Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 2, No. 2, Februari 2018, hlm. 584-593 http://j-ptiik.ub.ac.id

Implementasi Autentikasi JSON Web Token (JWT) Sebagai Mekanisme Autentikasi Protokol MQTT Pada Perangkat NodeMCU

Andri Warda Pratama Putra[1], Adhitya Bhawiyuga[2], Mahendra Data[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]andriwarda95@gmail.com, [2]bhawiyuga@ub.ac.id, [3]mahendra.data@ub.ac.id

Abstrak

IoT adalah mekanisme komunikasi machine-to-machine (M2M) yang diharapkan menjadi salah satu solusi jaringan masa depan. MQTT adalah protokol yang digunakan untuk komunikasi M2M/IoT yang berjalan diatas protokol TCP/IP yang dirancang sebagai broker berdasarkan pertukaran pesan publish/subscribe untuk kode kecil (misalnya 8-bit, 256KB RAM controller), bandwidth dan daya rendah, koneksi dan biaya yang tinggi, ketersediaan variabel, jaminan pengiriman (Špeh & Heđ, 2016). Untuk melakukan autentikasi, MQTT saat ini menggunakan username dan password. JWT adalah token berbentuk string panjang yang sangat random yang gunanya untuk melakukan autentikasi sistem dan pertukaran informasi. JWT mengamankan informasi menjadi sebuah klaim yang di encode ke dalam bentuk JSON dan menjadi payload dari JSON Web Signature (JWS) (Bradley, 2015). Isi dari signature pada JWT merupakan gabungan dari isi header dan payload, jika terjadi perubahan pada header/payload maka signature akan menjadi tidak valid. NodeMCU digunakan sebagai publisher dikarenakan konsumsi daya yang dibutuhkan oleh NodeMCU lebih sedikit jika dibandingkan node sensor lain seperti Raspberry Pi. Pada penelitian ini, dilakukan implementasi JWT pada MQTT dengan mengunakan NodeMCU sebagai publisher. Pada penelitian ini dilakukan tiga pengujian inti diantaranya pengujian validasi username dan password, pengujian expiration token, dan pengujian waktu generate token. Hasil dari penelitian ini adalah JWT yang diimplementasikan dapat melakukan validasi terhadap username dan password yang dikirimkan oleh publisher, JWT mampu melakukan autentikasi terhadap token yang telah expired, dan waktu yang dibutuhkan oleh server untuk men-generate token adalah bervariasi dipengaruhi oleh response server terhadap request publisher bervariasi.

Kata kunci: *IoT, Message Queuing Telemetry Transport (MQTT), JSON Web Token (JWT), NodeMCU*

Abstract

IoT is a concept that aims to expand the benefits of continuously connected internet connectivity. MQTT is a protocol that used for M2M / IoT communications running over TCP / IP protocols designed as brokers based on the exchange of publish / subscribe messages for small code (eg 8-bit, 256KB ram controller), low power and bandwidth, connection and high cost, Availability of varables, negotiation of delivery guarantees (Špeh & Heđ, 2016). To perform client verification, traditional MQTT authentication uses the username and password in the authentication message. JWT is a very random string tokens for system authentication and information exchange. JWT secures information into a claim that is encoded into JSON and becomes a payload of JSON Web Signature (JWS) (Bradley, 2015). Token can be verified and trusted because it has been digitally signed can use HMAC or RSA. The contents of the signature on JWT is a combination of header and payload content, if there is a change in the header / payload then the signature will become invalid. NodeMCU is used as a publisher because the power consumption required by NodeMCU is less when compared to other sensor nodes such as Raspberry Pi. In this research, the implementation of JWT on MQTT using NodeMCU as publisher. In this research, there are three core test such as testing of username and password validation, expiration token test, and testing of token generate time. The result of this research is JWT that was implemented to validate username and password submitted by the publisher, JWT is able to authenticate the expired tokens, and the time required by the server to generate tokens is varied influenced by the response server to the publisher request vary.

Keywords: *IoT, Message Queuing Telemetry Transport (MQTT), JSON Web Token (JWT), NodeMCU*

1. PENDAHULUAN

IoT merupakan mekanisme komunikasi *machine-to-machine* (M2M) yang diharapkan menjadi salah satu solusi jaringan masa depan. Jumlah perangkat IoT diperkirakan tumbuh pesat, hingga 50 miliar perangkat pada tahun 2020 (Niruntasukrat dkk., 2016). Kebanyakan *node* sensor yang digunakan untuk mengumpulkan data, ditempatkan di tempat dimana koneksi internet terbatas atau lambat. Pada lingkungan seperti ini, *packet loss* yang tinggi menjadi kelemahan utama. MQTT adalah protokol yang berbasis *publish-subscribe* untuk komunikasi M2M/IoT yang berjalan diatas protokol TCP/IP untuk data kecil (misalnya 8-bit, 256KB RAM *controller*), *bandwidth* dan daya rendah, ketersediaan variabel, dan jaminan pengiriman (Špeh & Heđ, 2016). Hal yang sama tentang alasan mengapa memilih protokol MQTT adalah karena protokol MQTT cocok digunakan pada lingkungan dimana *bandwidth* yang tersedia terbatas (Chooruang & Mangkalakeeree, 2016). Penelitian sebelumnya menggambarkan cara kerja dari IoT adalah dimana perangkat sensor mengirimkan data berupa ke *data center* (*central server*) melalui mekanisme *publish/subscribe* dengan menggunakan protokol MQTT (Špeh & Heđ, 2016). *Data center* berperan sebagai broker untuk melakukan *multicast* ke *subscriber*. Selanjutnya, data dari *node* akan dikirimkan ke web *server* dan SQL untuk disimpan.

