatasi permasalah tersebut dengan menerapkan metode autentikasi menggunakan JSON Web Token dan protokol TLS. Berdasarkan hasil pengujian yang dilakukan dapat diketahui bahwa sistem ini dapat melakukan autentikasi terhadap token valid, kedaluwarsa dan tanda tangan yang *invalid*. Selain itu, dapat diketahui NodeMCU ESP8266 membutuhkan waktu untuk memperoleh token paling sedikit yaitu 0,04 detik, paling banyak yaitu 0,991 detik dan rata-rata waktu yang dibutuhkan adalah 0,43028 detik. Kemudian, dengan menerapkan protokol keamanan TLS, sistem ini dapat menangkal serangan *man in the middle*.

Kata kunci: Autentikasi, JSON Web Token, Keamanan IoT, MQTT, Serangan *man in the middle*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Internet of Things (IoT) merupakan suatu konsep yang bertujuan untuk menghubungkan perangkat-perangkat yang ada disekitar melalui internet sehingga dapat menciptakan sebuah lingkungan yang cerdas dengan melakukan proses pengumpulan dan pertukaran data antar node. Perangkat-perangkat tersebut biasanya terhubung dengan mikrokontroler, sensor dan juga koneksi internet (Kashyap, Sharma & Gupta, 2018). Dalam mendukung proses pertukaran data pada perangkat IoT agar cepat dan ringan maka dibutuhkan sebuah protokol yang dapat bekerja dengan energi dan media penyimpanan yang kecil. Protokol *Message Queuing Telemetry Transport* (MQTT) adalah protokol pesan ringan (*lightweight*) berbasis *publish-subscribe* yang berjalan di atas protokol TCP/IP (Boyd dkk., 2014). Protokol ini mempunyai ukuran paket data *low overhead* kecil dan juga konsumsi catu daya kecil. MQTT bersifat terbuka, simpel dan didesain agar mudah untuk diimplementasikan. MQTT dapat menangani ribuan pengguna secara jarak jauh hanya dengan menggunakan satu *broker*. (Lampkin dkk., 2012). Karakteristik inilah yang membuat protokol MQTT ideal untuk digunakan pada perangkat IoT. Walaupun demikian, protokol tersebut masih rentan terhadap serangan siber (Singh dkk., 2015).

Penyebab utama masalah keamanan pada IoT adalah karena konfigurasi *default* yang tidak aman dan tidak adanya autentikasi. Setiap objek pada IoT harus dapat mengidentifikasi dan mengautentikasi objek lain agar dapat membangun proses pertukaran data yang aman (Mahmoud dkk., 2016). Selain itu, *wildcard* pada MQTT memungkinkan pengguna yang memiliki akses ke server dapat mengakses ke semua pesan yang mengalir melaluinya sehingga dibutuhkan sebuah konfigurasi pada MQTT *broker* yang benar. Berdasarkan survey yang dilakukan oleh shodan, terdapat hampir 67.000 server MQTT yang beredar di internet dengan sebagian besar tidak memiliki autentikasi. Secara konfigurasi *default*, protokol MQTT masih rentan terhadap serangan *man in the middle* yang dapat mencuri informasi penting seperti nama pengguna, kata sandi hingga isi pesan yang di-*publish* dan juga memungkinkan siapa saja untuk berlangganan topik yang disiarkan tanpa adanya

autentikasi sehingga mereka dapat menerima seluruh pesan berdasarkan topik yang telah di-*subscribe* (Lundgren, 2016).

Berdasarkan permasalahan tersebut maka diterapkanlah protokol *Transport Layer Security* (TLS) yang dapat menangkal serangan *man in the middle* dan metode autentikasi yang dapat memvalidasi bahwa data yang dikirim berasal dari pengirim yang sah. JSON Web Token (JWT) merupakan *open standard* (RFC 7519) yang digunakan untuk mengirimkan data secara padat dan aman sebagai objek JSON (Shingala, 2019) (Jones, Bradley & Sakimura, 2015) (Ahmed & Mahmood, 2019). Kemampuan JWT untuk melakukan verifikasi dan tanda tangan secara digital dapat mencegah terjadinya pengiriman data dari pengguna yang tidak terotorisasi (Bhawiyuga, Data & Warda, 2018). JWT digunakan sebagai metode autentikasi sehingga server dapat melakukan verifikasi bahwa data yang dikirim berasal dari pengguna yang sah. Apabila terdeteksi bahwa Token JWT yang dikirim tidak valid maka server akan mengirimkan notifikasi melalui aplikasi Telegram. Telegram dipilih untuk digunakan sebagai media notifikasi pada sistem ini karena Telegram merupakan aplikasi layanan pesan berbasis *cloud* dan bersifat terbuka yang telah menyediakan enkripsi *end-to-end, self destruction messages*, dan infrastruktur *multi data center*. Fitur-fitur tersebut lah yang membuat proses komunikasi pada aplikasi Telegram menjadi aman (Panjaitan & Syafari, 2019) (Fahana, Umar & Ridho, 2017). Oleh karena itu, dianggap penting untuk mengangkat penelitian mengenai keamanan perangkat IoT dengan metode autentikasi menggunakan JSON Web Token pada protokol MQTT.

1.2 Perumusan Masalah

Berdasarkan uraian dari latar belakang tersebut maka perumusan masalahnya adalah sebagai berikut:

a. Bagaimana implementasi metode autentikasi dengan JSON Web Token menggunakan protokol MQTT pada perangkat NodeMCU ESP8266?
b. Bagaimana hasil perancangan sistem keamanan dengan menerapkan metode autentikasi dengan JSON Web Token menggunakan protokol MQTT untuk menangkal serangan *man in the middle*?
c. Bagaimana implementasi sistem notifikasi Telegram pada sistem keamanan IoT?

1.3 Batasan Masalah

Adapun batasan masalah yang dibuat agar pembahasan lebih terukur dan terfokus. Pembatasan masalah tersebut antara lain sebagai berikut:

a. Metode autentikasi menggunakan JSON Web Token
b. Mikrokontroler yang digunakan adalah NodeMCU ESP8266
c. MQTT *broker* yang digunakan adalah mosquitto versi 2.0.14
d. Versi protokol TLS yang digunakan adalah 1.2
e. Sistem dibangun pada jaringan lokal (LAN).
f. Jenis ancaman yang diterapkan pada pengujian adalah serangan *man in the middle*

1.4 Tujuan dan Manfaat

Adapun tujuan dan manfaat dari dilakukannya penelitian ini adalah:

1.4.1 Tujuan

a. Membuat rancang bangun sistim keamanan IoT dengan mengimplementasikan metode autentikasi menggunakan JSON Web Token berbasis protokol MQTT dan notifikasi melalui Telegram.
b. Menguji sistem keamanan IoT dengan autentikasi menggunakan JSON Web Token pada protokol MQTT terhadap serangan *man in the middle*.

1.4.2 Manfaat

a. Dapat meningkatkan keamanan dalam proses pengiriman data pada perangkat IoT.
b. Dapat mengatasi serangan *man in the middle*.

1.5 Sistematika Penulisan

Sistematika penulisan dalam penelitian ini, disusun sebagai berikut:

1. **BAB I PENDAHULUAN**

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

2. **BAB II TINJAUAN PUSTAKA**

   Bab ini berisi pembahasan mengenai materi/teori yang mendukung dan membantu proyek yang dibuat pada skripsi.

3. **BAB III METODE PENELITIAN**

   Pada bab ini akan dijabarkan mengenai pendekatan penelitian yang digunakan, teknik pengumpulan dan analisis data yang digunakan. Selain itu juga dijelaskan bagaimana metode atau langkah-langkah yang dilakukan dalam menyelesaikan pekerjaan atau mengatasi permasalahan didalam skripsi. Pada bagian bab ini menjelaskan objek penelitian yang dijadikan sasaran dalam penelitian ini.

4. **BAB IV PEMBAHASAN**

   Pada bab ini terdapat pembahasan mengenai analisis kebutuhan, perancangan, implementasi sistem dan hasil dari pengujian yang dilakukan terhadap sistem IoT yang telah diimplementasikan metode autentikasi dengan JSON Web Token pada protokol MQTT dan notifikasi melalui aplikasi Telegram apabila terjadi serangan *man in the middle* pada sistem yang dibangun. Pengujian ini dilakukan untuk mengetahui apakah sistem yang dibangun telah berjalan sesuai yang diharapkan atau tidak.

5. **BAB V PENUTUP**

   Pada bab ini berisi pembahasan mengenai hasil akhir keseluruhan berupa kesimpulan dan saran dari penelitian yang dilakukan.

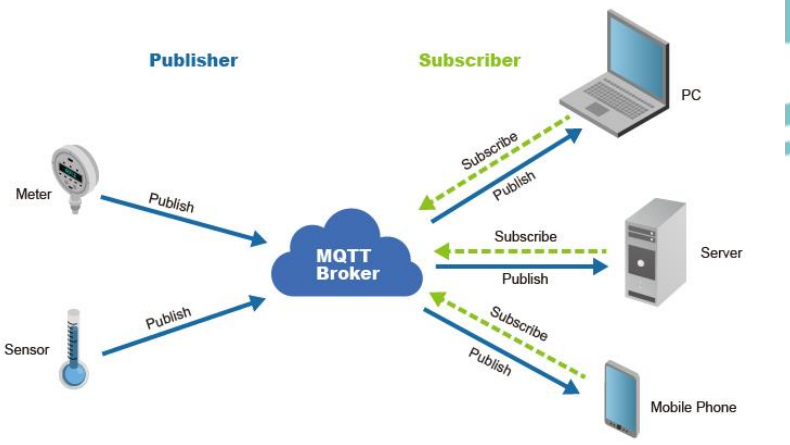
BAB II
TINJAUAN PUSTAKA

2.1 Internet of Things

Internet of Things (IoT) dapat didefinisikan sebagai kemampuan beberapa perangkat yang dapat saling berkomunikasi, terhubung dan bertukar data melalui jaringan internet. IoT merupakan sebuah teknologi komunikasi yang memungkinkan adanya pengendalian, komunikasi, kerjasama dengan berbagai perangkat–perangkat keras, berkomunikasi data melalui jaringan internet. Sehingga bisa disimpulkan bahwa *Internet of Things* (IoT) merupakan sebuah konsep menyambungkan atau menghubungkan sesuatu, yang tidak dioperasikan oleh manusia ke internet. Konsep dasar dari internet of things adalah dengan menggabungkan objek, sensor, kontroler, dan internet yang bisa menyebarkan informasi kepada pengguna. Objek akan dideteksi oleh sensor yang akan diproses oleh kontroler dan dilanjutkan untuk mengirim data yang sudah diolah sehingga menjadi sebuah informasi yang berguna dan secara *real-time* kepada pengguna (Li, Xu dan Zhao, 2015).

2.2 Message Queuing Telemetry Transport

Protokol *Message Queuing Telemetry Transport* (MQTT) adalah protokol yang berjalan pada *layer* aplikasi dan dirancang untuk perangkat dengan sumber daya terbatas. MQTT berbasis pada topik dengan arsitektur *publish-subcribe* yang berarti ketika klien mengirimkan pesan ke topik tertentu, maka klien yang berlangganan ke topik tersebut dapat menerima pesan.



Gambar 2.1 Protokol MQTT

(Sumber: https://oringnet.com/en-global/tech/detail/93)

MQTT menggunakan protokol TCP/IP sebagai standar untuk proses pertukaran data. Protokol ini memiliki ukuran paket data *low overhead* (maksimal 2 gigbyte) dengan konsumsi catu daya kecil. MQTT bersifat *open source*, didesain agar mudah diimplementasikan dan mampu menangani ribuan klien secara jarak jauh dengan hanya menggunakan satu server(Yudha Saputra dkk., 2017). Oleh karena itu, protokol MQTT sangat sesuai jika digunakan untuk komunikasi *Machine to Machine* (M2M) seperti perangkat *Internet of Things* yang memiliki sumber daya terbatas. Arsitektur *publish/subscribe* membutuhkan *broker* yang bertanggung jawab untuk mendistribusikan pesan kepada klien yang telah melakukan *subscribe* terhadap topik pesan. Berikut merupakan fitur dari protokol MQTT (Locke, 2010):

1. Pola pesan *publish/subscribe* yang dapat mendistribusikan pesan dari satu ke banyak dan *decoupling* aplikasi.
2. Menggunakan TCP/IP sebagai konektivitas dasar jaringan
3. Terdapat tiga tingkat *Qualities of Service* (QoS) dalam pengiriman pesan:
   - "*At most once*", dimana pesan dikirim dengan upaya terbaik dari jaringan TCP/IP. Pada tingkat kehilangan pesan dan duplikasi dapat terjadi.
   - "*At least once*", memastikan pesan dikirim sampai ke tujuan walaupun duplikasi juga dapat terjadi.
   - "*Exactly once*", dimana pesan yang dikirim dipastikan tiba ke tujuan tepat satu kali.

2.3 *Transport Layer Security*

Transport Layer Security (TLS) adalah protokol yang digunakan untuk mengamankan komunikasi antara klien dan server melalui jaringan. TLS dirancang untuk dapat mencegah penyadapan, gangguan dan pemalsuan pesan untuk aplikasi klien-server. Protokol TLS atau biasa disebut SSL (*Secure Sockets Layer*) merupakan protokol yang paling banyak digunakan di internet dengan menyediakan otentikasi, integritas dan kerahasiaan untuk dua pihak. Umumnya implementasi protokol ini diterapkan dengan protokol lainnya seperti HTTP dan MQTT. TLS dapat mengamankan informasi rahasia seperti nama pengguna, email dan kata sandi (Turner, 2014). TLS menggunakan *cipher suite* untuk memberikan keamanan komunikasi pada TCP/IP. Sebuah *cipher suite* terdiri dari algoritma kriptografi

yang mendukung proses pertukaran kunci, enkripsi dan pengamanan integritas serta keaslian melalui kode otentikasi pesan berupa alamat MAC. Dengan demikian, TLS dapat mengimplementasikan lapisan *transport* terenkripsi antara dua perangkat yang berkomunikasi secara langsung(Prantl dkk., 2021).

2.4 JSON Web Token

JSON Web Token (JWT) merupakan standar format untuk mengamankan informasi pribadi menjadi sebuah klaim yang akan di *encode* ke dalam bentuk JSON dan menjadi *payload* dari JSON Web Signature (JWS). Klaim tersebut dilindungi oleh tanda tangan digital seperti *Message Authentication Code* (MAC) dan juga dapat dienkripsi. JWT merupakan mekanisme autentikasi yang bersifat *stateless* karena *state* dari pengguna yang melakukan login tidak akan disimpan pada *memory* server. Jadi setiap *request* kepada API yang terproteksi akan diperiksa apakah token JWT yang ada di *Authorization header* valid atau tidak. Jika valid maka *request* akan diijinkan dan diproses. Berikut ini merupakan struktur dari token pada JWT (Jones, Bradley & Sakimura, 2013):

1. *Header*

   Header umumnya terdiri dari dua bagian yaitu jenis token yang digunakan dan algoritma *hashing* yang digunakan seperti HMAC, SHA256 atau RSA.

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Gambar 2.2 JWT *Header*

(Sumber: https://www.jwt.io)

2. *Payload*

   Bagian kedua dari token adalah *payload*. *Payload* dapat berisi klaim yang menyatakan suatu entitas (umumnya pengguna) dan meta data tambahan. Terdapat tiga jenis klaim yaitu *registered*, *public* dan *private.*

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

Gambar 2.3 JWT *Payload*

(Sumber: https://www.jwt.io)

3. *Signature*

   Signature berfungsi untuk memverifikasi bahwa tidak terjadi perubahan pada pesan selama pengiriman berlangsung. Dalam hal ini token yang telah ditandatangani dengan kunci pribadi juga dapat diverifikasi bahwa pengirim JWT dapat dibuktikan keabsahannya. Untuk membuat bagian *signature* dibutuhkan *header* JWT yang di *encode* berbasiskan dengan base64 dan dipisahkan dengan tanda titik lalu *payload* JWT yang di *encode* dengan base64. Selain itu juga dibutuhkan *secret key* yang harus dijaga secara rahasia agar dapat mencegah pengguna yang tidak diinginkan membuat *signature* yang sama.