Untuk melakukan verifikasi, MQTT menggunakan *username* dan *password* pada setiap *request* yang dikirimkan untuk proses autentikasi. Penggunaan *username* dan *password* untuk proses autentikasi sangat rentan terhadap penyadapan. *Database* yang digunakan untuk menyimpan *username* dan *password* rentan untuk diretas misalnya dengan menggunakan *SQL Injection*. Menambahkan mekanisme *hash* pada *password* mungkin menjadi solusi yang ampuh, namun akan berdampak pada performansi sistem dan juga berdampak besar pada memori penyimpanan. Kelemahan lain dari penggunaan *username* dan *password* adalah tidak adanya *expiration*, ketika *username* dan *password* dicuri sangat sulit untuk mengetahui terkecuali jika seorang *attacker* secara langsung melakukan pengubahan atau perusakan terhadap data yang dikirim.

JSON Web *Token* (JWT) menjadi solusi menggantikan *username* dan *password*. JWT adalah token berbentuk string panjang random yang gunanya untuk melakukan autentikasi sistem dan pertukaran informasi (Salma, 2017). JWT mengamankan informasi menjadi sebuah klaim yang di encode ke dalam bentuk JSON dan menjadi payload dari JSON Web Signature (JWS) (Bradley, 2015). Alasan penggunaan autentikasi JWT yang berbasis *token* adalah adanya *expiration*. Ukuran JWT yang kecil memungkinkannya dapat dikirimkan melalui URL, parameter HTTP POST, atau *header* HTTP POST. *Token* disimpan disisi klien sebagai *cookies* sehingga *server* tidak harus meyimpan *token*. Karena *token* sudah ditandatangani secara digital maka dapat diketahui siapa yang melakukan *request*, karena isi dari tanda tangan pada JWT merupakan gabungan dari isi *header* dan *payload*, jika terjadi perubahan maka *signature* akan menjadi tidak *valid* (www.jwt.io, diakses 02-02-2017).

Permasalahan selanjutnya yang muncul adalah bagaimana melakukan manajemen *token* pada perangkat seperti NodeMCU yang mempunyai memori kecil. Memori *flash* pada NodeMCU adalah 4MB dan ini akan dipakai untuk firmware, menjalankan program, dan juga digunakan sebagai penyimpanan. NodeMCU tidak dapat menyimpan *file* dengan ukuran yang besar dan juga tidak dapat menjalankan program yang sangat kompleks. Untuk manajemen *token*, NodeMCU akan menyimpan *token* sebagai hasil autentikasi, dan menggunakan *token* untuk melakukan komunikasi selanjutnya. Ketika *token* telah *expired*, NodeMCU akan mengirimkan *request token* kembali ke *server*. Karena kelebihan autentikasi berbasis *token* adalah ukuran *token* yang kecil yang memungkinkan dapat disimpan sebagai hasil autentikasi pada perangkat sensor yang mempunyai memori kecil.

Berdasarkan uraian diatas maka dibuatlah suatu penelitian untuk mengimplementasikan autentikasi berbasis *token* atau JWT pada protokol MQTT yang terdiri dari tiga komponen yaitu NodeMCU sebagai *publisher*, broker, dan JWT *auth server*. *Publisher* akan melakukan *request token* dengan mengirimkan *username* dan *password* ke *server* dan menggunakan *token* untuk autentikasi ketika melakukan *publish* ke broker. *Token* akan disimpan disisi *publisher* dan akan dilakukan *request token* kembali ketika *token* telah *expired*.

2. LANDASAN KEPUSTAKAAN

2.1. Kajian Pustaka

Referensi pertama terkait dengan mekanisme keamanan pada protokol IoT khususnya MQTT dengan judul "*Authorization Mechanism for MQTT-based Internet of Things*". Dalam penelitian tersebut dijelaskan bahwa salah seorang yang mengusulkan untuk mekanisme autorisasi pada protokol MQTT adalah dengan memodifikasi *OAuth1.0a* untuk mengakomodasi pembatasan perangkat IoT. Beberapa *hardware* dengan *platform* yang kecil mungkin tidak mampu membawa protokol berbasis enkripsi-tradisional seperti TSL/SSL. Selama autorisasi, satu set kredensial (*Access Token* dan *Access Token Secret*) melewati saluran terenkripsi antara perangkat dan titik akhir otorisasi dapat dicuri. Namun dengan mekanisme autorisasi ini membutuhkan dua set kredensial untuk perangkat melakukan akses ke broker MQTT. Melalui percobaan dan pelayanan yang nyata, proses otorisasi yang diusulkan terbukti bekerja sebagaimana dimaksud, dan *overhead* yang dikeluarkan tidak mempengaruhi pengalaman pengguna (Niruntasukrat dkk., 2016).

Referensi kedua tentang penerapan JWT dengan judul "*Personal Cloudlets : Implementing a User-Centric Datastore with Privacy Aware Access Control for Cloud-based Data Platforms*" yang menerapkan autentikasi JWT dengan melihat kredensial dan *secret key* yang dimasukkan oleh pengguna. Jika kredensial dan *secret key* yang dimasukkan oleh pengguna valid, maka akan sistem menghasilkan JWT yang berisi data pengguna dan data aplikasi. JWT nantinya akan disimpan di klien dan digunakan untuk autentikasi yang lain (McCarthy dkk., 2015). Penelitian selanjutnya terkait implementasi dari JWT yang dikombinasikan dengan *blockchain* dilakukan Jose G. Faisca, pada penelitian tersebut JWT diimplementasikan sebagai metode autentikasi untuk *endpoint* pada *personal cloud*. Dari penelitian tersebut dijelaskan bahwa mekanisme autentikasi JWT yang diterapkan pada *personal cloud* digunakan untuk menyediakan interopabilitas pada *personal cloud*. Karateristik dari *blackchain* dimana karakter dari *blackchain* sendiri yang terdistribusi, tidak beraturan, tidak berubah dan tidak dapat diubah memiliki atribut yang sesuai untuk menyimpan suatu kredensial terdistribusi dan manajemen identitas terdesentralisasi (Faisca & Rogado, 2016).

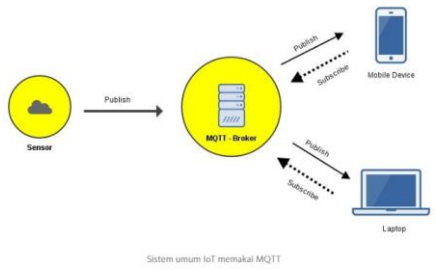
Referensi ketiga adalah penelitian yang dilakukan oleh Andrej Skraba dkk., mengenai NodeMCU yang berjudul "*Streaming Pulse Data to the Cloud with Bluetooth LE or* NODEMCU *ESP8266*" yang menggunakan NodeMCU dengan ditambahkan sensor dan mengirimkan data dari sensor ke *cloud*. Pengiriman data ke *cloud* menggunakan wifi yang sudah terintegrasi pada NodeMCU, sehingga hanya perlu menyesuaikan koneksi agar NodeMCU terhubung ke internet (Skraba dkk., 2016).

2.2. Message Queuing Telemetry Transport (MQTT)

Message Queuing Telemetry Transport adalah protokol yang digunakan untuk komunikasi pada IoT (Internet of Thing) berbasis broker *publish*/*subscribe* yang dirancang terbuka, sederhana, ringan, dan mudah diimplementasikan. Karakteristik ini membuat MQTT ideal untuk digunakan dilingkungan yang dibatasi tetapi tidak terbatas pada:

1. Dimana jaringan mahal, memiliki *bandwidth* rendah atau tidak dapat diandalkan
2. Ketika berjalan di perangkat *embedded* dengan sumber daya prosesor dan memori terbatas (Eurotech, 1999-2010).

Protokol *Message Queuing Telemety Transport* (MQTT) adalah protokol yang berjalan diatas *stack* TCP/IP dan mempunyai ukuran paket data dengan *low overhead* yang kecil (minimum 2 bytes) sehingga berefek pada konsumsi catu daya yang cukup kecil. Dengan protokol ini data yang dapat dikirimkan beragam seperti data binary, teks, bahkan xml ataupun JSON dan protokol ini memakai model *publish*/*subscribe* dari pada model *client-server*.

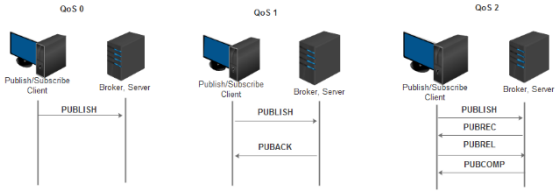


Gambar 2.1. Sistem Umum IoT memakai MQTT (Pr., 2015)

Keuntungan dari penggunaan *publish/subscribe* adalah antara klien dan sumber tidak perlu mengetahui satu sama lain karena ada broker diantara mereka atau yang biasa dikenal dengan *space decoupling*, dan yang paling penting dengan komunikasi MQTT adalah adanya *time decoupling* yaitu dimana antara klien dan sumber tidak perlu saling terkoneksi secara bersamaan, misalnya klien bisa saja *disconnect* setelah melakukan *subscribe* ke broker dan beberapa saat kemudian klien *connect* kembali, maka klien akan tetap menerima data yang terpending.

QoS merupakan kemampuan suatu jaringan dalam menyediakan layanan. Kinerja jaringan komputer dapat bervariasi akibat beberapa masalah, seperti halnya *bandwidth*, *latency*, dan *jitter* yang dapat membuat efek yang cukup besar bagi banyak aplikasi. Dengan kata lain, QoS adalah kemampuan suatu jaringan untuk menyediakan layanan yang baik dengan menyediakan *bandwidth*, mengatasi *jitter* dan *delay*.

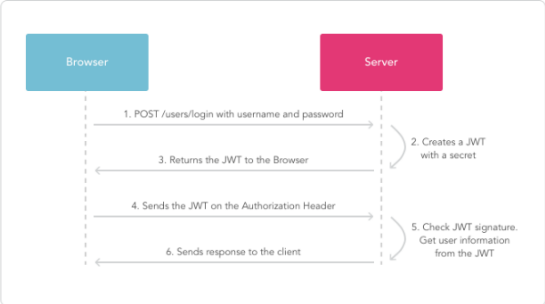
Untuk menjamin QoS, MQTT mendukung 3 tingkat *Quality of Services*. QoS level 0 mengirimkan pesan hanya sekali mengikuti aliran distribusi pesan dan tidak memeriksa apakah pesan sampai ke tujuan. QoS level 1 mengirimkan pesan setidaknya sekali, dan memeriksa status pengiriman pesan pesan dengan menggunakan cek status pesan, PUBACK. QoS level 2 melewatkan pesan yang tepat sekali dengan memanfaatkan *4-way handshake*. Hal ini tidak memungkinkan terjadi *loss* pesan di level ini, tapi karena proses yang rumit dari *4-way handshake*, mungkin memiliki *end-to-end delay* relatif lama.