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

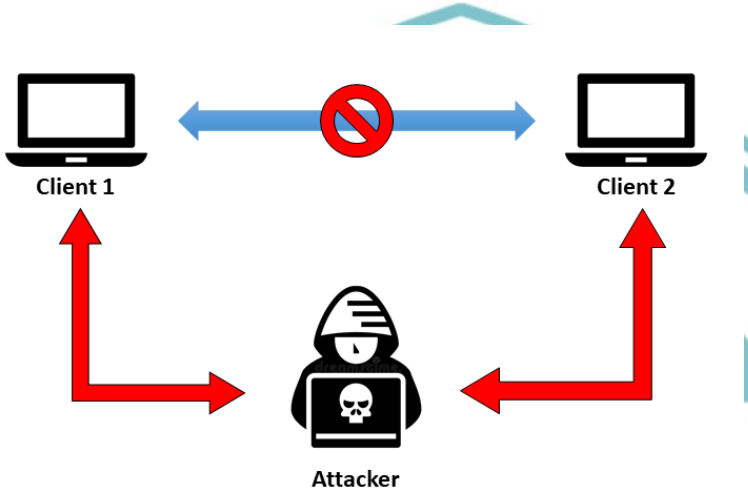
Gambar 2.4 JWT *Signature*

(Sumber: https://www.jwt.io)

2.5 Serangan *Man in The Middle*

Man in the middle merupakan sebuah serangan yang bertujuan untuk membuat protokol autentikasi dimana penyerangan dapat menempatkan dirinya di antara 2 pihak yaitu pemohon autentikasi dan pihak yang bertanggung jawab untuk memverifikasi identitas sehingga penyerang dapat membaca, mencegat dan mengubah data atau pesan yang dikirimkan (Tareq, 2021). Serangan *man in the middle* bekerja dengan mengeksploitasi ARP (*Address Resolution Protocol*). Protokol ARP merupakan sebuah protokol yang bertanggung jawab mencari tahu

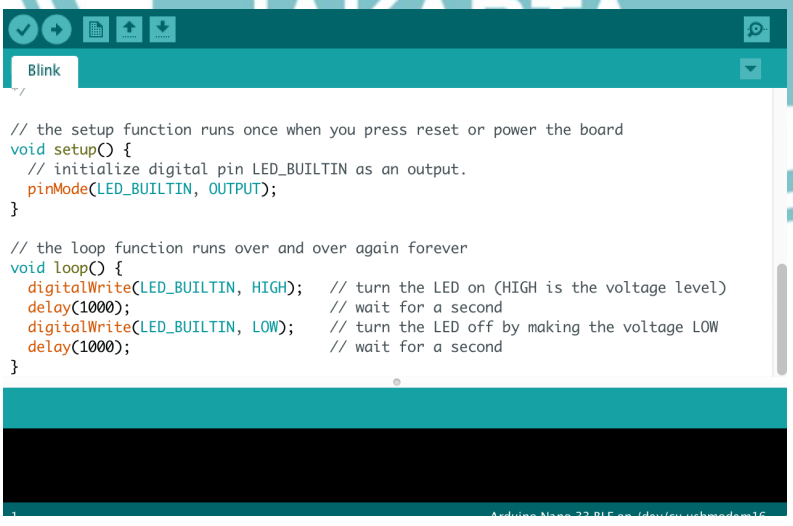
MAC *Address* atau alamat hardware dari suatu host yang tergabung dalam sebuah jaringan dengan memanfaatkan atau berdasarkan IP Address yang terkonfigurasi pada *host* yang bersangkutan (Setiyadi, 2017). Berikut ini merupakan ilustrasi pengiriman data setelah dilakukan serangan *man in the middle* yang dapat dilihat pada gambar



Gambar 2.5 Ilustrasi Serangan *Man in The Middle*

2.6 Arduino IDE

Arduino IDE merupakan singkatan dari Arduino *Integrated Development Environment* yang dapat diartikan sebagai perangkat lunak yang digunakan untuk memprogram perangkat Arduino. Program yang dibuat pada teks editor Arduino IDE disebut *sketch* dan memiliki ekstensi .ino. Aplikasi ini tersedia diberbagai sistem operasi seperti windows, macOS dan juga linux (Endra dkk., 2019).



Gambar 2.6 Tampilan *User Interface* Arduino IDE

(Sumber: https://www.arduino.cc/en/software)

Menurut (Wahyudi & Suhartati, 2016) perangkat lunak Arduino yang digunakan merupakan IDE yang menggunakan bahasa pemrograman Java. Arduino IDE terdiri dari:

a. *Program Editor, window* yang digunakan untuk menulis dan mengedit program.
b. *compiler*, modul yang mengkonversikan kode program menjadi biner.
c. *uploader*, modul yang memindahkan kode biner dari komputer ke dalam memori pada papan Arduino

2.7 Mosquitto

Mosquitto merupakan MQTT *broker* yang memungkinkan pengguna dapat terhubung dengan sensor, perangkat dan aplikasi yang berbasis pada arsitektur *publish/subscribe* dengan fleksibilitas dalam pola komunikasi *request*/*response* dari arsitektur tradisional *client/server*. Mosquitto ditulis dalam bahasa C yang membuatnya sangat cepat dan cocok untuk digunakan oleh semua perangkat mulai dari komputer dengan papan *lower power single* hingga ke server. Mosquitto bersifat *open source* yang artinya dapat diakses oleh siapapun secara gratis. Perangkat lunak ini dapat menyediakan autentikasi dan otorisasi dengan plugin keamanan yang dinamis dengan dukungan pada banyak *platform* seperti Cedalo cloud, Docker, lingkungan *cloud*, Kubernetes, OpenShift, Windows, MacOs dan Linux. Saat ini mosquitto telah hadir dengan versi yang paling terbaru yaitu 2.0.14 (Vannebäck, 2018).

2.8 NodeMCU ESP8266

NodeMCU ESP8266 merupakan sebuah papan elektronik yang berbasis pada *chipset* ESP8266 yang dapat menjalankan fungsi sebagai mikrokontroler dan mampu membuat koneksi ke internet melalui WiFi. Pada perangkat ini terdapat beberapa pin I/O yang dapat dikembangkan menjadi sebuah aplikasi *monitoring* maupung *controlling* pada proyek *Internet of Things* (IoT). NodeMCU ESP8266 dapat diprogram dengan menggunakan *compiler* Arduino melalui aplikasi Arduino IDE dengan menghubungkannya melalui port *micro USB*. Perangkat tersebut juga merupakan modul turunan pengembangan dari modul platform IoT keluarga ESP8266 tipe ESP-12 (Kodali & Mahesh, 2016). NodeMCU ESP8266 memiliki ukuran papan yang kecil yaitu sebesar 57mm x 30mm dengan dilengkapi pin GPIO

sebanyak 13 pin. Perangkat ini mampu menerima tegangan sebesar 3.3 volt hingga 5 volt

Gambar 2.7 NodeMCU ESP8266

(Sumber: https://www.components101.com/)

2.9 Telegram

Aplikasi Telegram pada awalnya adalah sebuah aplikasi layanan pesan yang bertujuan untuk mengirim dan menerima teks atau pesan multimedia antar pengguna pada ponsel cerdas. Telegram didirikan oleh Rusia Pavel Durov pada tahun 2013. Telegram merupakan aplikasi berbasis *cloud* dan bersifat terbuka yang memungkinkan pengguna dapat mengunduh dan menggunakannya secara gratis. Telegram telah menyediakan enkripsi *end-to-end*, *self destruction messages*, dan infrastruktur *multi data center*. Fitur-fitur tersebut lah yang membuat proses komunikasi pada aplikasi Telegram menjadi aman. Telegram memberikan kemudahan akses bagi pengguna karena aplikasi ini dapat berjalan pada banyak *platform* seperti *mobile, desktop* dan lain-lain. Selain itu, dapat membantu administrator untuk membangun sistem notifikasi dengan memanfaatkan fasilitas *open Application Programming Interface* (API) melalui *bot* Telegram agar dapat mengirim dan menerima pesan secara otomatis(De Oliveira, Santos & Neto, 2016)(Panjaitan & Syafari, 2019)(Fahana, Umar & Ridho, 2017).

BAB III
METODE PENELITIAN

3.1 Rancangan Penelitian

Penelitian ini menggunakan pendekatan kuantitatif dengan membuat suatu rancang bangun sistim keamanan untuk perangkat IoT khususnya NodeMCU ESP8266 pada protokol MQTT dengan menerapkan metode autentikasi. Protokol MQTT yang dirancang dengan menggunakan arsitektur *publish/subscribe* yang memungkinkan setiap *publisher* dapat mengirim pesan ke *message broker* yang selanjutnya dapat diterima oleh *subscriber* yang telah berlangganan pada topik yang sama. Oleh karena itu, dibutuhkanlah sebuah mekanisme autentikasi yang dapat memverifikasi pengirim yang sah. Hal ini juga diperlukan untuk dapat mengatasi terjadinya serangan *man in the middle* pada sistem yang dibangun. Penelitian kuantitatif ini lebih menekankan pada bagaimana sistem yang telah dibangun dapat berjalan sesuai dengan yang diinginkan dengan melakukan pengujian fungsional, pengujian performansi & resistansi sistem terhadap serangan *man in the middle*.

3.1.1 Teknik Pengumpulan Data

Proses pengumpulan data terhadap penelitian yang dilakukan harus memiliki cara atau teknik untuk mendapatkan data atau informasi yang baik dan terstruktur serta akurat dari setiap apa yang diteliti, sehingga kebenaran informasi data yang diperoleh dapat dipertanggung jawabkan. Teknik pengumpulan data yang akan digunakan untuk penelitian ini adalah observasi. Metode observasi dalam penelitian ini dilaksanakan untuk memperoleh data dengan mengamati apakah sistem yang telah dibangun dapat berjalan sesuai dengan tujuan berdasarkan dari hasil pengujian. Terdapat beberapa pengujian yang akan dilakukan. Masing-masing pengujian tersebut dirancang sebagai berikut.

- Pengujian fungsional

  Pengujian fungsional dilakukan untuk mengetahui apakah sistem yang telah dibangun telah memenuhi kebutuhan fungsional yaitu sistem dapat melakukan proses autentikasi dengan menggunakan JSON Web Token yang berjalan pada protokol MQTT dan notifikasi melalui aplikasi Telegram.

- Pengujian performansi

  Pengujian performansi merupakan pengujian yang dilakukan untuk mengetahui bagaimana kinerja server dalam menghasilkan token JWT berdasarkan berapa lamanya waktu yang dibutuhkan. Perhitungan waktu dimulai ketika perangkat NodeMCU ESP8266 berhasil melakukan login ke MQTT *broker* dan melakukan *request* kepada server hingga token tersebut diterima oleh NodeMCU ESP8266.

- Pengujian resistansi serangan

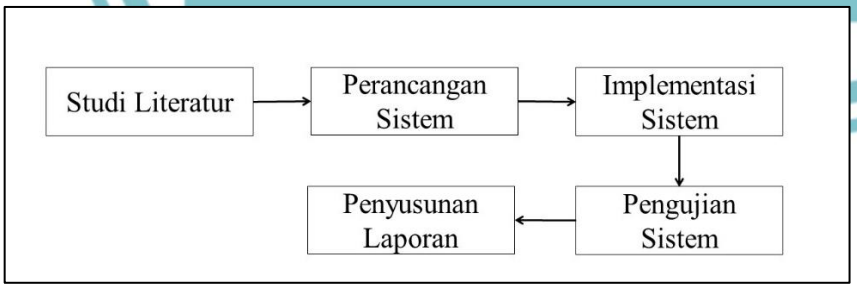
  Pengujian ini dilakukan untuk mengetahui apakah sistem memiliki resistansi terhadap serangan yang ditentukan berdasarkan lingkup penelitian ini. Serangan yang akan dilakukan untuk menguji sistem ini adalah *man in the middle* dengan melakukan interferensi dan memanipulasi pesan yang dikirim oleh NodeMCU ESP8266 kepada server.

3.1.2 Analisis Data

Berdasarkan data yang telah diperoleh dari hasil pengujian fungsional, pengujian performansi dan resistansi terhadap serangan *man in the middle* maka selanjutnya data tersebut akan diolah untuk menghitung tingkat keberhasilan sistem dalam menanggulangi serangan *man in the middle* sehingga dapat mencari jawaban atas pertanyaan atau permasalahan yang dirumuskan sebelumnya. Pengujian tersebut bertujuan untuk memperoleh data yang berguna untuk pengambilan kesimpulan.

3.2 Tahapan Penelitian

Dibawah ini merupakan tahapan penelitian:



Gambar 3.1 Tahapan Penelitian

1. **Studi Literatur**

   Studi literatur merupakan salah satu teknik atau cara yang digunakan untuk mencari ide atau sumber referensi dalam penelitian. Tahapan ini dilakukan untuk

membantu dalam menyelesaikan persoalan dengan menelusuri sumber-sumber tulisan atau literatur berupa jurnal, buku, artikel dan lain-lain yang pernah dibuat sebelumnya.

2. **Perancangan Sistem**

   Tahapan perancangan sistem dilakukan untuk menentukan topologi sesuai untuk diimplementasikan pada sistem dan dapat mengambarkan rangkaian sistem yang akan dibuat sesuai dengan kebutuhan-kebutuhan yang diperlukan. Hal tersebut dilakukan untuk dapat menjelaskan alur sistem yang dibuat dan membantu pada tahapan berikutnya yaitu implementasi sistem.

3. **Implementasi Sistem**

   Implementasi sistem adalah prosedur sistem yang dilakukan untuk menyelesaikan perancangan sistem yang telah disetujui seperti menguji, menginstal, dan memulai menggunakan sistem yang baru atau sistem yang diperbaiki. Implementasi sistem merupakan tahap meletakkan sistem supaya siap untuk dioperasikan.

4. **Pengujian Sistem**

   Setelah sistem berhasil dibangun maka dilakukanlah tahapan pengujian. Pengujian yang akan dilakukan terdiri dari pengujian fungsional, pengujian performansi dan pengujian resistansi terhadap serangan *man in the middle*.

5. **Penyusunan Laporan**

   Pada tahapan ini, dilakukan penyusunan laporan hasil penelitian yang berisi pendahuluan, tinjauan pustaka, metode penelitian, hasil dan pembahasan serta kesimpulan dan saran.

3.3 Objek Penelitian

Dalam penelitian ini, diperlukan data-data atau informasi dari sumber yang dapat memberikan informasi mengenai segala hal yang berhubungan dengan penelitian. Oleh karena itu, diperlukan objek penelitian yang dapat dijadikan sumber informasi yang diperlukan. Sesuai dengan judul penelitian yang telah diuraikan pada Bab I, objek penelitian yang diteliti adalah pengamanan sistem IoT dari serangan *man in the middle*. Penerapan keamanan pada sistem IoT sangat perlu dilakukan agar dapat mengatasi serangan *man in the middle* yang dapat menginterferensi dan memanipulasi data selama proses pengiriman data berlangsung. Dengan

menggunakan metode autentikasi JSON Web Token berbasis protokol MQTT dan notifikasi melalui aplikasi Telegram maka dapat meningkatkan keamanan pada sistem IoT dalam proses pengiriman data dan menanggulangi serangan *man in the middle*.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Analisis Kebutuhan

Analisis kebutuhan merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi ini juga meliputi elemen atau komponen-komponen apa saja yang dibutuhkan untuk sistem yang akan dibangun sampai dengan sistem ini diimplementasikan.

4.1.1 Analisis Perangkat Keras

Perangkat keras yang digunakan pada sistem ini memiliki spesifikasi sebagai berikut.

Tabel 4.1 Spesifikasi Perangkat Keras

Spesifikasi Perangkat Keras		
No	Nama Alat	Spesifikasi
1	NodeMCU ESP8266	Mikrokontroler: ESP8266
		Ukuran papan: 57 mm x 30 mm
		Tegangan input: 3.3 – 5volt
		GPIO: 13 pin
		Kanal PWM: 10 Kanal
		10 bit ADC Pin: 1 Pin
		Flash Memory: 4 MB
		Clock Speed: 40/26/24 MHz
		WiFi: IEEE 802.11 b/g/n
		Frekuensi: 2.4 GHz – 22.5 Ghz
		USB Port: Micro USB
		USB to Serial Converter: CH340G
2	Laptop ASUS A442UR	*Processor*: 8th Gen Intel Core i5-8250U (1.60 – 3.40 GHz, 6 MB SmartCache)
		Memory: 4 GB DDR4 (2133 Mhz)
		Operating System: Windows 11
		Display: 14" HD Color Shine (LED) (1366 x 768)
		Graphic: Intel UHD Graphics 620, Nvidia GeForce GT 930MX 2 GB
		Storage: 1TB SATA (5400 rpm)
		Optical Drive: Super-Multi DVD
		Keyboard: Standart keyboard
		Card Reader: Multi-format card reader (SDXC/SD/SDHC)
		WebCam: VGA Web Camera

		Networking: Built-in Bluetooth V4.0, WiFi Integrated 802.11b/g/n

4.1.2 Analisis Perangkat Lunak

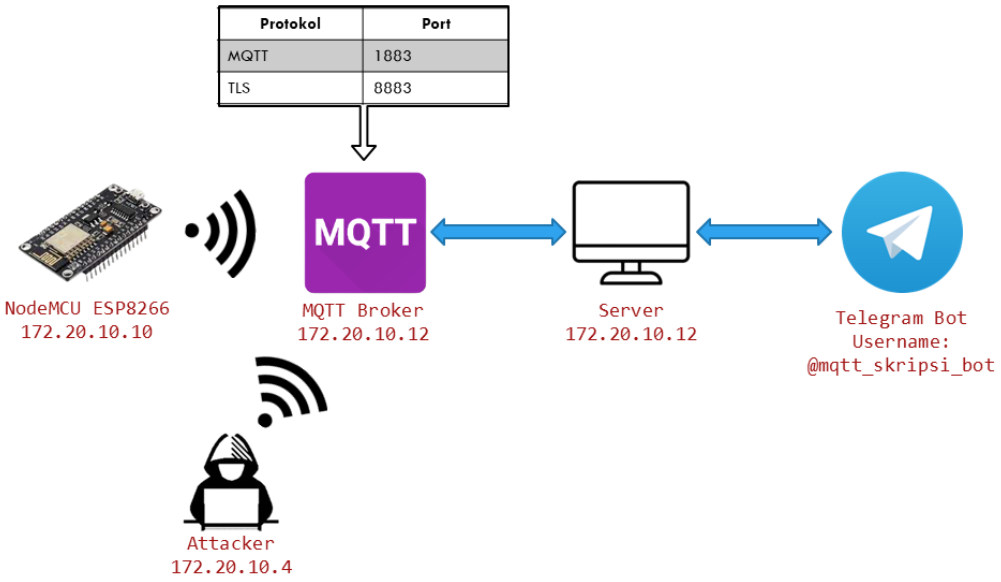
Perangkat lunak yang digunakan pada sistem ini memiliki spesifikasi sebagai berikut.