Gambar 2.2. Metode Transmisi Paket Pada Level QoS (Lee, Kim, Hong, & Ju, 2013)

2.3. JSON Web *Token* (JWT)

JWT adalah 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 akan dapat dilindungi dengan tanda tangan digital seperti *Message authentication code* (MAC) atau dienkripsi (Bradley, 2015).



Gambar 2.3. JSON Web *Token* Proses (www.jwt.io, diakses 02-02-2017)

Berikut adalah karakteristik dari *JSON Web Token* (JWT):

1. *Header*

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

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

Gambar 2.4. JWT *Header* (www.jwt.io, diakses 02-02-2017)

2. *Payload*

   Bagian kedua dari *token* adalah *payload*, yang berisi klaim. Klaim adalah pernyataan tentang suatu entitas (biasanya, pengguna) dan metadata tambahan. Ada tiga jenis klaim: *reserved*, *public*, dan *private*.

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

Gambar 2.5. JWT *Payload* (www.jwt.io, diakses 02-02-2017)

3. *Signature*

   Untuk membuat bagian tanda tangan kita harus mengambil *header* yang dikodekan, *payload* yang dikodekan, *secret*, algoritma yang ditentukan dalam *header*, dan menandatangani itu.

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

Gambar 2.6. JWT Signature (www.jwt.io, diakses 02-02-

2017)

Signature digunakan untuk memverifikasi pengirim bahwa ia adalah pengirim yang benar dan untuk memastikan bahwa pesan yang dikirim tidak diubah (Pr., 2015)

2.4. NodeMCU

NodeMCU adalah firmware berbasis LUA untuk ESP8266 WiFi SOC. Model pemrograman pada NodeMCU mirip dengan *Node*.js namun di LUA. Pada pemrograman LUA juga terdapat parameter untuk fungsi callback. Ini seperti *asynchronous* dan *event-driven* (Charoenporn, 2016). ESP8266 merupakan modul wifi yang berfungsi sebagai perangkat tambahan mikrokontroler seperti Arduino agar dapat terhubung langsung dengan wifi dan membuat koneksi TCP/IP.

Gambar 2.7. NodeMCU (www.seeedstudio.com, diakses 02-04-2017)

Modul ini membutuhkan daya sekitar 3.3v dengan memiliki tiga mode wifi yaitu *Station*, *Access Point* dan *Both* (Keduanya).

Firmware default yang digunakan oleh perangkat ini menggunakan *AT Command*, selain itu ada beberapa *Firmware* SDK yang digunakan oleh perangkat ini berbasis *opensource* yang diantaranya adalah sebagai berikut:

- NodeMCU dengan menggunakan *basic programming* lua
- MicroPython dengan menggunakan *basic programming* python
- *AT Command* dengan menggunakan perintah perintah *AT command*

Untuk pemrogramannya sendiri kita bisa menggunakan ESPlorer untuk *Firmware* berbasis NodeMCU dan menggunakan putty sebagai terminal control untuk *AT Command* (TRESNA WIDIYAMAN, 2016).

3. PENGUJIAN

Pengujian dalam penelitian ini terdiri dari tiga bagian inti yang dijabarkan menjadi 3 macam pengujian diantaranya pengujian fungsionalitas, pengujian keamanan/*security*, pengujian performansi. Pengujian fungsionalitas adalah pengujian yang dilakukan untuk melihat apakah kebutuhan fungsional dari sistem sudah terpenuhi dan dapat berjalan dengan baik. Pengujian ini dilakukan disesuaikan dengan hal-hal yang telah didefinisikan dalam kebutuhan fungsional. Hasil dari pengujian ini merupakan proses-proses yang harus dijalankan dan dipenuhi oleh sistem..

Pengujian keamanan/*security* adalah pengujian yang dilakukan untuk melihat apakah *server* JWT mampu melakukan validasi dan autentikasi terhadap *username* dan *password* jika *password* yang dikirimkan oleh *publisher* merupakan *username* dan *password* yang salah. Selain itu, dalam pengujian keamanan ini akan dilihat apakah JWT juga mampu melakukan autentikasi dan validasi terhadap *token* ketika *token* belum *expired* ataupun ketika *token* sudah *expired*. Pengujian keamanan dibagi menjadi tiga skenario yaitu pengujian validasi *username* dan *password invalid*, pengujian *expiration token* tanpa *disconnect*, dan pengujian *expiration token* dengan *disconnect*.

Pengujian validasi *username* dan *password invalid* adalah pengujian yang dilakukan untuk mengetahui apakah *server* mampu melakukan validasi pada *publisher* jika *username* dan *password* yang dikirimkan tidak valid. Skenario dari pengujian ini adalah *publisher* akan mengirimkan *request token* ke *server* dengan *username* dan *password* yang tidak valid. *Server* akan melakukan autentikasi, dan menampilkan pesan jika *username* dan *password* tidak valid.

Pengujian *expiration token* tanpa *disconnect* adalah pengujian yang dilakukan untuk melihat apakah *server* dapat melakukan autentikasi terhadap *token* yang telah *expired* atau masa berlakunya telah habis. Skenario dari pengujian ini adalah ketika *publisher* telah mendapatkan *token*, maka *token* tersebut digunakan untuk melakukan komunikasi selanjutnya atau *publish* ke broker. Ketika *token* telah *expired*, maka *server* akan melakukan autentikasi dan menampilkan pesan *error*.

Pengujian *expiration token* dengan *disconnect* adalah pengujian yang sama yang dilakukan yaitu untuk menguji apakah *server* mampu melakukan autentikasi terhadap *token* yang telah *expired*, hanya saja terdapat penambahan kondisi dimana *publisher* akan berhenti melakukan *publish* sementara ketika diketahui bahwa *token* yang digunakan telah *expired*. Selanjutnya, *publisher* akan melakukan *connect* kembali ke broker dan melakukan *publish* dengan menggunakan *token* yang sama dengan *token* yang digunakan sebelumnya.

Pengujian performansi adalah pencatatan waktu yang dilakukan di sisi *publisher* yang dalam penelitian ini menggunakan NodeMCU. Skenario pengujian waktu ini adalah *publisher* akan mengirimkan *request* berupa *username* dan *password* ke *server* sebanyak 50 kali tanpa melakukan *publish*. Pencatatan waktu dilakukan ketika *publisher* mengirimkan *request* dan *server* mengirimkan *response* terhadap *request* berupa *token*.

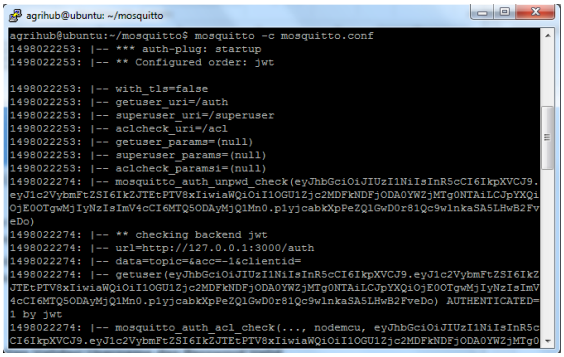
4. HASIL DAN PEMBAHASAN

4.1. Pengujian Fungsionalitas

Pengujian fungsionalitas merupakan pengujian yang dilakukan untuk melihat apakah kebutuhan fungsional dari sistem sudah dapat berjalan dengan baik.

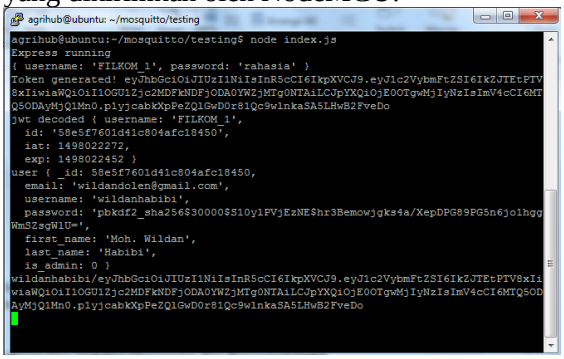
Gambar 4.1. Komunikasi MQTT NodeMCU

Dari gambar 4.1 dapat dilihat bahwa NodeMCU melakukan publish topik ke broker dengan menggunakan protokol MQTT.



Gambar 4.2. Autentikasi Token oleh Broker

Dari gambar 4.2 dapat dilihat bahwa broker dapat melakukan autentikasi terhadap token yang dikirimkan oleh NodeMCU.



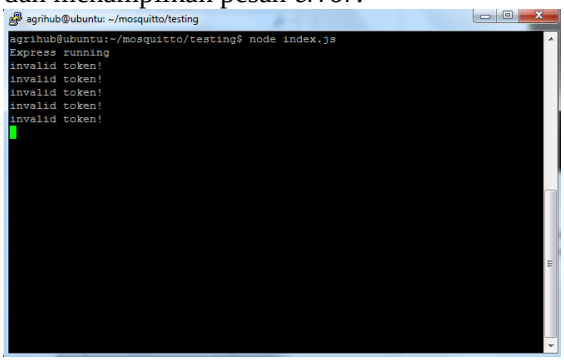
Gambar 4.3. Autentikasi di Server

Dari gambar 4.3 dapat dilihat bahwa *server* dapat melakukan autentikasi pada *token*.

4.2. Pengujian Keamanan

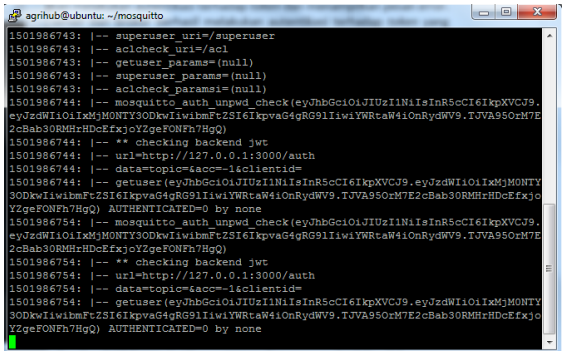
4.2.1 Pengujian Validasi Token Invalid

Skenario yang digunakan dalam pengujian ini adalah mengirimkan *token* yang tidak valid sehingga *server* akan melakukan autentikasi dan menampilkan pesan *error*.