Tabel 4.2 Spesifikasi Perangkat Lunak

Spesifikasi Perangkat Lunak		
No	Nama Alat	Versi
1	Mosquitto	2.0.14
2	VirtualBox *Graphical User Interface*	6.1.34
3	Kali Linux	2022.2
4	Windows 11	21H2
5	Wireshark	3.6.6
6	Python	3.10.2
7	Arduino	1.8.19

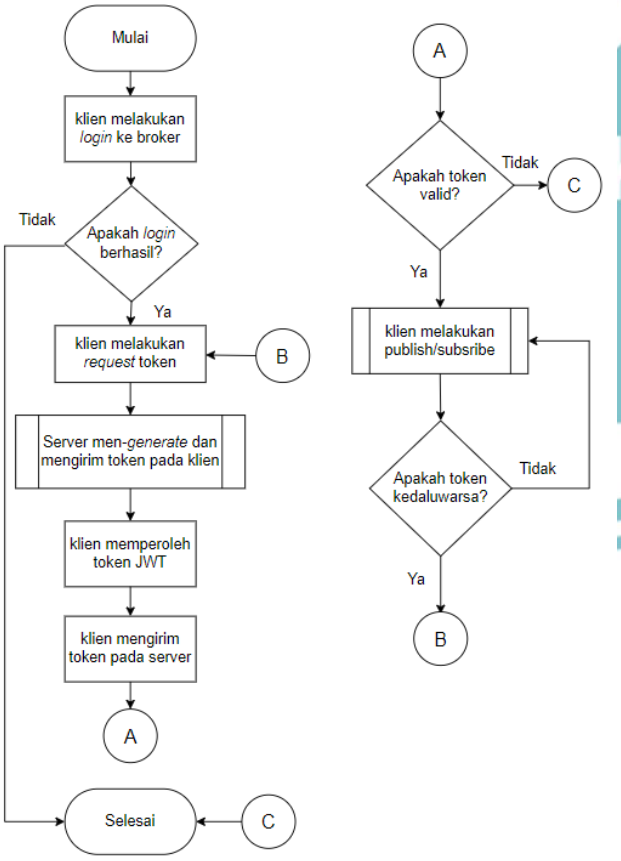
4.2 Perancangan Sistem

Penelitian ini berfokus pada rancang bangun sistim keamanan perangkat IoT dengan menerapkan metode autentikasi. Perangkat IoT yang digunakan adalah NodeMCU ESP8266 dengan menggunakan autentikasi JSON Web Token berbasis protokol MQTT. Secara umum gambaran sistem dapat dilihat pada gambar 4.1.



Gambar 4.1 Topologi Sistem

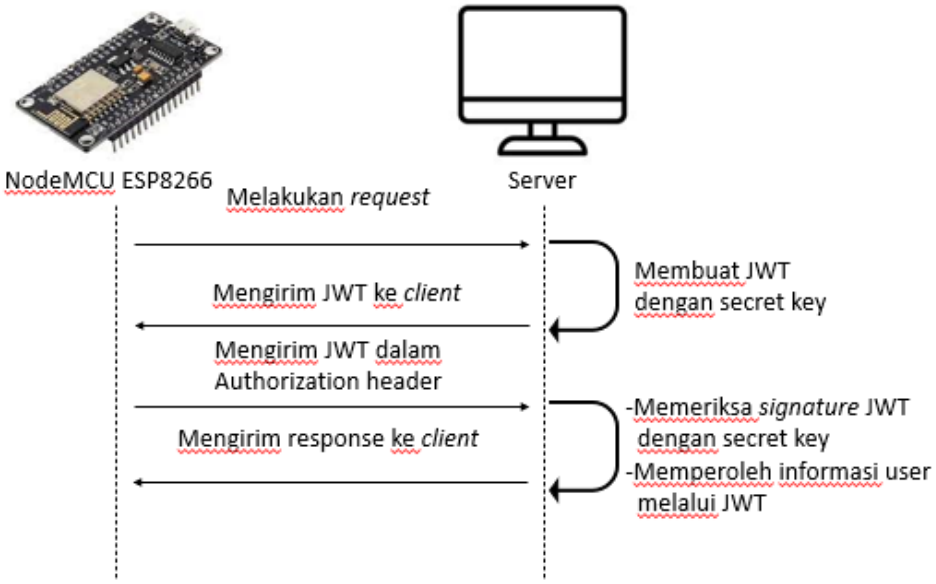
Pada rancangan sistem ini, terdapat empat komponen utama yang terdiri dari NodeMCU ESP8266, MQTT *broker*, server dan aplikasi Telegram sebagai media notifikasi jika terjadi upaya penyerangan berupa *man in the middle* oleh penyerang dengan mendeteksi token JWT yang memiliki tanda tangan *invalid*. NodeMCU ESP8266 dan server saling berkomunikasi menggunakan protokol MQTT. Dalam penerapan protokol MQTT, sebuah sistem harus memiliki tiga entitas utama yang dapat bertindak sebagai *publisher*, *broker* dan *subscriber*. Ketiga entitas tersebut digunakan dalam sistem ini dengan menerapkan beberapa modifikasi. Modifikasi yang diterapkan adalah pada *publisher* dan *subscriber*. Hal ini dilakukan agar NodeMCU ESP8266 dan server dapat saling bertukar informasi secara dua arah. NodeMCU ESP8266 berperan sebagai *publisher* dan *subscriber*. Begitupun dengan server yang juga berperan sebagai *publisher* dan *subscriber*. Alur keseluruhan yang menjelaskan urutan mulai dari proses klien terkoneksi dengan *broker* hingga server autentikasi berhasil mendeteksi adanya pengguna tidak terautentikasi dapat dilihat pada gambar 4.2.



Gambar 4.2 Diagram Alir Sistem

4.2.1 Perancangan Sistem Autentikasi

Sistem autentikasi pada penelitian ini dirancang untuk dilakukan oleh dua pihak yaitu NodeMCU ESP8266 yang akan berperan sebagai klien yang akan menggunakan token JWT dan server yang berperan sebagai pembuat JWT. JWT merupakan token yang merepresentasikan sebuah pesan ringkas yang disebut *claims* dan dilindungi dengan *digital signature*. JWT dapat disimpan pada penyimpanan lokal server sehingga tidak akan membebani perangkat IoT. Oleh karena itu, hal ini lah yang membuat JWT cocok untuk diimplementasikan pada sistem IoT. Token yang dihasilkan JWT terdiri dari *header*, *payload* dan *signature*. *Header* berisi mengenai informasi dari algoritma yang digunakan dan jenis token yang digunakan. *Payload* berisi data-data unik yang digunakan sebagai pengenal dari pemilik token seperti *id*, nama pengguna, *email* ataupun data lain yang berguna dan tidak bersifat sensitif. *Signature* merupakan *hash* gabungan dari *header*, *payload* dan sebuah *secret key* yang bersifat rahasia. *Signature* berguna untuk memverifikasi bahwa tidak terjadi perubahan pada *header* maupun *payload* sehingga apabila terjadi modifikasi terhadap token maka server dapat memastikan bahwa token tersebut tidak valid. Rancangan sistem autentikasi yang dibangun dengan menggunakan JSON Web Token pada penelitian ini dapat dilihat pada gambar 4.3.



Gambar 4.3 Rancangan Sistem Autentikasi Menggunakan JWT

Ketika NodeMCU ESP8266 telah berhasil terhubung dengan server melalui jaringan lokal, NodeMCU ESP8266 sebagai *client* akan melakukan request kepada server untuk dapat memperoleh token JWT. Server akan menerima *request* tersebut dan membuat sebuah token JWT dengan *secret key* yang telah ditentukan yang bersifat rahasia lalu mengirim token tersebut ke NodeMCU ESP8266. Kemudian NodeMCU ESP8266 akan mengirimkan token JWT dalam *authorization header* ke server. Selanjutnya server akan memeriksa *signature* JWT dengan *secret key* untuk memvalidasi apakah token berasal dari pengirim yang sah dan tidak terjadi perubahan selama pengiriman data berlangsung. Jika *signature* JWT yang diterima oleh server tidak valid maka server akan secara otomatis mengirimkan notifikasi melalui aplikasi Telegram. Selain itu juga server akan memperoleh informasi yang dikirim oleh *client* pada token JWT yang berada pada *payload*. Apabila token yang diterima server telah valid, maka server akan mengirim *response* kepada klien.

4.2.2 Perancangan Sistem Komunikasi

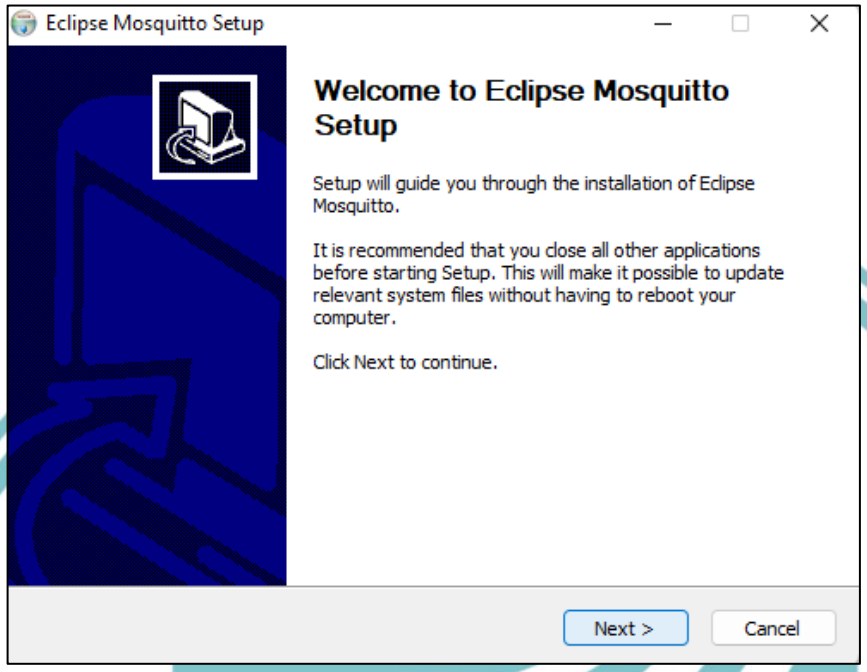
Dalam komunikasi sistem selama proses autentikasi berlangsung, pertukaran data dilakukan dengan protokol MQTT menggunakan skema komunikasi *peer-to-peer*, dimana NodeMCU ESP8266 dan server akan melakukan *publish* dan *subscribe* disaat yang bersamaan. Proses *publish* akan dilakukan didalam fungsi *callback* yang akan merespon setiap pesan yang diterima oleh *subscriber*, sedangkan pada proses *subscribe* terdapat mekanisme tambahan dalam menerima pesan. Mekanisme tambahan tersebut bertujuan untuk memverifikasi pengirim pesan berdasarkan token JWT yang dimiliki. Hal ini diperlukan karena mekanisme *subscribe* yang umum dapat menerima seluruh pesan yang di *publish* pada topik tertentu, sehingga perlu diberikan penanda unik pada setiap pesan yang dikirimkan.

4.3 Implementasi Sistem

a. Instalasi Mosquitto

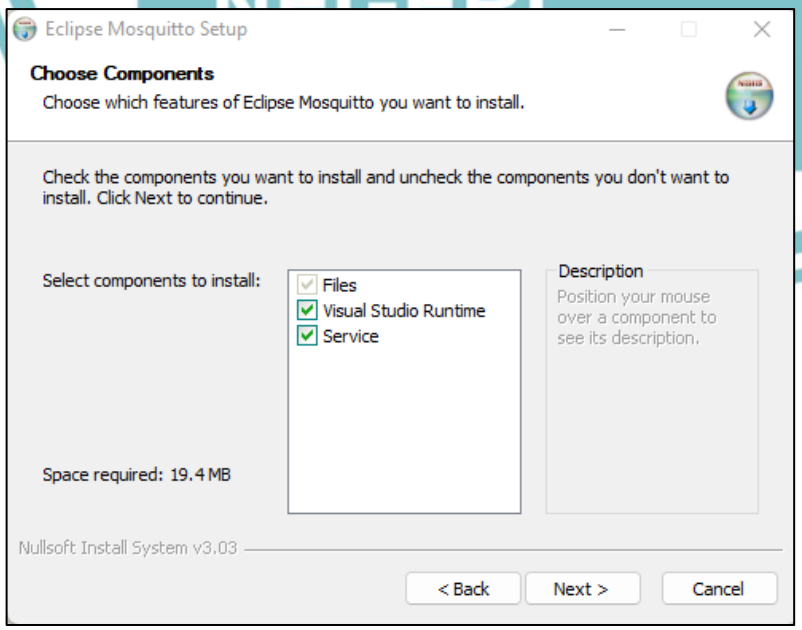
Mosquitto adalah sebuah *message broker* yang bersifat *open source* dan telah mendukung MQTT versi 5.0, 3.1.1 dan 3.1. Mosquitto juga telah mendukung *bridge*, yaitu mekanisme yang dapat menghubungkan para pengguna sehingga dapat saling bertukar pesan. Mosquitto versi 2.0.14 diinstal pada server untuk berperan sebagai MQTT *broker* sehingga NodeMCU ESP8266 dan server yang juga berperan sebagai *publisher* dan *subscriber* dapat saling bertukar informasi

melalui MQTT *broker* yang telah diinstal. Versi ini merupakan versi luncuran mosquitto yang terbaru. Tahapan pertama untuk melakukan instalasi aplikasi mosquitto terlihat pada Gambar 4.4.



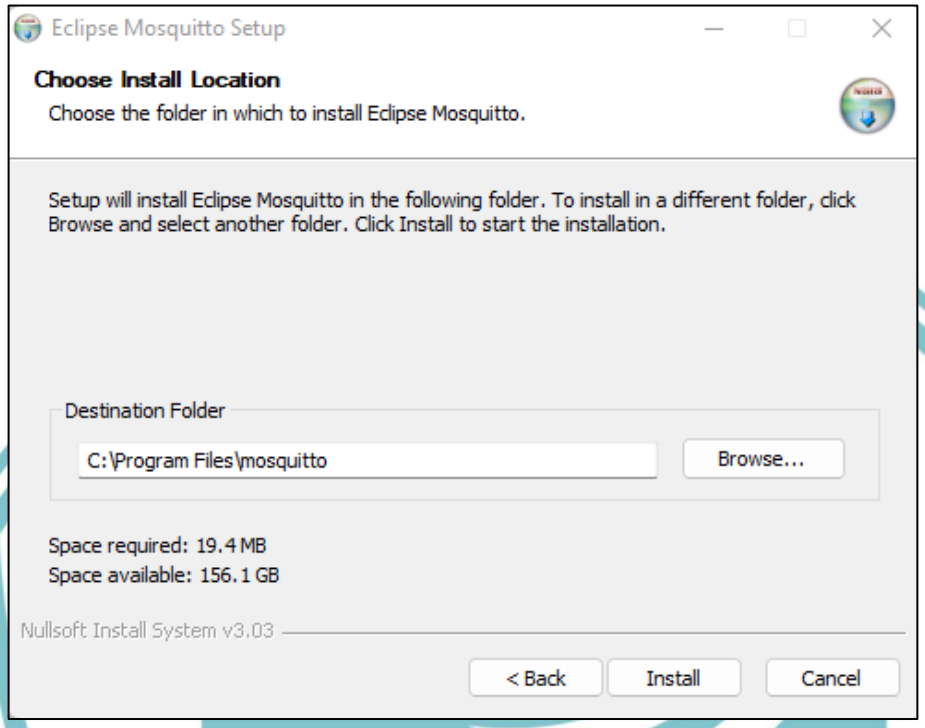
Gambar 4.4 Instalasi Mosquitto Tahap 1

Setelah itu, pada tahapan instalasi mosquitto yang dapat dilihat pada gambar 4.5 adalah menentukan komponen yang akan diinstal pada mosquitto. Pada sistem ini komponen yang akan diinstal adalah *Files*, *Visual Studio Runtime* dan *Service.*



Gambar 4.5 Komponen yang akan Diinstal

Jika tahapan instalasi komponen telah dilakukan, selanjutnya adalah menentukan lokasi folder dimana aplikasi mosquitto akan diinstal. Seperti yang dapat dilihat pada gambar 4.6



Gambar 4.6 Menentukan Folder Destinasi untuk Mosquitto

b. Konfigurasi MQTT *Broker*

Broker pada MQTT berfungsi untuk mengatur alur pengiriman data yang dikirim oleh publisher sehingga dapat diterima oleh *subscriber* yang telah berlangganan dengan topik yang sama dari berbagai perangkat berbeda. Sebagai penyedia layanan MQTT *broker*, mosquitto menyediakan *file* bernama "mosquitto.conf" agar administrator dapat menerapkan konfigurasi MQTT *broker* sesuai dengan kebutuhan. Konfigurasi MQTT *broker* yang diterapkan pada sistem ini tersimpan pada *file* yang bernama "Prog.conf". Konfigurasi tersebut akan membuat MQTT *broker* yang ada pada aplikasi mosquitto berada dalam mode *listener* sehingga MQTT *broker* dapat menerima pesan yang masuk dari *publisher* pada topik tertentu melalui port 1883. Selain itu, pada MQTT *broker* juga diterapkan konfigurasi TLS yang dapat diakses melalui port 8883 untuk membantu mengamankan komunikasi antara *broker* dan mqtt klien. Selain itu, pada MQTT *broker* juga ditambahkan *plugin* yang bernama '*dynamic security*'.

Plugin tersebut berfungsi untuk mengelola setiap akun yang akan terhubung dengan *broker*. Konfigurasi *file* tersebut dapat dilihat pada gambar 4.7.