Gambar 4.4. Hasil Pengujian Autentikasi

Dari gambar 4.4 didapatkan hasil bahwa *server* dapat melakukan autentikasi terhadap *token invalid* yang dikirimkan oleh *publisher* dan menampilkan pesan *error* berupa "*invalid token*".

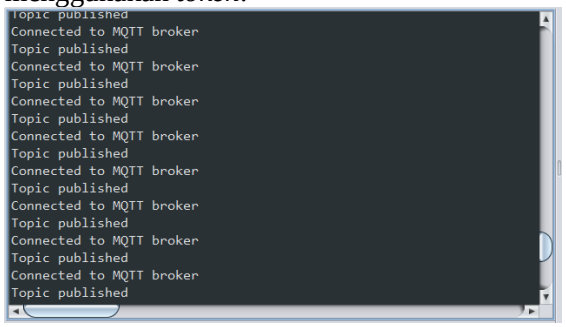


Gambar 4.5. Hasil Pengujian Autentikasi Disisi Broker

Disisi *publisher* ditampilkan pesan *error* berupa "*Authenticated=0 by none*" ketika *token* yang dikirimkan tidak valid.

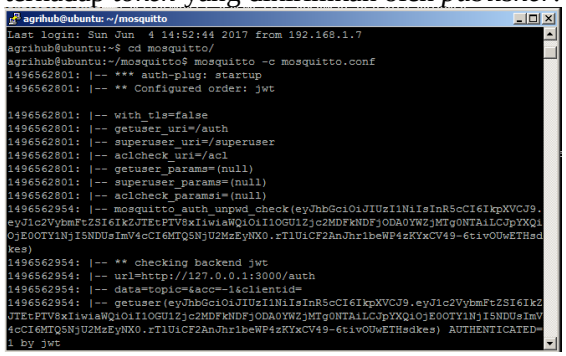
4.2.2 Pengujian *Expiration Token* Tanpa *Disconnect*

Setelah *publisher* melakukan ke broker dengan menggunakan token, selanjutnya *publisher* melakukan *publish* ke broker dengan menggunakan *token*.



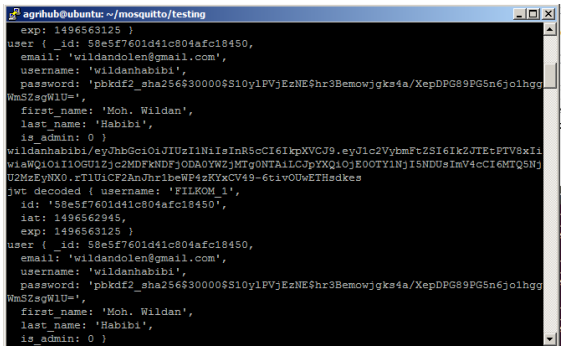
Gambar 4.6. Publisher Melakukan Publish

Disisi Broker dilakukan autentikasi terhadap *token* yang dikirimkan oleh *publisher*.



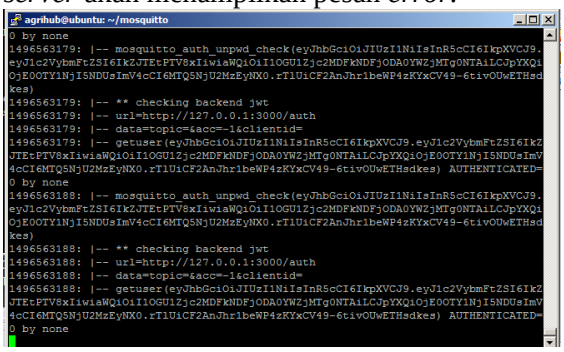
Gambar 4.7. Autentikasi Disisi Broker

Ketika *token* yang digunakan oleh *publisher* masih valid atau belum *expired*, maka broker akan menampilkan pesan bahwa *token* tersebut teautentikasi. Setelah autentikasi disisi broker selanjutnya autentikasi disisi *server*.

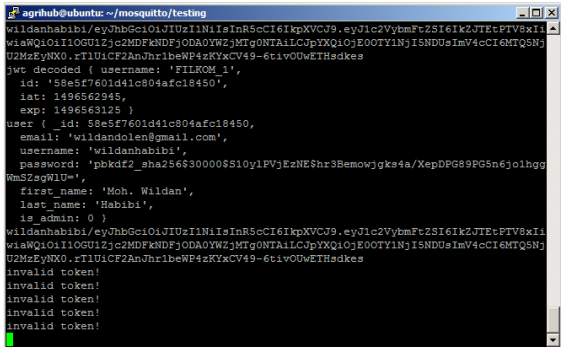


Gambar 4.8. Autentikasi Disisi Server

Mekanisme *publish* dengan *token* berlangsung hingga *token* yang digunakan telah *expired*. Ketika *token expired* maka broker dan *server* akan menampilkan pesan *error*.



Gambar 4.9. Autentikasi Token yang Expired Broker

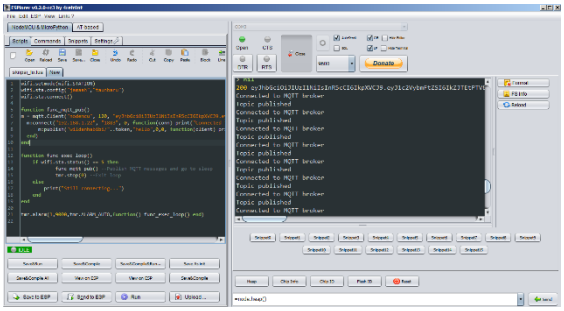


Gambar 4.10. Autentikasi Token yang Expired Server

Dari gambar 4.9 dan 4.10 diperoleh hasil bahwa broker dan *server* dapat melakukan autentikasi terhadap *token* ketika masa berlaku *token* tersebut telah habis atau *expired*.