Prog.conf

```
1  listener 1883 192.168.100.193
2  listener 8883 192.168.100.193
3
4  #TLS
5  cafile D:\mosquitto\cert\ca.crt
6  keyfile D:\mosquitto\cert\server.key
7  certfile D:\mosquitto\cert\server.crt
8
9  tls_version tlsv1.2
10
11 #Konfigurasi dynamic security plugin
12 per_listener_settings false
13 plugin D:\mosquitto\mosquitto_dynamic_security.dll
14 plugin_opt_config_file D:\mosquitto\dynamic-security.json
15
16 log_dest stdout
17 log_type error
18 log_type warning
19 log_type notice
20 log_type information
21
22
23 connection_messages true
24 log_timestamp true
25 log_timestamp_format %Y-%m-%dT%H:%M:%S
26
27 log_dest file D:\mosquitto\log.log
```

Gambar 4.7 Isi dari Berkas Konfigurasi MQTT Broker

c. Konfigurasi NodeMCU ESP8266

Dalam mendukung proses pertukaran data pada protokol MQTT agar dapat berjalan dengan baik, perangkat NodeMCU ESP8266 diprogram untuk dapat terhubung pada area jaringan lokal yang sama dengan *broker* melalui media nirkabel menggunakan koneksi WiFi. Setelah terhubung, NodeMCU ESP8266 akan melakukan koneksi terhadap *broker* menggunakan akun yang telah dibuat sebelumnya. Jika berhasil terhubung, NodeMCU ESP8266 akan menjalankan fungsi *callback*. Fungsi tersebut akan membuat NodeMCU ESP8266 dapat melakukan *publish* dan *subscribe* sehingga perangkat tersebut dapat mengirim dan menerima pesan berdasarkan topik yang telah ditentukan dan dalam satu jaringan yang sama. Selain itu, untuk membangun komunikasi yang aman ke MQTT *broker*, NodeMCU ESP8266 akan mengakses port 8883 pada *broker* sehingga dapat menerapkan protokol TLS dengan menggunakan sertifikat yang

telah dibuat oleh *broker*. Program dibuat menggunakan Arduino IDE versi 1.8.19 dan disimpan dengan nama *file* 'ProgramTLS' yang menggunakan *library* untuk protokol MQTT seperti yang dapat dilihat pada gambar 4.8.

ProgramTLS

```
#include <ArduinoJson.h>
#include <ArduinoJWT.h>
#include <sha256.h>
#include <PubSubClient.h>
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>

#define CHECK_CA_ROOT

//wifi
const char* ssid = "UNKNOWN_DEVICE";
const char* password_wifi = "mamaratu";

//mqtt
const char* mqtt_broker = "192.168.100.193";
int mqtt_port = 8883;
String clientID = "NodeMCUESP8266";
String mqtt_username = "nodemcu";
String mqtt_password = "123456";

const char* topic2_lvl1 = "topic_2";//subscribe_by_esp_lvl1
const char* topic1_lvl1 = "topic_1";//publish_by_esp_lvl1

const char* topic1_lvl2 = "topic_1/authUser";//publish_by_esp_lvl2

//jwt
ArduinoJWT jwt = ArduinoJWT("secret");
String myToken = "";
String token;

//otoritasi
int msgCount = -1;
bool approved = false;

//TLS
#ifdef CHECK_CA_ROOT
static const char digicert[] PROGMEM = R"EOF(
```

Gambar 4.8 Isi dari *File* Skrip NodeMCU ESP8266

d. Konfigurasi Server

Konfigurasi server dilakukan agar server dapat melakukan pertukaran data dengan NodeMCU ESP8266 melalui MQTT *broker*. Server harus dapat mengirim dan menerima pesan sesuai dengan topik yang telah di-*subscribe*. Komunikasi antara server dengan MQTT *broker* dilakukan melalui protokol keamanan TLS untuk melindungi dari teknik *sniffing* yang dilakukan peretas untuk mendapatkan informasi rahasia. Selain itu, konfigurasi ini juga dilakukan

agar server dapat terhubung dengan aplikasi Telegram melalui API Telegram sehingga server dapat mengirimkan pesan berupa peringatan mengenai adanya upaya serangan *man in the middle* terhadap sistem ini. Program dibuat dengan menggunakan bahasa pemrograman python agar dapat memenuhi kebutuhan tersebut.

Pada gambar 4.9 merupakan isi dari *file* skrip yang bernama 'ConnectBroker.py' skrip tersebut berfungsi agar server dapat terhubung dengan *broker* menggunakan akun yang telah dibuat khusus untuk server dengan mengakses port 8883 yang berjalan pada protokol keamanan TLS.

ConnectBroker.py | ServerTLS.py | Log.py

E: > POLITEKNIK NEGERI JAKARTA > Semester8PNJ > Produk Skripsi_Server > PROGRAM SK

```python
import ssl
from paho.mqtt import client as mqtt_client

broker = "192.168.100.193" #ip broker
# port = 1883
port = 8883

client_id = "server"
username = "server"
password = "123456"

dirtls="D:/mosquitto/cert/ca.crt"

def connect_mqtt() -> mqtt_client:
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print("--Connected to MQTT Broker--")
        else:
            print("Failed to connect, return code %d\n", rc)


    client = mqtt_client.Client(client_id)

    client.tls_set(dirtls, tls_version=ssl.PROTOCOL_TLSv1_2)
    client.tls_insecure_set(True)

    client.on_connect = on_connect
    client.username_pw_set(username, password)
    client.connect(broker, port)

    return client
```

Gambar 4.9 Isi dari *File* Skrip Koneksi antara Server ke Broker

Pada gambar 4.10 merupakan *file* skrip yang bernama 'ServerTLS.py'. Skrip tersebut berfungsi agar server dapat menjalankan fungsi *callback* dan dapat menangani setiap pesan yang masuk pada aplikasi telegram. Fungsi *callback* dapat membuat server menerima setiap pesan yang di-*subscribe* pada topik pada level satu yaitu 'topic_1' maupun topik level dua yaitu 'topic_1/authUser' yang digunakan oleh NodeMCU ESP8266 sebagai *publisher* setelah token JWT yang dikirim telah berhasil diautentikasi oleh server.

ConnectBroker.py | ServerTLS.py | Log.py

E: > POLITEKNIK NEGERI JAKARTA > Semester8PNJ > Produk Skripsi_Server > PROGRAM SKRIPSI > ServerTLS.py > ...

```
377  def run():
378      try:
379          t = threading.Thread(name='runtelegram', target=runTelegram)
380          t.daemon = True
381          t.start()
382      except Exception as e:
383          print("Error ni thread"+e)
384      sendMessageTele("Server is online!")
385      c = threading.Thread(name='ceksignaturebackground', target=cekSignature)
386      c.daemon = True
387      c.start()
388      setAccessRole("1","nodemcu")
389      subscribe(client)
390      client.loop_forever()
391
392  if __name__ == '__main__':
393      run()
```

Gambar 4.10 Isi dari *File* Skrip Server

Agar dapat mendeteksi seluruh aktivitas yang dilakukan oleh setiap akun yang terkoneksi dengan MQTT *broker*. File Skrip 'Log.py' dibuat untuk memenuhi kebutuhan tersebut. Seperti yang dapat dilihat pada gambar 4.11.

ConnectBroker.py | ServerTLS.py | Log.py

E: > POLITEKNIK NEGERI JAKARTA > Semester8PNJ > Produk Skripsi_Server > PROGRAM SKRIPSI > Log.py > ...

```
169  if __name__ == '__main__':
170      writeClientOnline()
171      logFile("candidate.txt")
172      logfile = open("D:\mosquitto\log.log","r")
173      loglines = follow(logfile)
174
175      for line in loglines:
176          print(line, end='')
177          if (line.find("New client connected from") != -1):
178              lineSplit = line.split(" ")
179              #['2022-06-11T21:03:01:', 'New', 'client', 'connected', 'from', '192.168.100.193:55851'
180              ipSplit = lineSplit[5].split(":")
181              # Baca Blacklist
182              users=readBlacklist()
183              if (users != False):
184                  if (users != ""):
185                      for user in users:
186                          if (user.rfind(ipSplit[0]) != -1):
187                              userSplit = user.split(":")
188                              usrnm=userSplit[0]
189                              ipAdd=userSplit[1]
190                              # lineSplit[11] : u'nodemcu').
191                              subUser = lineSplit[11].split("'")
```

Gambar 4.11 Isi dari *File* Skrip Log.py pada Server

e. Konfigurasi JSON Web Token

JSON Web Token merupakan sebuah token berbentuk JSON yang dapat digunakan untuk proses autentikasi. JSON Web Token akan diimplementasikan pada server. Ketika NodeMCU ESP8266 berhasil terhubung ke server dan dapat mengirim *request*, server akan membuat sebuah token yang akan digunakan oleh NodeMCU ESP8266 sebagai tiket untuk melakukan transaksi data yang sah dengan server. Token tersebut berisi *payload* yang mengandung informasi mengenai pesan apa yang dikirim dan juga beberapa konten lain. Untuk memverifikasi apakah pengirim token merupakan pengirim yang sah dan tidak ada perubahan pada pesan selama proses pengiriman data terjadi, terdapat *signature* yang berisi rangkaian dari *header*, *payload* dan *secret key* yang di-*encode* menggunakan base64. Seluruh konfigurasi untuk menangani token JWT yang diterima oleh server sebagai *subscriber* pada topik 'topic_1' dibuat didalam *function* 'subscribe' yang dapat dilihat pada gambar 4.12. *Function* akan membuat server dapat mendeteksi apakah token tersebut valid, kedaluwarsa atau *invalid*.

ectBroker.py | ServerTLS.py | Log.py

ITEKNIK NEGERI JAKARTA > Semester8PNJ > Produk Skripsi_Server > PROGRAM SKRIPSI > ServerTLS.py > ...

```
def subscribe(client: mqtt_client):
    def on_message(client, userdata, msg):
        global TokenJWTNodeMCU
        tokenJWT=msg.payload.decode()
        print(f"Received `{tokenJWT}` from `{msg.topic}` topic")
        try:
            plainMsg = jwt.decode(tokenJWT,secretKey, algorithms=["HS256"])
            print(f"\n---Decoded---")
            print("id :",plainMsg["id"],"\niss :",plainMsg["iss"],"\nsts :",plainMsg["sts"],"\nexp :",plainMsg["exp"])
            if (plainMsg["sts"] == "1") & (plainMsg["id"] == "nodemcu"):
                #ubah menjadi AccessRoleLevel2
                setAccessRole("2", plainMsg["id"])
                #Simpan token
                TokenJWTNodeMCU = tokenJWT
        except InvalidSignatureError as error:
            print(f"\nUnable to decode the token, error : {error}")
            sleep(1)
            try:
                file = open(f"D:\mosquitto\candidate.txt", "r")
                if (file != ""):
                    for line in file:
                        try:
                            sub = line.split(":")
                            id=sub[0]
                            username=sub[1]
                            ipAddress=sub[2]
                            if username != "admin":
                                msg=f"\nThis token was sent by:\nID: {id}\nUsername: {username}\nIP Address: {ipAddress}"
                                print(f"{msg}")
                                # Mengirim informasi ke telegram
                                msg = f"JSON Web Token with invalid signature has been detected!{msg}\nYou can block this IP
                                sendMessageToTele(msg)
                        except:
                            print("")
                file.close()
            except FileNotFoundError:
                print("File candidate.txt not found")
```

Gambar 4.12 *Function* untuk Menangani Token JWT yang Diterima

f. Pengujian Sistem

Pengujian sistem dilakukan bertujuan untuk mengetahui apakah sistem yang telah dibangun telah sesuai dengan dengan kebutuhan. Pada tahap ini terdapat tiga pengujian yang dilakukan yaitu pengujian fungsional yang bertujuan untuk mengetahui apakah implementasi metode autentikasi dengan JSON Web Token telah berhasil, pengujian performansi untuk mengetahui berapa lama waktu yang dibutuhkan klien agar dapat memperoleh token JWT dan pengujian resistansi terhadap serangan *man in the middle*.

4.4 Pengujian

Pengujian dilakukan setelah sistem keamanan perangkat IoT yang telah diterapkan metode autentikasi menggunakan JSON Web Token berbasis protokol MQTT selesai dibangun. Pengujian dilakukan untuk mengetahui apakah sistem tersebut dapat berfungsi sesuai dengan tujuan serta dapat menangkal serangan *man in the middle*.

4.4.1 Deskripsi Pengujian

- Pengujian fungsional

  Pengujian fungsional bertujuan untuk mengetahui apakah sistem yang telah dibangun telah memenuhi kebutuhan fungsional yaitu sistem dapat melakukan proses autentikasi pada token JWT yang diterima oleh server dan berjalan pada protokol MQTT. Pengujian fungsional yang akan dilakukan meliputi sebagai berikut:

  1) Autentikasi dengan token yang valid
  2) Autentikasi dengan token yang telah kedaluwarsa
  3) Autentikasi dengan token yang memiliki tanda tangan *invalid*

- Pengujian performansi

  Pengujian performansi merupakan pengujian yang dilakukan untuk mengetahui bagaimana kinerja server dalam menghasilkan token JWT berdasarkan berapa lamanya waktu yang dibutuhkan. Perhitungan waktu dimulai ketika perangkat NodeMCU ESP8266 berhasil melakukan *login* ke MQTT *broker* dan melakukan permintaan token kepada server hingga token tersebut diterima oleh NodeMCU ESP8266.

- Pengujian resistansi

  Pengujian ini dilakukan untuk mengetahui apakah sistem memiliki resistansi atau ketahanan terhadap serangan yang ditentukan berdasarkan lingkup penelitian ini. Serangan yang akan dilakukan untuk menguji sistem ini adalah *man in the middle* dengan melakukan teknik *sniffing* menggunakan aplikasi wireshark. Peretas akan melakukan *sniffing* dengan menangkap setiap data yang masuk maupun keluar dari MQTT *broker* sehingga dapat memperoleh informasi penting seperti nama pengguna, kata sandi maupun isi pesan yang di-*publish*. Pada tahap pengujian ini akan dapat dilihat perbedaan antara sebelum menerapkan TLS dan setelah menerapkan TLS pada sistem terhadap keamanan pertukaran data yang dilakukan antara *broker* dengan klien MQTT.

4.4.2 Prosedur Pengujian

Berikut ini merupakan prosedur pengujian yang akan dilakukan pada pengujian fungsional, pengujian performansi dan pengujian resistansi.

1. Pengujian Fungsional

Pada pengujian ini perangkat NodeMCU ESP8266, MQTT *broker* dan juga server akan berjalan pada satu area jaringan lokal yang sama. Pengujian ini dilakukan bertujuan untuk mengetahui apakah server dapat melakukan autentikasi dari setiap token yang dikirim oleh *publisher*. Token yang digunakan bertipe JSON Web Token dengan algoritma HS256 dan memiliki struktur *payload* yang terdiri dari id, iss, sts dan exp. Server harus dapat mengautentikasi token dan mengklasifikasikan apakah token tersebut berstatus valid, kedaluwarsa atau *invalid*. Jika token yang diterima oleh server adalah valid. Maka server akan melakukan *decode* terhadap token tersebut untuk mendapatkan nilai dari id yang terdapat pada *payload* token. Nilai dari id tersebut berisi nama pengguna dari pengirim token. Kemudian server akan mengubah aturan akses pemilik token sehingga dapat mengakses topik yang lebih tinggi yaitu topik_1/authUser. Akan tetapi, apabila server mendeteksi bahwa token tersebut memiliki tanda tangan yang tidak valid maka server akan mengirimkan notifikasi melalui aplikasi telegram bahwa telah terjadi pengiriman token dengan tanda tangan *invalid* diikuti dengan keterangan pengirim token berupa ID, nama pengguna dan juga alamat IP. Token yang dibuat oleh server hanya

berlaku selama dua puluh detik. Hal ini dilakukan untuk alasan keamanan. Seluruh kebutuhan fungsional yang harus dipenuhi oleh sistem ini tertera pada tabel 4.3.

Tabel 4.3 Kebutuhan Fungsional

No	Kebutuhan Fungsional
1	Server dapat melakukan autentikasi dengan token yang valid
2	Server dapat mendeteksi token yang telah kedaluwarsa atau *expired token*
3	Server dapat mendeteksi token yang memiliki tanda tangan *invalid*
4	Server dapat mengirimkan notifikasi melalui aplikasi telegram jika terdeteksi adanya token yang memiliki tanda tangan *invalid*
5	MQTT *broker* dapat melakukan blokir terhadap akses dari alamat IP MQTT klien sesuai dengan *input* yang diberikan oleh administrator melalui aplikasi telegram

2. Pengujian Performansi

Pengujian performansi dilakukan untuk mengukur performansi dari sistem ini berdasarkan dari waktu yang dibutuhkan oleh *publisher/subscriber* untuk dapat memperoleh token JWT dari server autentikasi. NodeMCU ESP8266 sebagai klien atau penerima token akan melakukan kalkukasi waktu mulai dari melakukan permintaan token pada server autentikasi hingga NodeMCU ESP8266 menerima token tersebut. Pengujian akan dilakukan oleh NodeMCU ESP8266 dengan melakukan permintaan token pada server dan menerima token tersebut sebanyak lima puluh kali menggunakan protokol MQTT yang telah diterapkan protokol keamanan TLS.

3. Pengujian Resistansi

Pengujian ini dilakukan untuk mengetahui apakah sistem yang telah dibangun dapat menangkal serangan *man in the middle*. Serangan ini akan dilakukan oleh peretas melalui sistem operasi kali linux yang berjalan pada mesin virtual. Peretas akan melakukan teknik *sniffing* menggunakan aplikasi wireshark dan membuat koneksi dengan MQTT *broker* sehingga peretas dapat melakukan *capturing* paket berdasarkan protokol MQTT dan TLS yang melintas masuk maupun keluar dari

broker. Pada pengujian ini akan dapat diketahui apakah penerapan protokol keamanan TLS pada sistem dapat menangkal serangan yang dilakukan oleh peretas.

4.4.3 Data Hasil Pengujian

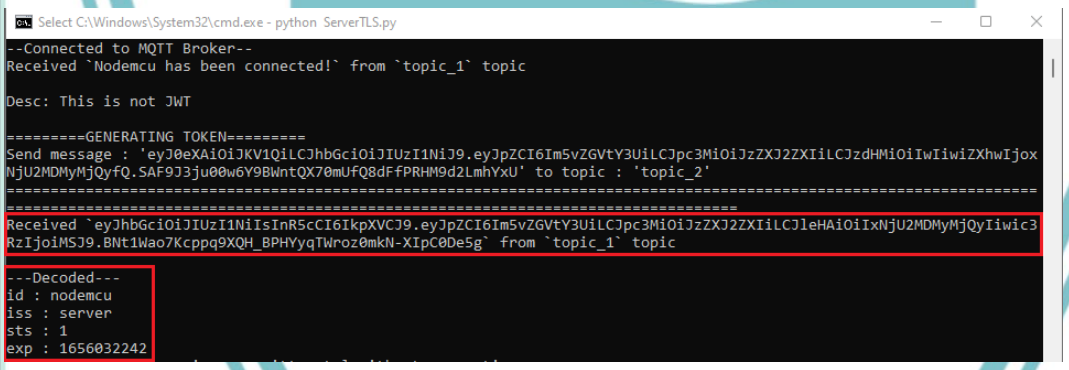
Berikut ini merupakan data hasil dari ketiga pengujian yang telah dilakukan.

1) Pengujian Fungsional

Pada pengujian fungsional terdapat tiga jenis token yang akan dilakukan autentikasi yaitu token yang berstatus valid, token yang telah kedaluwarsa dan token yang memiliki tanda tangan *invalid*.

a) Autentikasi dengan token yang valid

Pada gambar 4.13 merupakan hasil dari respon server autentikasi terhadap token yang dikirim oleh klien. Pada gambar tersebut dapat dilihat bahwa server dapat melakukan autentikasi dan *decode* terhadap token yang diterima. Hasil *decode* tersebut terdiri dari id yang berisi nama pemilik token yaitu ‘nodemcu’, iss yang berisi penerbit atau pembuat token yaitu ‘server’, sts yang berisi status dari token tersebut yaitu satu yang menunjukkan bahwa token yang dikirim oleh server autentikasi telah diterima dan dikonfirmasi oleh penerima token, lalu pada bagian exp menunjukan masa berlaku token tersebut.

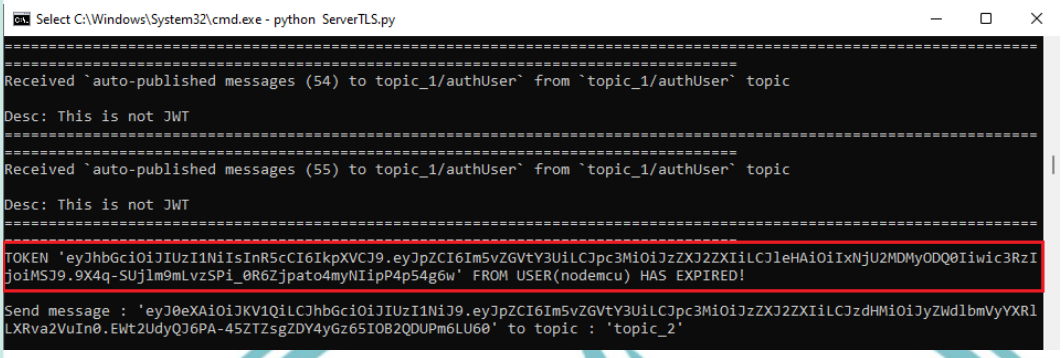


Gambar 4.13 Respon Server Terhadap Token Valid

b) Autentikasi dengan token yang telah kedaluwarsa

Setiap token yang dibuat oleh server autentikasi hanya berlaku selama dua puluh detik. Setelah dua puluh detik, token akan kedaluwarsa dan klien harus meminta token baru kepada server. Server harus dapat melakukan autentikasi pada token yang telah kedaluwarsa. Seperti yang dapat dilihat pada gambar 4.14, server berhasil melakukan autentikasi terhadap token yang telah kedaluwarsa dengan

menampilkan respon bahwa token yang diterima telah kedaluwarsa sehingga klien dapat melakukan *request* kembali dan token dapat membuat token yang baru.

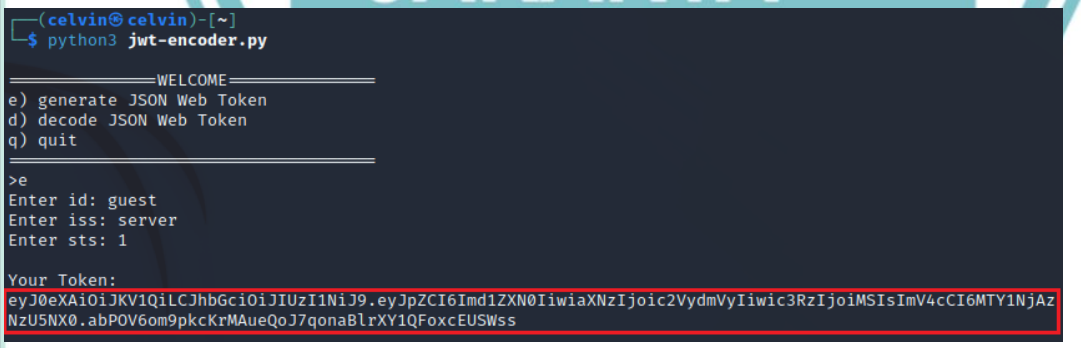


Gambar 4.14 Respon Server Terhadap Token Kedaluwarsa

c) Autentikasi dengan token yang memiliki tanda tangan *invalid*

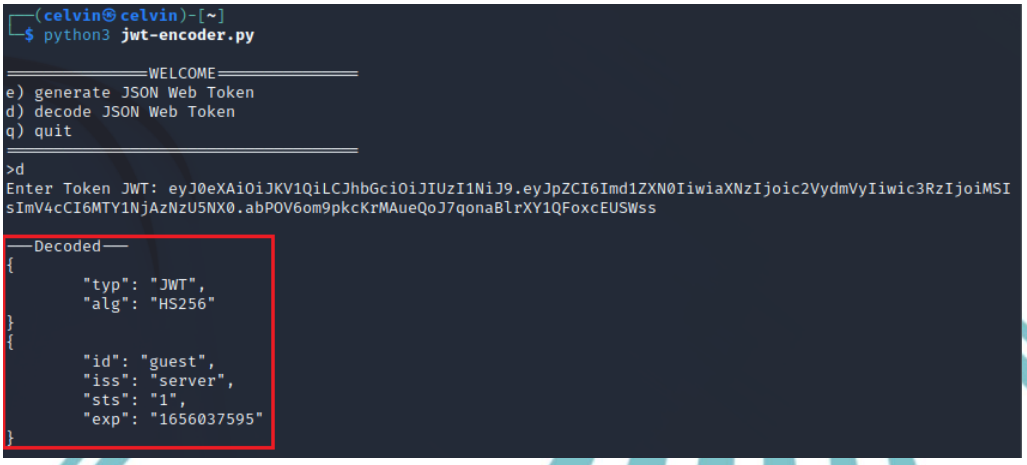
Untuk mencegah peretas dapat mengirimkan token JWT palsu, server harus dapat melakukan autentikasi terhadap setiap token yang diterima. Token JWT terdiri dari *header*, *payload* dan *signature*. *Signature* atau tanda tangan digital dibuat dari *hash* yang merupakan gabungan antara *header*, *payload* dan sebuah kunci rahasia. Apabila token JWT dibuat menggunakan kunci rahasia yang berbeda dengan server atau selama pengiriman token terjadi perubahan data maka struktur dari *signature* JWT akan berubah sehingga token akan *invalid.*

Pada gambar 4.15 menunjukkan peretas membuat token JWT dengan struktur payload yang sama yang terdiri dari id, iss, sts dan exp. Nilai exp telah diatur dalam program yang digunakan peretas bahwa token tersebut dapat berlaku selama satu jam.



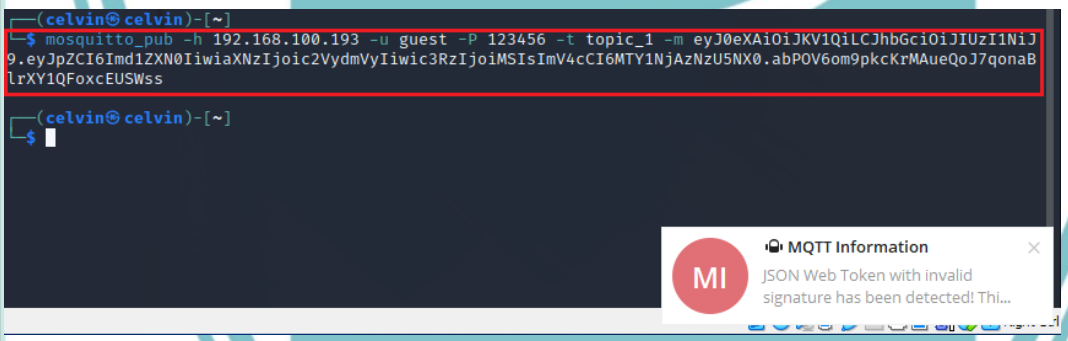
Gambar 4.15 Hasil *Encode* Token JWT oleh Peretas

Pada gambar 4.16 menunjukkan bahwa hasil *decode* dari token yang telah dibuat sebelumnya memiliki tipe token, algoritma dan struktur *payload* yang sama dengan token yang dibuat oleh server.



Gambar 4.16 Hasil *Decode* Token JWT oleh Peretas

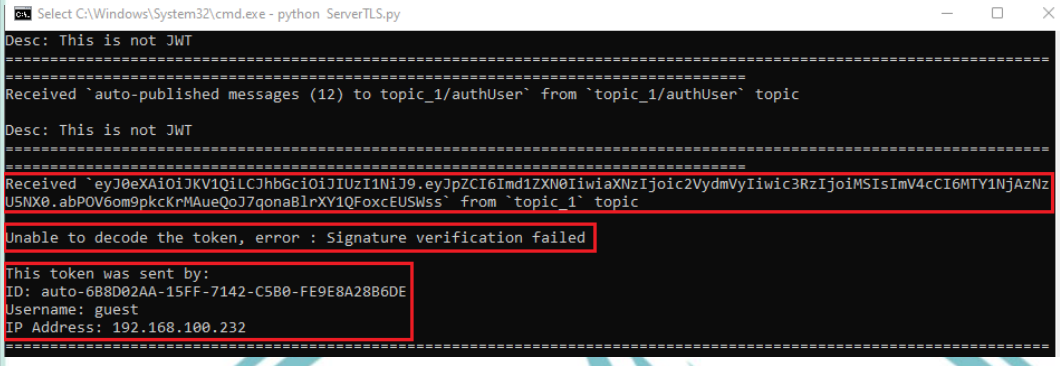
Pada gambar 4.17 menunjukkan bahwa peretas menggunakan akun dengan nama pengguna guest telah melakukan *publish* yang berisi token yang telah dibuat sebelumnya pada topic_1. Topik tersebut digunakan oleh server sebagai *subscriber* untuk melayani setiap pesan yang masuk.



Gambar 4.17 Peretas Melakukan *Publish* Token

Server akan menerima seluruh pesan yang di-*publish* pada topic_1 dan mengenali apakah pesan tersebut merupakan sebuah token. Jika pesan tersebut adalah token, maka server akan melakukan validasi. Apabila token tersebut telah terdeteksi memiliki tanda tangan invalid dan memiliki struktur payload yang sama dengan token yang dibuat oleh server maka server akan memberikan informasi dan

notifikasi melalui aplikasi telegram berupa id, nama pengguna dan alamat ip pengirim token, seperti yang dapat dilihat pada gambar 4.18 dan 4.19.



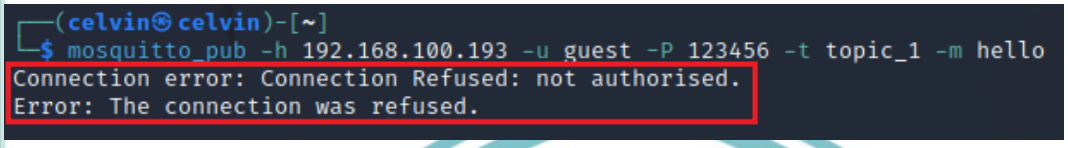
Gambar 4.18 Respon Server Terhadap Token *Invalid Signature*

Gambar 4.19 Notifikasi Telegram Berisi Token dengan *Invalid Signature*

Administrator dapat melakukan blokir terhadap akses dari pengirim token palsu berdasarkan alamat IP yang diperoleh pada notifikasi melalui aplikasi telegram yang tertera pada gambar 4.20.

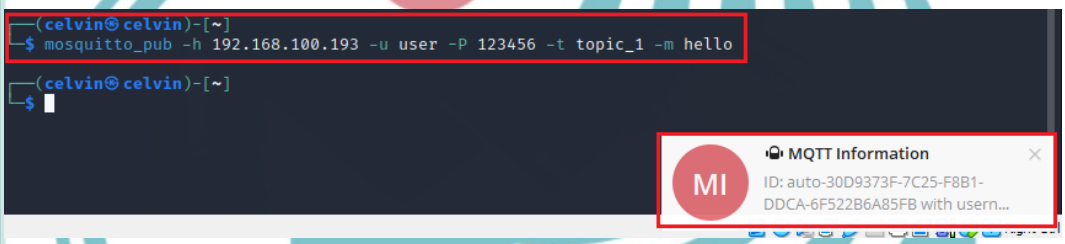
Gambar 4.20 Blokir Alamat IP melalui Telegram

Pada gambar 4.21 menunjukkan bahwa setelah alamat IP diblokir oleh administrator, peretas tidak dapat terkoneksi dengan MQTT broker. Pada gambar tersebut terlihat bahwa koneksi telah ditolak dan pengguna tidak dapat terotorisasi.



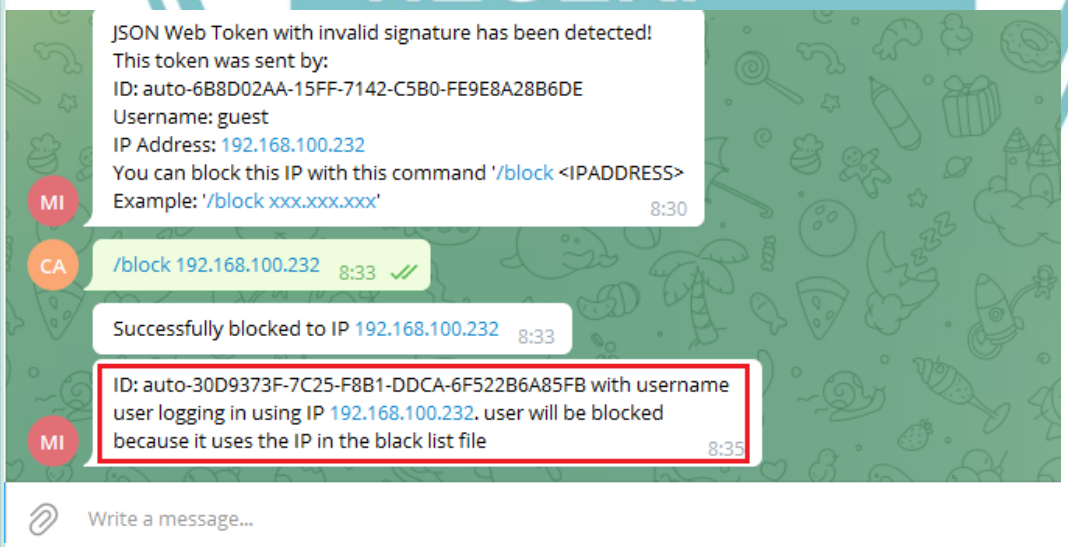
Gambar 4.21 Peretas Tidak Dapat Terhubung dengan *Broker*

Pada gambar 4.22, peretas berupaya melakukan *publish* dengan menggunakan akun lain yang memiliki nama pengguna user akan tetapi dengan isi pesan bukan berupa token JWT melainkan ‘hello’.



Gambar 4.22 Peretas Melakukan *Publish* dengan Akun lain

Pada gambar 4.23 menunjukkan bahwa *Broker* segera mendeteksi bahwa terjadi tindakan *publish* dari alamat ip yang sama dengan alamat ip peretas walaupun menggunakan akun berbeda. Kemudian *broker* melakukan blokir secara otomatis pada pengguna tersebut.



Gambar 4.23 Notifikasi Telegram Berisi Blokir Secara Otomatis

```
┌──(celvin㉿celvin)-[~]
└─$ mosquitto_pub -h 192.168.100.193 -u user -P 123456 -t topic_1 -m hello
Connection error: Connection Refused: not authorised.
Error: The connection was refused.

┌──(celvin㉿celvin)-[~]
└─$
```

Gambar 4.24 Peretas Tidak Dapat Terhubung Menggunakan Akun lain

Pada gambar 4.24 menunjukkan bahwa setelah broker memblokir alamat IP peretas yang menggunakan akun berbeda. Akun tersebut tidak dapat terkoneksi dengan MQTT broker.

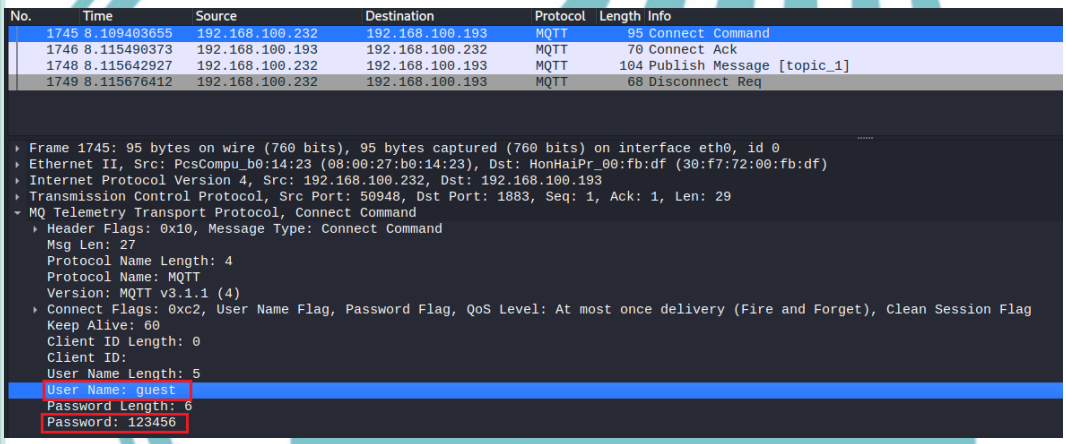
2) Pengujian Performansi

Pengujian performansi dilakukan untuk mengetahui berapa lama waktu yang dibutuhkan oleh klien yaitu NodeMCU ESP8266 untuk dapat memperoleh token JWT dari server. Pengukuran waktu dihitung ketika NodeMCU ESP8266 melakukan *request* token ke server hingga token tersebut diterima. Pengujian dilakukan sebanyak lima puluh kali. Pada gambar 4.25 menunjukkan bahwa setelah NodeMCU ESP8266 menerima token dan melakukan *decode* pada token tersebut, terlihat isi dari *payload* dari token yang terdiri dari id yang merujuk pada pemilik token, iss yang berisi nama pembuat token dan sts yang berisi status token. Apabila nilai sts berisi ‘regenerate-token’ maka secara otomatis NodeMCU ESP8266 akan melakukan *request* token baru pada server. Pada gambar tersebut juga terlihat pada salah satu pengujian yang dilakukan yaitu pengujian keenam belas, waktu yang diperlukan NodeMCU ESP8266 untuk memperoleh token adalah 494 *millisecond* atau 0,494 detik.