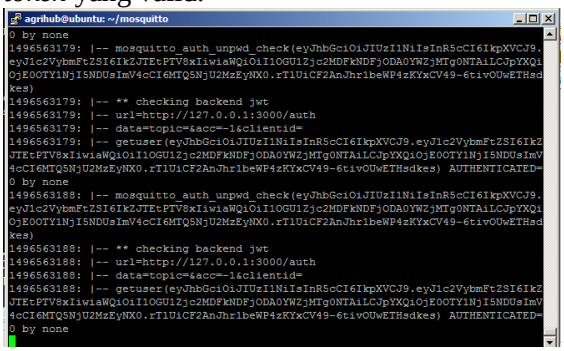
4.2.3 Pengujian *Expiration Token* Dengan *Disconnect*

Skenario yang dilakukan dalam pengujian ini sama dengan skenario pengujian sebelumnya hanya saja terdapat penambahan kondisi dimana ketika *publisher* sedang melakukan *publish* topik dan *token* yang digunakan telah *expired*, *publisher* akan melakukan *disconnect* dan berhenti melakukan *publish* sementara.

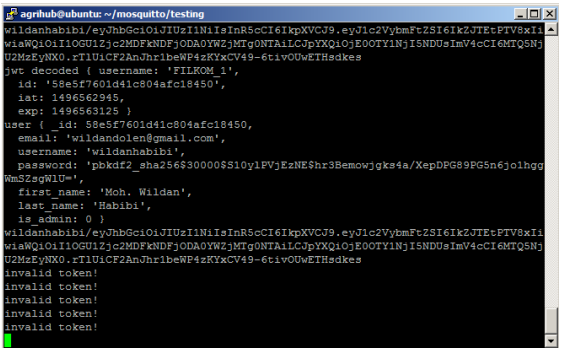


Gambar 4.11. Publisher Melakukan Publish

Gambar 4.11 menunjukkan bahwa *publisher* melakukan *publish* topik dengan *token* yang valid.

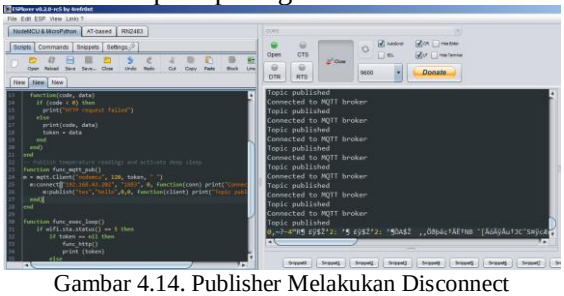


Gambar 4.12. Token Expired Disisi Broker



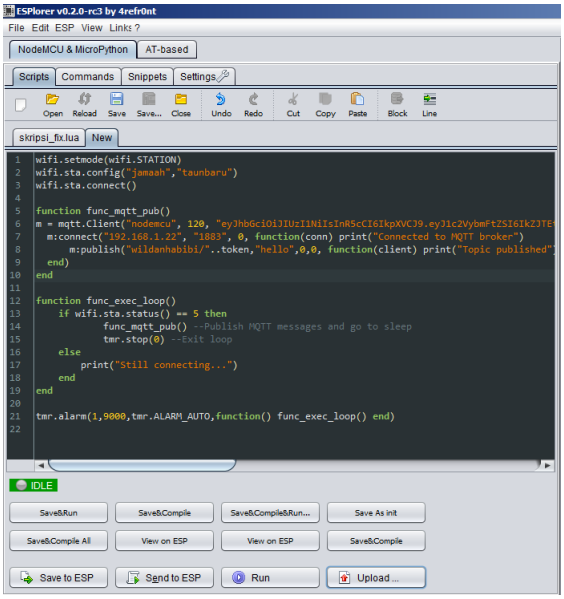
Gambar 4.13. Token Expired Disisi Server

Gambar 4.12 dan 4.13 menunjukkan bahwa *token* yang digunakan telah *expired* dan sudah tidak terautentikasi disisi broker dan *server*. Selanjutnya *publisher* melakukan *disconnect* sementara seperti pada gambar 4.14.



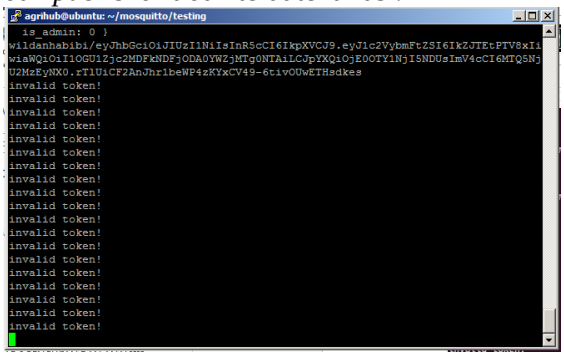
Gambar 4.14. Publisher Melakukan Disconnect

Setelah beberapa saat, *publisher* melakukan *connect* kembali dengan menggunakan *token* yang telah *expired*.