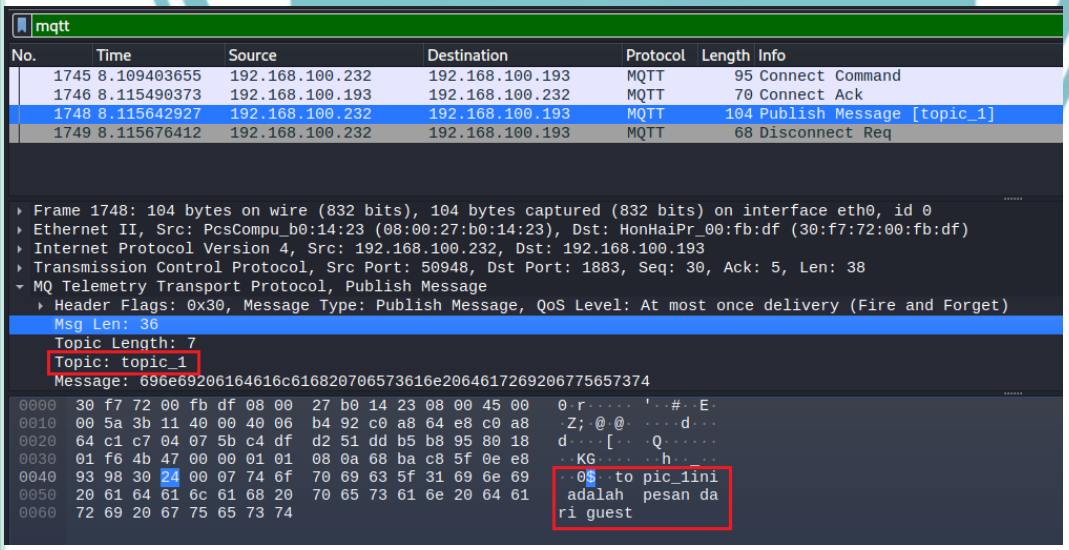
Gambar 4.25 Hasil Waktu *Generate* Token pada Salah Satu Pengujian

3) Pengujian resistansi

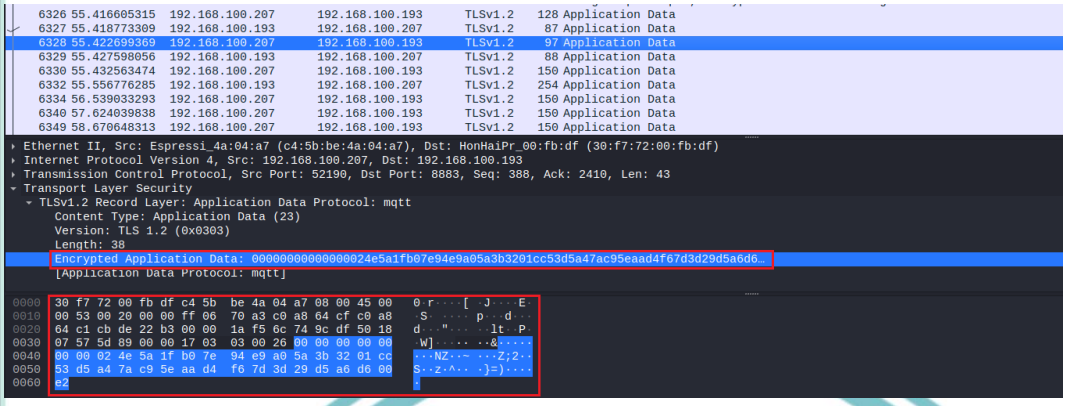
Pada pembuatan sistem ini, MQTT *broker* dibangun untuk dapat melayani koneksi tiap klien melalui port 1883 dan port 8883. Port 1883 hanya melayani klien yang melakukan koneksi atau pertukaran data melalui protokol MQTT tanpa adanya enkripsi. Sedangkan port 8883 digunakan oleh klien untuk melakukan pertukaran data menggunakan protokol keamanan TLS dengan menggunakan sertifikat yang telah digenerate oleh *broker*. Pada gambar 4.26 dan 4.27 menunjukkan bahwa klien yang melakukan koneksi menggunakan port *default* yaitu 1883 dapat terkena serangan *man in the middle* berupa *sniffing* sehingga informasi penting seperti isi pesan, nama pengguna dan kata sandi dari akun tersebut dapat terlihat dengan jelas dengan menggunakan aplikasi wireshark.



Gambar 4.26 Informasi Akun yang Berhasil Ditangkap oleh Wireshark



Gambar 4.27 Isi Pesan yang Berhasil Ditangkap oleh Wireshark



Gambar 4.28 Hasil *Capturing* TLS oleh Wireshark

Pada gambar 4.28 menunjukkan bahwa klien yang melakukan koneksi dan pertukaran data melalui port 8883 dengan menggunakan protokol keamanan TLS dapat terlindungi dari serangan *man in the middle* berupa *sniffing*. Gambar tersebut menunjukkan bahwa pesan yang berhasil di-*capture* telah terenkripsi sehingga tidak dapat terlihat dengan jelas informasi berupa isi pesan, nama pengguna maupun kata sandi.

4.4.4 Evaluasi Pengujian

Setelah pengujian dilakukan, data hasil pengujian dianalisis sesuai dengan masing-masing pengujiannya. Berikut ini merupakan hasil analisis data pengujian.

1. Pengujian Fungsional

Hasil pengujian fungsional yang telah dilakukan pada sistem ini dapat dilihat pada tabel 4.4.

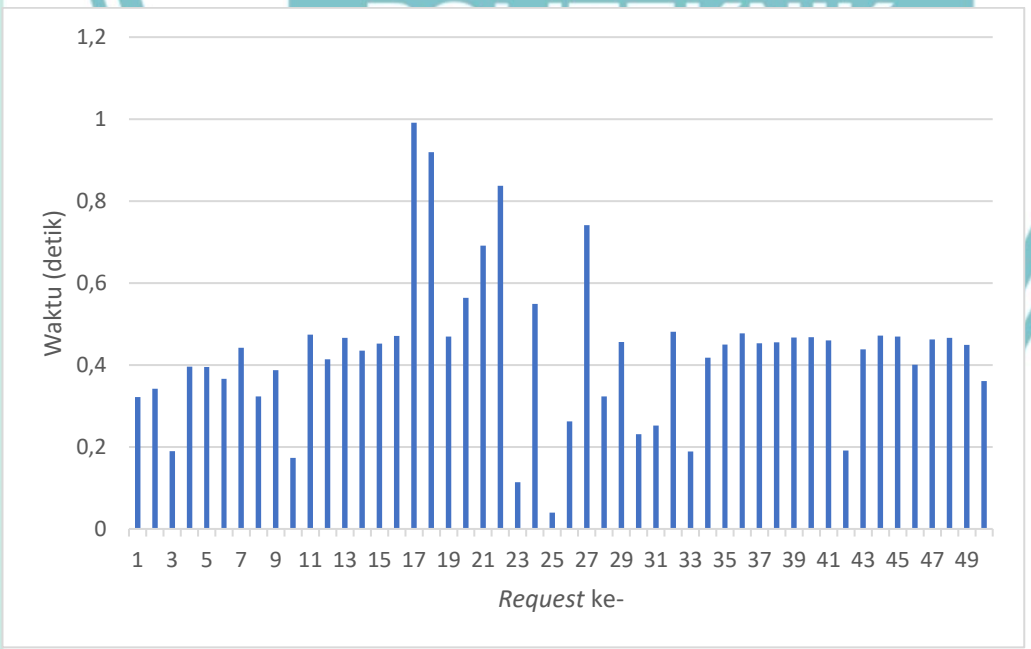
Tabel 4.4 Hasil Pengujian Fungsional

No	Kebutuhan Fungsional	Keterangan (✓)	
		Berhasil	Tidak berhasil
1	Server dapat melakukan autentikasi dengan token yang valid	✓	
2	Server dapat mendeteksi token yang telah kedaluwarsa atau *expired token*	✓	
3	Server dapat mendeteksi token yang memiliki tanda tangan *invalid*	✓	
4	Server dapat mengirimkan notifikasi melalui aplikasi telegram jika	✓	

	terdeteksi adanya token yang memiliki tanda tangan *invalid*		
5	MQTT *broker* dapat melakukan blokir terhadap akses dari alamat IP MQTT klien sesuai dengan *input* yang diberikan oleh administrator melalui aplikasi telegram	✓	

Pada tabel 4.4 menunjukkan bahwa seluruh kebutuhan fungsional pada sistem telah berhasil dipenuhi. Server dapat melakukan autentikasi pada token yang valid, kedaluwarsa dan token yang memiliki tanda tangan *invalid*. Untuk memberikan respon terhadap diterimanya token dengan tanda tangan *invalid*, server berhasil mengirimkan notifikasi melalui aplikasi telegram agar dapat diterima oleh administrator untuk tindakan lebih lanjut. Selain itu, pada tabel tersebut juga dapat diketahui bahwa setelah administrator melakukan blokir terhadap alamat IP peretas yang diperoleh dari aplikasi telegram, MQTT *broker* berhasil melakukan blokir pada alamat IP tersebut.

2. Pengujian Performansi



Gambar 4.29 Grafik Waktu *Request* Token

Gambar 4.29 merupakan grafik yang menunjukkan hasil dari pengujian performansi yang telah dilakukan untuk mengetahui berapa lama waktu yang dibutuhkan oleh klien yaitu NodeMCU ESP8266 dapat memperoleh token JWT yang dibuat oleh server. Pengujian dilakukan sebanyak lima puluh kali dan berjalan menggunakan protokol keamanan TLS. Perhitungan waktu dimulai ketika NodeMCU ESP8266 melakukan *request* token hingga token tersebut diterima. Pada grafik tersebut menunjukkan bahwa waktu paling sedikit yang dibutuhkan yaitu 0,04 detik, waktu yang paling banyak dibutuhkan yaitu 0,991 detik dan rata-rata waktu yang dibutuhkan oleh NodeMCU ESP8266 untuk memperoleh token JWT berdasarkan hasil pengujian ini yaitu sebesar 0,43028 detik.

3. Pengujian Resistansi

Setelah dilakukan pengujian resistansi serangan *man in the middle* pada sistem ini, maka dapat diketahui bahwa penerapan protokol keamanan TLS berhasil menangkal serangan *man in the middle*. Hal ini dapat dilihat dari pembahasan sebelumnya bahwa setiap pertukaran data yang dilakukan diatas protokol TLS dienkripsi sehingga peretas tidak dapat memperoleh informasi penting dari hasil paket yang telah di-*capture*.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil dari penelitian yang telah dilakukan, maka dapat disimpulkan bahwa:

1. Sistem dapat melakukan autentikasi pada setiap token JWT yang diterima.
2. Sistem dapat mendeteksi token JWT yang memiliki tanda tangan *invalid* dan mengirimkan notifikasi melalui aplikasi Telegram.
3. NodeMCU ESP8266 membutuhkan waktu untuk memperoleh token paling sedikit yaitu 0,04 detik, paling banyak yaitu 0,991 detik dan rata-rata waktu yang dibutuhkan adalah sebesar 0,43028 detik.
4. Sistem yang telah dibangun dapat menangkal serangan *man in the middle* berupa *sniffing* dengan menerapkan protokol keamanan *Transport Layer Security* (TLS) versi 1.2 sehingga mampu mencegah peretas untuk dapat memperoleh informasi penting dari lalu lintas data yang di-*capture*.

5.2 Saran

Berdasarkan hasil penelitian ini, maka dapat dilakukan beberapa peningkatan yang bisa diimplementasikan, berikut diantaranya

1. Melakukan analisis terhadap penggunaan memori pada perangkat NodeMCU ESP8266.
2. Menerapkan enkripsi tambahan pada NodeMCU ESP8266. Hal ini dikarenakan data yang di-*capture* melalui lalu lintas data antara MQTT klien dan *broker* yang menggunakan protokol keamanan TLS dan telah terenkripsi masih dapat didekripsi menggunakan teknik *brute force*.

DAFTAR PUSTAKA

Ahmed, S. & Mahmood, Q. (2019) 'An authentication based scheme for applications using JSON web token', *Proceedings - 22nd International Multitopic Conference, INMIC 2019* [Preprint]. doi:10.1109/INMIC48123.2019.9022766.

Bhawiyuga, A., Data, M. & Warda, A. (2018) 'Architectural design of token based authentication of MQTT protocol in constrained IoT device', *Proceeding of 2017 11th International Conference on Telecommunication Systems Services and Applications, TSSA 2017*, 2018-January, pp. 1–4. doi:10.1109/TSSA.2017.8272933.

Boyd, B. dkk. (2014) *Building Real-time mobile solutions with MQTT and IBM MessageSight*. IBM Redbooks. Available at: https://www.redbooks.ibm.com/abstracts/sg248228.html?Open (Accessed: 15 March 2022).

Endra, R.Y. dkk. (2019) *Smart Room Menggunakan Internet Of Things Untuk Efisiensi Biaya dan Keamanan Ruangan*. Bandar Lampung: Aura Publishing. Available at: https://publikasi.ubl.ac.id/index.php/Monograf/catalog/book/34 (Accessed: 30 March 2022).

Fahana, J., Umar, R. & Ridho, F. (2017) 'Pemanfaatan Telegram Sebagai Notifikasi Serangan untuk Keperluan Forensik Jaringan', *Query: Journal of Information Systems*, 1(2), pp. 6–14. Available at: http://jurnal.uinsu.ac.id/index.php/query/article/view/1036 (Accessed: 26 March 2022).

Jones, M., Bradley, J. & Sakimura, N. (2015) 'JSON Web Token (JWT)', pp. 1–38. Available at: https://www.ietf.org/archive/id/draft-ietf-oauth-json-web-token-13.pdf (Accessed: 15 March 2022).

Kashyap, M., Sharma, V. & Gupta, N. (2018) 'Taking MQTT and NodeMcu to IOT: Communication in Internet of Things', *Procedia Computer Science*, 132, pp. 1611–1618. doi:10.1016/J.PROCS.2018.05.126.

Kodali, R.K. & Mahesh, K.S. (2016) 'A low cost implementation of MQTT using ESP8266', *Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics, IC3I 2016*, pp. 404–408.

doi:10.1109/IC3I.2016.7917998.

Lampkin, V. dkk. (2012) *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry*, *IBM Redbooks*. IBM Redbooks. Available at: https://books.google.com/books/about/Building_Smarter_Planet_Solutions_with_M.html?hl=id&id=F_HHAgAAQBAJ (Accessed: 15 March 2022).

Li, S., Xu, L. Da & Zhao, S. (2015) 'The internet of things: a survey', *Information Systems Frontiers 2014 17:2*, 17(2), pp. 243–259. doi:10.1007/S10796-014-9492-7.

Locke, D. (2010) 'MQ Telemetry Transport (MQTT) V3. 1 Protocol Specification', p. 15.

Lundgren, L. (2016) 'Lightweight Protocol! Serious Equipment! Critical Implications!', in. San Francisco, pp. 1–33.

Mahmoud, R. dkk. (2016) 'Internet of things (IoT) security: Current status, challenges and prospective measures', *2015 10th International Conference for Internet Technology and Secured Transactions, ICITST 2015*, pp. 336–341. doi:10.1109/ICITST.2015.7412116.

De Oliveira, J.C., Santos, D.H. & Neto, M.P. (2016) 'Chatting with Arduino platform through Telegram Bot', *Proceedings of the International Symposium on Consumer Electronics, ISCE*, pp. 131–132. doi:10.1109/ISCE.2016.7797406.

Panjaitan, F. & Syafari, R. (2019) 'PEMANFAATAN NOTIFIKASI TELEGRAM UNTUK MONITORING JARINGAN', *Simetris: Jurnal Teknik Mesin, Elektro dan Ilmu Komputer*, 10(2), pp. 725–732. doi:10.24176/SIMET.V10I2.3530.

Prantl, T. dkk. (2021) 'Performance Impact Analysis of Securing MQTT Using TLS', *ICPE 2021 - Proceedings of the ACM/SPEC International Conference on Performance Engineering*, pp. 241–248. doi:10.1145/3427921.3450253.

Setiyadi, A. (2017) 'Implementasi Modul Network MITM Pada Websploit sebagai Monitoring Aktifitas Pengguna dalam Mengakses Internet', *Jurnal Teknik Informatika FTIK UNIKOM*, pp. 113–120. Available at: https://ojs.unikom.ac.id/index.php/senaski/article/view/934 (Accessed: 30 March 2022).

Shingala, K. (2019) 'JSON Web Token (JWT) based client authentication in Message Queuing Telemetry Transport (MQTT)', pp. 1–14. doi:10.48550/arxiv.1903.02895.

Singh, M. *dkk.* (2015) 'Secure MQTT for Internet of Things (IoT)', *Proceedings - 2015 5th International Conference on Communication Systems and Network Technologies, CSNT 2015*, pp. 746–751. doi:10.1109/CSNT.2015.16.

Tareq, M. (2021) 'Man In The Middle Attack in Wireless Network'. Available at: https://www.researchgate.net/publication/356290033_Man_In_The_Middle_Attack_in_Wireless_Network (Accessed: 15 March 2022).

Turner, S. (2014) 'Transport layer security', *IEEE Internet Computing*, 18(6), pp. 60–63. doi:10.1109/MIC.2014.126.

Vannebäck, E. (2018) 'Using the Mosquitto implementation in an embedded environment', *Umeå University*, p. 56.

Wahyudi, A. and Suhartati, A. (2016) 'Implementasi Otomatisasi Mesin Grating Menggunakan Mikrokontroler Arduino MEGA 2560', pp. 177–187. Available at: https://journal.untar.ac.id/index.php/tesla/article/view/304/248 (Accessed: 30 March 2022).

Yudha Saputra, G. dkk. (2017) 'Penerapan Protokol MQTT Pada Teknologi Wan (Studi Kasus Sistem Parkir Universitas Brawijaya)', *Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer*, 12(2), pp. 69–75. doi:10.30872/JIM.V12I2.653.

LAMPIRAN

Lampiran 1 Daftar Riwayat Hidup

DAFTAR RIWAYAT HIDUP PENULIS

Celvin Arya Mangkurat

Lulus dari SDN Kapuk 06 Pagi tahun 2012, SMPN 100 Jakarta tahun 2015 dan SMAN 84 Jakarta tahun 2018

Lampiran 2 Isi dari File Skrip ProgramTLS.ino

ProgramTLS

```
#include <ArduinoJson.h>
#include <ArduinoJWT.h>
#include <sha256.h>
#include <PubSubClient.h>
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>

#define CHECK_CA_ROOT

//wifi
const char* ssid = "UNKNOWN_DEVICE";
const char* password_wifi = "mamaratu";

//mqtt
const char* mqtt_broker = "192.168.100.193";
int mqtt_port = 8883;
String clientID = "NodeMCUESP8266";
String mqtt_username = "nodemcu";
String mqtt_password = "123456";

const char* topic2_lvl1 = "topic_2";//subscribe_by_esp_lvl1
const char* topic1_lvl1 = "topic_1";//publish_by_esp_lvl1

const char* topic1_lvl2 = "topic_1/authUser";//publish_by_esp_lvl2

//jwt
ArduinoJWT jwt = ArduinoJWT("secret");
String myToken = "";
String token;

//otoritasi
int msgCount = -1;
bool approved = false;

//TLS
#ifdef CHECK_CA_ROOT
static const char digicert[] PROGMEM = R"EOF(
```

(Lanjutan)

ProgramTLS

```
-----BEGIN CERTIFICATE-----
MIID9zCCAt+gAwIBAgIUNwHE6DSAD2YspzFVmpdZXluTuY4wDQYJKoZIhvcNAQEL
BQAwgYoxCzAJBgNVBAYTAklEMRAwDgYDVQQIDAdKYWthcnRhMQ8wDQYDVQQHDAZK
YWtiYXIxDDAKBgNVBAoMA1BOSjEPMA0GA1UECwwGY2xpZW50MRgwFgYDVQQDDA8x
OTIuMTY4LjEwMC4xOTMxHzAdBgkqhkiG9w0BCQEWEGNlbHZpbkBnbWFpbC5jb20w
HhcNMjIwNjIxMDQlOTIxWhcNMjcwNjI4MDQlOTIxWjCBijELMAkGA1UEBhMCSUQx
EDAOBgNVBAgMB0pha2FydGExDzANBgNVBAcMBkpha2JhcjEMMAoGA1UECgwDUE5K
MQ8wDQYDVQQLDAZjbGllbnQxGDAWBgNVBAMMDzE5Mi4xNjguMTAwLjE5MzEfMB0G
CSqGSIb3DQEJARYQY2VsdmluQGdtYWlsLmNvbTCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBALdRnOAcMMLftIHHoaROCj39k+04riRaeLqrgluTFCqCzF77
LxBtZs0l+CvGJ7SWUpbPHEGBSyJAhH8ZaoiVQI5pLUsytG6Joti7Z3nrhCpnchjL
sPgxv/fJ2FQqn2usPlKtaksUJiqV66NG74I+ti3Yj6vL8alq6jlJN3TAwgnHTcLB
vytlcqpuZwXkZNL+b7PqlwiP6sgfA90wt9SFXtjvjYNdrbK5erLs8ucTHk/k+pLL
3gunG306wDCbbIm8inzQvmqJlUu58Sjd9TM/jE8jsCcyJYzlu3yba65HWKy93cXC
Y6QNFcC4v7xto+lEMkIN6j3YUb+W9NCrpysVHTMCAwEAAaNTMFEwHQYDVR0OBBYE
FNlc9+plNA6ghXVyz8o9SKjElllGMB8GA1UdIwQYMBaAFNlc9+plNA6ghXVyz8o9
SKjElllGMA8GA1UdEwEB/wQFMAMBAf8wDQYJKoZIhvcNAQELBQADggEBAEn0agj7
Zy5jy3gxxQvXAqgsJyWSbWIxly66wsCFYSIPr8+PwzteTwMvfYgaSM9Gn4oNy32v
a5hmIczvqpo8ht9tgJfqtbwvwuukUQ6L2pulB+YK4H+AaR83dPJfaC7IZB378+Xb
LdKlz4wRe7a8+Cj7PopjN+OuMDnZ/HYw7NxYnRhhXdFKxYIKy6qVUsWA+mx3sEXb
ycemS937S5+v2bkUO+RdSsZ3rdZY4Qce6sMi7gWUkIi+82dmG3XZrvA/WnKgTtYp
sHgQx8S4xgJRkaXumRtTFQF7cbnjGQWN6hjsvQ6aYNEB+qsyuCvwamaS22lyQi0y
9K2gvMHhx+D9JEk=
-----END CERTIFICATE-----
)EOF";
#endif

BearSSL::WiFiClientSecure espClient;
PubSubClient client(espClient);

void setup_wifi() {
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password_wifi);
```

(Lanjutan)

ProgramTLS

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(200);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi Connected");
  Serial.print("IP Address : ");
  Serial.println(WiFi.localIP());
}

//menerima pesan
void callback(char* topic, byte* payload, unsigned int length) {
  String pesanCallback;
  Serial.print("Message arrived [");
  Serial.print(topic2_lvl1);
  Serial.print("]: ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
    pesanCallback += String((char)payload[i]);
  }
  //melakukan validasi jwt
  String hasilDecoded;
  boolean decodedToken = jwt.decodeJWT(pesanCallback, hasilDecoded);
  if (decodedToken == 1) {
    Serial.println("\nJWT Decoded: ");
    Serial.println(hasilDecoded);

    //parsing untuk mengubah status sebagai konfirmasi ke server otentikasi
    DynamicJsonDocument doc(1024);
    deserializeJson(doc, hasilDecoded);
    JsonObject obj = doc.as<JsonObject>();
    String id = obj["id"];
    String iss = obj["iss"];
    String expToken = obj["exp"];
    String sts = obj["sts"];

    if (sts == "0" && myToken == ""){
```

(Lanjutan)

ProgramTLS

```
        String pesan = "{\"id\":\"";
        pesan += id;
        pesan += "\",\"iss\":\"";
        pesan += iss;
        pesan += "\",\"exp\":\"";
        pesan += expToken;
        pesan += "\",\"sts\":\"";
        pesan += "1";
        pesan += "\"}";
        token = jwt.encodeJWT(pesan);
        int str_len = token.length()+1;
        char char_array[str_len];
        token.toCharArray(char_array, str_len);

        for (int i = 0; i < str_len; i++)
          Serial.print(char_array[i]);

        Serial.println("");
        myToken = String(char_array);
        Serial.println("---------TOKENS SAVED---------");
        if (client.publish(topicl_lvll, char_array)){
          myToken = String(char_array);
          Serial.println("BERHASIL MENGIRIM KONFIRMASI JWT KE SERVER");
          approved = true;
          if (msgCount < 0){
              msgCount += 1;
            }
          }else{
            Serial.println("GAGAL MENGIRIM KONFIRMASI JWT KE SERVER");
          }
        delay(1000);
    }else if (sts == "Not Aprroved"){
        approved = false;
    }else if (sts == "regenerate-token"){
        approved = false;
        myToken = "";
        Serial.println("---------REGENERATING TOKEN---------");
```

(Lanjutan)

ProgramTLS

```
    }else if (sts == "restart-nodemcu" && iss == "admin"){
        Serial.println("Server shut down. Reset NodeMCU ESP8266");
        ESP.restart();
    }
   } else{
    Serial.println("\nStatus: JWT not Detected");
    if (pesanCallback == "Send Token" && myToken != ""){
        int str_len = myToken.length()+1;
        char char_array[str_len];
        token.toCharArray(char_array, str_len);
        for (int i = 0; i < str_len; i++)
          Serial.print(char_array[i]);
        Serial.println("");
        if (client.publish(topicl_lvl1, char_array)){
          myToken = String(char_array);
          Serial.println("BERHASIL MENGIRIM JWT KE SERVER(SERVER REQUEST)");
        }else{
          Serial.println("GAGAL MENGIRIM JWT KE SERVER(SERVER REQUEST)");
        }
      }
  }
}

//mengirim pesan
void sendMessage(){
      if (msgCount > -1 && approved == true){
        String text = "";
        if (msgCount == 0){
          text = "Start publishing messages to " + String(topicl_lvl2);
        }else{
          text = "auto-published messages (" + String(msgCount)+") to "+ topicl_lvl2;
        }
        int str_len = text.length()+1;
        char char_array[str_len];
        text.toCharArray(char_array, str_len);

        for (int i = 0; i < str_len; i++)
```

(Lanjutan)

ProgramTLS

```
          Serial.print(char_array[i]);

        Serial.println("");
        client.publish(topicl_lvl2, char_array);
        delay(1000);
        msgCount++;
      }
}


//memastikan kalau kita terhubung dengan broker
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    //attempt to connect
    if (client.connect(clientID.c_str(), mqtt_username.c_str(), mqtt_password.c_str())) {
      Serial.println("connected");
      client.subscribe(topic2_lvl1);
      approved = true;
    }
    //jika koneksinya gagal
    else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
    }
  }
}
```

```
void setup() {
  Serial.begin(115200);
  setup_wifi();
  #ifdef CHECK_CA_ROOT
    BearSSL::X509List cert(digicert);
    espClient.setTrustAnchors(&cert);
    espClient.setInsecure();
  #endif
  client.setServer(mqtt_broker, mqtt_port);
  client.setCallback(callback);
  myToken = "";
}

void loop() {
  if (!client.connected()) {
    approved = false;
    reconnect();
  }
  client.loop();
  sendMessage();
}
```

Lampiran 3 Isi dari File Skrip ServerTLS.py

```
from datetime import datetime, timedelta
import sys
import threading
from time import sleep
import subprocess
from ConnectBroker import *
import jwt
from jwt.exceptions import *
from telegram.ext.updater import Updater
from telegram.update import Update
from telegram.ext.callbackcontext import CallbackContext
from telegram.ext.commandhandler import CommandHandler
from telegram.ext.messagehandler import MessageHandler
from telegram.ext.filters import Filters

MQTT_TOPIC = [("topic_1",0),("topic_1/authUser",0)]
topic2_lvl1 = "topic_2" #sebagai publisher
ipBroker = broker

secretKey = "secret"
TokenJWTNodeMCU = ""
exp = ""

tokenTele = "5485208834:AAHyHrjm5Msc1JqqXqibHrpcup4WBuYvxwE"
idChat = "1340083557"

client = connect_mqtt()


def readBlacklist():
    try:
        ip=[]
        file = open(f"D:\mosquitto\\blacklist.txt", "r")
        for line in file:
            line = line.replace("\n","")
            ip.append(line)

        file.close()
        return ip
    except FileNotFoundError:
        print("File blacklist.txt not found")
        file = open(f"D:\mosquitto\\blacklist.txt", "w")
        file.write("")
        file.close()
        return False
```

(Lanjutan)

```
def unblockClient(ipAddress):
    listIP = readBlacklist()
    send = True
    if (listIP != False):
        if (listIP != ""):
            lines=[]
            for line in listIP:
                if (line.rfind(ipAddress) != -1):
                    lineSplit = line.split(":")
                    username=lineSplit[0]
                    ipSplit=lineSplit[1]
                    if (ipSplit == ipAddress):
                        # USING DYNSEC
                        command= f"D:\\mosquitto\\mosquitto_ctrl.exe
-u admin -P 123456 -i admin -h {ipBroker} dynsec enableClient
{username}"
                        subprocess_call(command)
                        if send == True:
                            sendMessageToTele(f"Successfully unblock
IP {ipAddress}")
                            send = False
                    else:
                        sendMessageToTele(f"Failed to unblock.
{ipAddress} not found!")
                        lines.append(line)
                else:
                    lines.append(line)
            file = open("D:\\mosquitto\\blacklist.txt",'w')
            for baris in lines:
                file.write(f"{baris}\n")
            file.close()


def readClientOnline():
    users = []
    file = open("D:\mosquitto\Online Client.txt", "r")
    for line in file:
        line = line.replace("\n","")
        users.append(line)
    file.close()
    return users


def setAccessRole(level, username):
    if (level == "1"):
```

(Lanjutan)

```
        file= f"D:\\mosquitto\\mosquitto_ctrl.exe -h {ipBroker} -u
admin -P 123456 -i admin dynsec removeClientRole {username}
AccessRoleLevel2"
            subprocess_call(file)
            file= f"D:\\mosquitto\\mosquitto_ctrl.exe -h {ipBroker} -u
admin -P 123456 -i admin dynsec addClientRole {username}
AccessRoleLevel1"
            subprocess_call(file)
        elif (level == "2"):
            file= f"D:\\mosquitto\\mosquitto_ctrl.exe -h {ipBroker} -u
admin -P 123456 -i admin dynsec removeClientRole {username}
AccessRoleLevel1"
            subprocess_call(file)
            file= f"D:\\mosquitto\\mosquitto_ctrl.exe -h {ipBroker} -u
admin -P 123456 -i admin dynsec addClientRole {username}
AccessRoleLevel2"
            subprocess_call(file)


import subprocess
IS_WIN32 = 'win32' in str(sys.platform).lower()


def subprocess_call(*args, **kwargs):
    #also works for Popen. It creates a new *hidden* window, so it
will work in frozen apps (.exe).
    if IS_WIN32:
        startupinfo = subprocess.STARTUPINFO()
        startupinfo.dwFlags = subprocess.CREATE_NEW_CONSOLE |
subprocess.STARTF_USESHOWWINDOW
        startupinfo.wShowWindow = subprocess.SW_HIDE
        kwargs['startupinfo'] = startupinfo
    retcode = subprocess.call(*args, **kwargs)
    return retcode

def cekSignature():
    global TokenJWTNodeMCU
    print(TokenJWTNodeMCU)
    while True:
        if TokenJWTNodeMCU != "":
            try:
                jwt.decode(TokenJWTNodeMCU,secretKey,
algorithms=["HS256"])
            except jwt.ExpiredSignatureError:
```

(Lanjutan)

```
      # mengirim token untuk memberi informasi kepada nodemcu
                print(f"TOKEN '{TokenJWTNodeMCU}' FROM USER(nodemcu)
HAS EXPIRED!\n")
                payload_data = {
                    "id"  : "nodemcu",
                    "iss" : "server",
                    "sts" : "regenerate-token"
                }
                encoded_jwt =
jwt.encode(payload_data,secretKey,algorithm="HS256")
                msg = f"{encoded_jwt}"
                result = client.publish(topic2_lvl1, msg)
                # result: [0, 1]
                status = result[0]
                if status == 0:
                    print(f"Send message : '{msg}' to topic :
'{topic2_lvl1}'\n")
                elif status == 4:
                    print("MQTT Broker is offline. You need to
restart this server!")
                    break
                else:
                    print(f"Failed to send message to topic:
{topic2_lvl1}")
                sleep(1)
                # topic_1 "Nodemcu has been connected!"
                file= f"D:\\mosquitto\\mosquitto_pub.exe -h
{ipBroker} -u admin -P 123456 -i admin -t topic_1 -m \"Nodemcu has
been connected!\""
                subprocess.call(file)
                sleep(5)


def publishJWT():
    global exp
    exp = datetime.utcnow()+timedelta(seconds=20)
    print("\n=========GENERATING TOKEN=========")
    payload_data = {
        "id"  : "nodemcu",
        "iss" : "server",
        "sts" : "0",
        "exp" : exp
    }
    encoded_jwt =
jwt.encode(payload_data,secretKey,algorithm="HS256")
```

(Lanjutan)

```
    msg = f"{encoded_jwt}"
    result = client.publish(topic2_lvl1, msg)
    # result: [0, 1]
    status = result[0]
    if status == 0:
        print(f"Send message : '{msg}' to topic : '{topic2_lvl1}'")
    else:
        print(f"Failed to send message to topic: {topic2_lvl1}")

def publishStop():
    payload_data = {
        "id"  : "nodemcu",
        "iss" : "server",
        "sts" : "Not Aprroved"
    }
    encoded_jwt =
jwt.encode(payload_data,secretKey,algorithm="HS256")
    msg = f"{encoded_jwt}"
    result = client.publish(topic2_lvl1, msg)
    # result: [0, 1]
    status = result[0]
    if status == 0:
        print(f"Send message : '{msg}' to topic '{topic2_lvl1}'")
    else:
        print(f"Failed to send message to topic: {topic2_lvl1}")

def subscribe(client: mqtt_client):
    def on_message(client, userdata, msg):
        global TokenJWTNodeMCU
        tokenJWT=msg.payload.decode()
        print(f"Received `{tokenJWT}` from `{msg.topic}` topic")
        try:
            plainMsg = jwt.decode(tokenJWT,secretKey,
algorithms=["HS256"])
            print(f"\n---Decoded---")
            print("id :",plainMsg["id"],"\niss
:",plainMsg["iss"],"\nsts :",plainMsg["sts"],"\nexp
:",plainMsg["exp"])
            if (plainMsg["sts"] == "1") & (plainMsg["id"] ==
"nodemcu"):
                #ubah menjadi AccessRoleLevel2
                setAccessRole("2", plainMsg["id"])
                #Simpan token
                TokenJWTNodeMCU = tokenJWT
        except InvalidSignatureError as error:
```

(Lanjutan)

```
            print(f"\nUnable to decode the token, error : {error}")
            sleep(1)
            try:
                file = open(f"D:\mosquitto\candidate.txt", "r")
                if (file != ""):
                    for line in file:
                        try:
                            sub = line.split(":")
                            id=sub[0]
                            username=sub[1]
                            ipAddress=sub[2]
                            if username != "admin":
                                msg=f"\nThis token was sent by:\nID:
{id}\nUsername: {username}\nIP Address: {ipAddress}"
                                print(f"{msg}")
                                # Mengirim informasi ke telegram
                                msg = f"JSON Web Token with invalid
signature has been detected!{msg}\nYou can block this IP with this
command '/block <IPADDRESS>\nExample: '/block xxx.xxx.xxx'"
                                sendMessageToTele(msg)
                        except:
                            print("")
                file.close()
            except FileNotFoundError:
                print("File candidate.txt not found")
        except ExpiredSignatureError:
            #Signature has expired
            print("\n=======Token has expired.=======")
            setAccessRole("1", "nodemcu")
            publishJWT()
        except:
            print(f"\nDesc: This is not JWT")
            if (tokenJWT == "Nodemcu has been connected!"):
                publishJWT()
            if (tokenJWT == "Denied PUBLISH from node"):
                publishStop()

        print(100*"==")

    client.subscribe(MQTT_TOPIC)
    client.on_message = on_message


# Listen by Telegram
updater = Updater(tokenTele,use_context=True)
```

(Lanjutan)

```
def start(update: Update, context: CallbackContext):
    update.message.reply_text(
        "Hello, Welcome to the MQTT Information Bot. Please write\
        /help to see the commands available.")

def help(update: Update, context: CallbackContext):
    update.message.reply_text("""Available Commands :-
    /onlineClient - To get information of currently connected
clients\n/blacklistClient - To get blocked clients
information\n/block <IPADRESS> - To block IP Address\n/unblock
<IPADRESS> - To unblock IP Address""")


def onlineClient(update: Update, context: CallbackContext):
    users = readClientOnline()
    msg="CURRENTLY CONNECTED CLIENT\n"
    for user in users:
        msg += f"{user}\n"
    update.message.reply_text(msg)


def blacklistClient(update: Update, context: CallbackContext):
    users = readBlacklist()
    if users != False:
        msg="IP Blacklist\n"
        tmp = ""
        for user in users:
            ipSplit = user.split(":")
            if tmp != ipSplit[1]:
                msg += f"{ipSplit[1]}\n"
                tmp = ipSplit[1]
        update.message.reply_text(f"{msg}")
    else:
        msg="IP Blacklist\n"
        update.message.reply_text(f"{msg}")


def unknown(update: Update, context: CallbackContext):
    msg = update.message.text
    subMsg = msg.split(" ")
    if (subMsg[0] == "/block"):
        # cek candidate.txt jika ip tercatat. subMsg[1] berisi IP
        try:
            file = open(f"D:\mosquitto\candidate.txt", "r")
            if (file != ""):
```

(Lanjutan)

```
                for line in file:
                    try:
                        sub = line.split(":")
                        id=sub[0]
                        username=sub[1]
                        ipAddress=sub[2]
                        try:
                            if ipAddress == subMsg[1]:
                                # Memeriksa di file text jika ip
belum di block
                                try:
                                    users = readBlacklist()
                                    ipTarget=subMsg[1]
                                    ipFound = False
                                    for user in users:
                                        userSplit=user.split(":")
                                        usernameSplit = userSplit[0]
                                        ipSplit = userSplit[1]
                                        if (ipSplit == ipTarget) &
(usernameSplit == username):
                                            update.message.reply_tex
t(f"Unable to block because IP ({subMsg[1]}) has been blocked")
                                            ipFound = True
                                            break
                                        ipFound = False

                                    if ipFound == False:
                                        file =
open(f"D:\mosquitto\\blacklist.txt", "a")
                                        file.write(f"{username}:{ipT
arget}\n")
                                        file.close()
                                        # USING DYNSEC FOR BLOCK
                                        if (username != "server") &
(username != "nodemcu") & (username != "admin"):
                                            command=
f"D:\\mosquitto\\mosquitto_ctrl.exe -u admin -P 123456 -i admin -h
{ipBroker} dynsec disableClient {username}"
                                            subprocess_call(command)
                                            update.message.reply_tex
t(f"Successfully blocked to IP {subMsg[1]}")
                                        else:
                                            update.message.reply_tex
t(f"Sorry you can't block to IP {subMsg[1]} with this
username({username})")
```

(Lanjutan)

```
                                except FileNotFoundError:
                                    #Make A File
                                    file2 =
open(f"D:\\mosquitto\\blacklist.txt", "w")
                                    file2.write("")
                                    file2.close
                                    update.message.reply_text(f"File
dibuat")
                            else:
                                update.message.reply_text(
                                    f"Failed to block! IP
({subMsg[1]}) is not available")
                        except:
                            update.message.reply_text("Please input
the IP address")
                    except IndexError:
                        update.message.reply_text("Sorry the service
is currently unavailable. Type /help for mor information.")
            file.close()
        except FileNotFoundError:
            print("File blacklist.txt not found")
    elif (subMsg[0] == "/unblock"):
        try:
            ipAddress = subMsg[1]
            unblockClient(ipAddress)
        except IndexError:
            update.message.reply_text("Sorry the service is
currently unavailable. Type /help for mor information.")
    else:
        update.message.reply_text(
            "Sorry '%s' is not a valid command. You can type
'/help'" % update.message.text)


def unknown_text(update: Update, context: CallbackContext):
    update.message.reply_text(
        "Sorry I can't recognize you , you said '%s'" %
update.message.text)

# send Message by server
def sendMessageToTele(msg):
    updater.bot.sendMessage(chat_id=idChat, text=msg)


updater.dispatcher.add_handler(CommandHandler('start', start))
```

(Lanjutan)

```
updater.dispatcher.add_handler(CommandHandler('help', help))
updater.dispatcher.add_handler(CommandHandler('onlineClient',
onlineClient))
updater.dispatcher.add_handler(CommandHandler('blacklistClient',
blacklistClient))
updater.dispatcher.add_handler(MessageHandler(Filters.text,
unknown))
updater.dispatcher.add_handler(MessageHandler(
    Filters.command, unknown)) # Filters out unknown commands

# Filters out unknown messages.
def runTelegram():
    updater.dispatcher.add_handler(MessageHandler(Filters.text,
unknown_text))
    updater.start_polling()



updaterSend = Updater(tokenTele,use_context=True)
def sendMessageTele(msg):
    updaterSend.bot.sendMessage(chat_id=idChat, text=msg)



def run():
    try:
        t = threading.Thread(name='runtelegram', target=runTelegram)
        t.daemon = True
        t.start()
    except Exception as e:
        print("Error ni thread"+e)
    sendMessageTele("Server is online!")
    c = threading.Thread(name='ceksignaturebackground',
target=cekSignature)
    c.daemon = True
    c.start()
    setAccessRole("1","nodemcu")
    subscribe(client)
    client.loop_forever()

if __name__ == '__main__':
    run()
```

Lampiran 4 Isi dari File Skrip Log.py

```
from time import sleep
import os
import subprocess
from telegram.ext.updater import Updater
import subprocess
import sys
import jwt
from ConnectBroker import broker,dirtls

ipBroker = broker
dir_tls = dirtls

def follow(thefile):
    thefile.seek(0, os.SEEK_END)
    while True:
        line = thefile.readline()
        if not line:
            sleep(0.1)
            continue
        yield line

IS_WIN32 = 'win32' in str(sys.platform).lower()

def subprocess_call(*args, **kwargs):
    #also works for Popen. It creates a new *hidden* window, so it
will work in frozen apps (.exe).
    if IS_WIN32:
        startupinfo = subprocess.STARTUPINFO()
        startupinfo.dwFlags = subprocess.CREATE_NEW_CONSOLE |
subprocess.STARTF_USESHOWWINDOW
        startupinfo.wShowWindow = subprocess.SW_HIDE
        kwargs['startupinfo'] = startupinfo
    retcode = subprocess.call(*args, **kwargs)
    return retcode

def writeClientOnline():
    file = open("D:\\mosquitto\\Online Client.txt",'w')
    file.write("--Online Client--\n")
    file.close()

def readClientOnline():
    users = []
    file = open("D:\mosquitto\Online Client.txt", "r")
    for line in file:
        line = line.replace("\n","")
```

(Lanjutan)

```
        users.append(line)
    file.close()
    return users

def addClientOnline(username):
    users = readClientOnline()
    if username in users:
        print(f"{username} still online!")
    else:
        file = open("D:\\mosquitto\\Online Client.txt",'a')
        file.write(username+"\n")
        file.close()


def removeClientOnline(username):
    users=readClientOnline()
    if username in users:
        users.remove(username)
        file = open("D:\\mosquitto\\Online Client.txt",'w')
        for element in users:
            file.write(element + "\n")
        file.close()
    else:
        print(f"{username} has been disconnected!")


def logFile(dir):
    try:
        lines = []
        file = open(f"D:\mosquitto\{dir}", "r")
        if (file != ""):
            for line in file:
                line = line.replace("\n","")
                lines.append(line)
            file.close()
            return lines
        else:
            return ""
    except FileNotFoundError:
        if (dir == "candidate.txt"):
            file = open("D:\\mosquitto\\candidate.txt",'w')
            file.write("---Candidate Client---")
            file.close()
        return ""
```

(Lanjutan)

```
def writeUsernameIP(id, panjangPesan):
    # untuk mencari username dari log
    lines=logFile("log.log")
    if(lines != ""):
        for line in lines:
            if (line.find("New client connected from") != -1) &
(line.rfind(id) != -1):
                lineSplit = line.split(" ")
                userSplit = lineSplit[11].split("'")
                username=userSplit[1]
                ipSplit = lineSplit[5].split(":")
                ipAddress=ipSplit[0]
                userKeyword =
f"{id}:{username}:{ipAddress}:{panjangPesan}"
                users=logFile("candidate.txt")
                if users != "":
                    count = 0
                    for user in users:
                        if count > 0:
                            line = user.split(":")
                            file =
open("D:\\mosquitto\\candidate.txt",'w')
                            file.write(userKeyword)
                            file.close()
                        count = count + 1
                    # if count == 1:
                    #     file =
open("D:\\mosquitto\\candidate.txt",'a')
                    #     file.write(userKeyword+"\n")
                    #     file.close()
                    if count == 1:
                        file =
open("D:\\mosquitto\\candidate.txt",'w')
                        file.write(userKeyword)
                        file.close()



def publish(command):
    if command == "Request JWT":
        file= f"D:\\mosquitto\\mosquitto_pub.exe -h {ipBroker} -t
topic_1 -u admin -P 123456 -i admin -m \"Nodemcu has been
connected!\""
        subprocess.call(file)
    elif command == "Send JWT":
```

(Lanjutan)

```
        file= f"D:\\mosquitto\\mosquitto_pub.exe -h {ipBroker} -t
topic_2 -u admin -P 123456 -i admin -m \"Send Token\""
        subprocess.call(file)
    elif command == "Refresh NodeMCU":
        payload_data = {
        "iss" : "admin",
        "sts" : "restart-nodemcu",
        }
        encoded_jwt =
jwt.encode(payload_data,"secret",algorithm="HS256")
        msg = f"{encoded_jwt}"
        file= f"D:\\mosquitto\\mosquitto_pub.exe -h {ipBroker} -t
topic_2 -u admin -P 123456 -i admin -m {msg}"
        subprocess.call(file)


def readBlacklist():
    try:
        ip=[]
        file = open(f"D:\mosquitto\\blacklist.txt", "r")
        for line in file:
            line = line.replace("\n","")
            ip.append(line)

        file.close()
        return ip
    except FileNotFoundError:
        print("File blacklist.txt not found")
        file = open(f"D:\mosquitto\\blacklist.txt", "w")
        file.write("")
        file.close()
        return False


tokenTele = "5485208834:AAHyHrjm5Msc1JqqXqibHrpcup4WBuYvxwE"
idChat = "1340083557"
updater = Updater(tokenTele,use_context=True)

# send Message To Tele
def sendMessageToTele(msg):
    updater.bot.sendMessage(chat_id=idChat, text=msg)



if __name__ == '__main__':
    writeClientOnline()
```

(Lanjutan)

```
    logFile("candidate.txt")
    logfile = open("D:\mosquitto\log.log","r")
    loglines = follow(logfile)

    for line in loglines:
        print(line, end='')
        if (line.find("New client connected from") != -1):
            lineSplit = line.split(" ")
            #['2022-06-11T21:03:01:', 'New', 'client', 'connected',
'from', '192.168.100.193:55851', 'as', 'NodeMCUESP8266', '(p2,',
'c1,', 'k60,', "u'nodemcu').\n"]
            ipSplit = lineSplit[5].split(":")
            # Baca Blacklist
            users=readBlacklist()
            if (users != False):
                if (users != ""):
                    for user in users:
                        if (user.rfind(ipSplit[0]) != -1):
                            userSplit = user.split(":")
                            usrnm=userSplit[0]
                            ipAdd=userSplit[1]
                            # lineSplit[11] : u'nodemcu').
                            subUser = lineSplit[11].split("'")
                            if (ipSplit[0] == ipAdd) & (subUser[1]
!= "admin") & (subUser[1] != "server") & (subUser[1] != "nodemcu"):
                                file =
open(f"D:\mosquitto\\blacklist.txt", "a")
                                file.write(f"{subUser[1]}:{ipSplit[0
]}\n")
                                file.close()
                                command=
f"D:\\mosquitto\\mosquitto_ctrl.exe -u admin -P 123456 -i admin -h
{ipBroker} dynsec disableClient {subUser[1]}"
                                subprocess_call(command)
                                sendMessageToTele(f"ID:
{lineSplit[7]} with username {subUser[1]} logging in using IP
{ipAdd}. {subUser[1]} will be blocked because it uses the IP in the
black list file")

            # menambahkan klien online ke OnlineClient.txt
            addClientOnline(lineSplit[7])
            if (lineSplit[7] == "NodeMCUESP8266"):
                # mengirim informasi ke server bahwa nodemcu telah
terkoneksi agar dapat memperoleh JWT
                publish("Request JWT")
```

(Lanjutan)

```
        elif (line.find("Client") != -1 ) &
(line.rfind("disconnected.") != -1):
            lineSplit = line.split(" ")
            # ['2022-06-12T11:28:14:', 'Client', 'server',
'disconnected.\n']
            print(f"User({lineSplit[2]}) has been disconnect as
Publisher!")
            # menghapus klien online ke OnlineClient.txt
            removeClientOnline(lineSplit[2])
        elif (line.find("Client") != -1 ) & (line.rfind("closed its
connection.") != -1):
            lineSplit = line.split(" ")
            # 2022-06-12T12:13:04: Client server closed its
connection.
            print(f"User({lineSplit[2]}) has been disconnect as
Subscriber!")
            # menghapus klien online ke OnlineClient.txt
            removeClientOnline(lineSplit[2])
            # memberi informasi bahwa server shutdown
            if (lineSplit[2] == "server"):
                publish("Refresh NodeMCU")
        elif (line.find("Denied PUBLISH from NodeMCUESP8266") != -
1):
            # 2022-06-13T16:44:32: Denied PUBLISH from
NodeMCUESP8266 (d0, q0, r0, m0, 'topic_1/authUser', ... (48 bytes))
            publish("Send JWT")
        elif (line.find("Received PUBLISH from") != -1):
            lineSplit = line.split(" ")
            # ['2022-06-13T19:37:31:', 'Received', 'PUBLISH',
'from', 'auto-0D395F7D-559C-64E0-8174-AD939D8FDB1D', '(d0,', 'q0,',
'r0,', 'm0,', "'topic_1',", '...', '(141', 'bytes))\n']
            panjangPesan = lineSplit[11].replace("(", "")
            if (int(panjangPesan) >= 100) & (lineSplit[4] !=
"server") & (lineSplit[4] != "NodeMCUESP8266") & (lineSplit[4] !=
"admin"):
                writeUsernameIP(lineSplit[4], panjangPesan)
```

Lampiran 5 Tabel Hasil Pengujian Performansi

No	Pengujian ke-	*Millisecond*	Detik
1	1	322	0,322
2	2	342	0,342
3	3	190	0,19
4	4	396	0,396
5	5	395	0,395
6	6	366	0,366
7	7	442	0,442
8	8	323	0,323
9	9	387	0,387
10	10	173	0,173
11	11	474	0,474
12	12	414	0,414
13	13	466	0,466
14	14	435	0,435
15	15	452	0,452
16	16	471	0,471
17	17	991	0,991
18	18	919	0,919
19	19	469	0,469
20	20	564	0,564
21	21	691	0,691
22	22	837	0,837
23	23	114	0,114
24	24	549	0,549
25	25	40	0,04
26	26	262	0,262
27	27	741	0,741
28	28	323	0,323
29	29	456	0,456
30	30	231	0,231
31	31	252	0,252
32	32	481	0,481
33	33	189	0,189
34	34	418	0,418
35	35	450	0,45
36	36	477	0,477
37	37	453	0,453
38	38	455	0,455
39	39	467	0,467
40	40	468	0,468
41	41	460	0,46
42	42	191	0,191
43	43	438	0,438
44	44	472	0,472

(Lanjutan)

45	45	469	0,469
46	46	401	0,401
47	47	462	0,462
48	48	466	0,466
49	49	449	0,449
50	50	361	0,361
Nilai rata-rata		430,28	0,43028