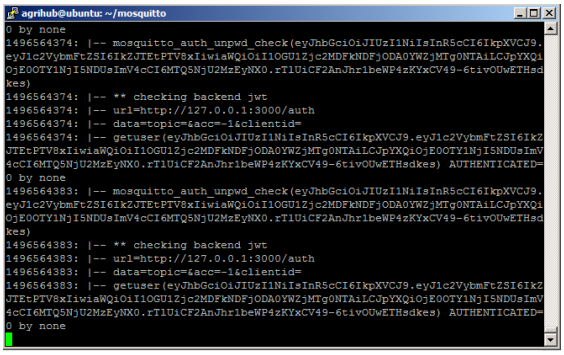


Gambar 4.15. Publisher Melakukan Connect

Ketika *publisher* *connect* dengan menggunakan *token* yang telah *expired* broker dan *server* melakukan autentikasi dan menampilkan hasil bahwa *token* telah *expired* dan *publisher* tidak terautentikasi.



Gambar 4.16. Token Expired Disisi Broker



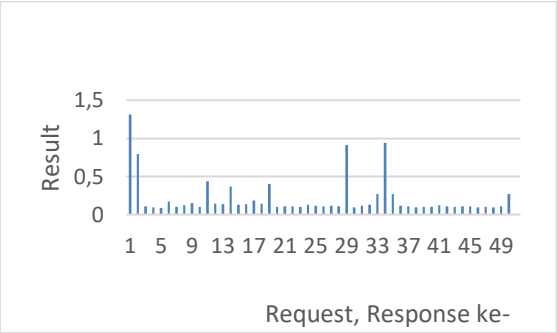
Gambar 4.17. Token Expired Disisi Broker

Gambar 4.16 dan 4.17 menunjukkan bahwa *token* yang digunakan adalah *token* yang telah expired.

4.3. Pengujian Performansi

Pencatatan waktu dilakukan disisi *publisher*, tidak dilakukan disisi *server* dan

broker. Pengujian performansi merupakan pengujian yang dilakukan untuk melihat kinerja dari *server*, berapa lama waktu yang dibutuhkan oleh *server* untuk men-*generate token* untuk *publisher* mulai dari awal *request* dikirimkan hingga *token* sampai di ter-*generate*.



Gambar 4.18. Grafik Hasil Pengujian Waktu

Dari grafik hasil pengujian waktu *generate token* yang dilakukan diperoleh hasil bahwa waktu yang dibutuhkan oleh *server* untuk *generate token* dan mengirimkan *token* kembali ke NodeMCU adalah bervariasi atau tidak konstan. Hal ini disebabkan oleh waktu yang dibutuhkan oleh *server* untuk memberikan *response* terhadap setiap *request* yang dilakukan oleh *publisher* berbeda-beda sehingga *request* yang dikirimkan membutuhkan waktu yang bervariasi untuk sampai ke *server* atau *response* yang dikirimkan oleh *server* ke NodeMCU membutuhkan waktu yang bervariasi. Selain itu, kondisi jaringan juga mempengaruhi waktu yang dibutuhkan *request* untuk sampai ke *server* dan *token* untuk sampai ke *publisher*.

5. KESIMPULAN

Hasil pengujian dari implementasi autentikasi JWT dengan menggunakan NodeMCU dengan 3 pengujian yang dilakukan diperoleh hasil bahwa *server* yang diimplementasikan dengan menggunakan *framework Node*.js dapat melakukan autentikasi terhadap *token* yang dikirimkan oleh *publisher*.

Ketika *publisher token* yang tidak valid, *server* dan broker dapat meng-autentikasi. Ketika *token* yang digunakan telah *expired*, broker dan *server* berhasil melakukan autentikasi dan menampilkan pesan *error*. Ketika hal ini terjadi maka *publisher* harus melakukan *request* ulang *token*.

6. DAFTAR PUSTAKA

Niruntasukrat, A., Issariyapat, C., Pongpaibool, P., Meesublak, K., Aiumsupucgul, P., & Panya, A. (2016). Authorization mechanism for MQTT-based Internet of Things. 2016 IEEE International Conference on Communications Workshops, ICC 2016, 6, 290–295. https://doi.org/10.1109/ICCW.2016.7503802.

Špeh, I., & Heđ, I. (2016). A Web - Based IoT Solution for Monitoring Data Using MQTT Protocol, 249–253. https://doi.org/10.1109/SST.2016.7765668.

Chooruang, K., & Mangkalakeeree, P. (2016). Wireless Heart Rate Monitoring System Using MQTT. Procedia Computer Science, 86(March), 160–163. https://doi.org/10.1016/j.procs.2016.05.045.

Salma, A. I. (n.d.). Mengenal Konsep JSON Web Token (JWT). Retrieved August 4, 2017, from https://www.dumetschool.com/blog/mengenal-konsep-json-web-token-jwt.

Bradley, J. (2015). JSON Web *Token* (JWT), 1–30.

Anonymous. (n.d.). JWT. Retrieved February 2, 2017, from https://jwt.io/introduction/

McCarthy, D., Malone, P., Hange, J., Doyle, K., Robson, E., Conway, D., … Lampathaki, F. (2015). Personal Cloudlets: Implementing a User-centric Datastore with Privacy Aware Access Control for Cloud-Based Data Platforms. Proceedings - 1st International Workshop on TEchnicaland LEgal Aspects of Data pRIvacy and Security, TELERISE 2015, 38–43. https://doi.org/10.1109/TELERISE.2015.15

Faisca, J. G., & Rogado, J. Q. (2016). Personal *cloud* interoperability. WoWMoM 2016 - 17th International Symposium on a World of Wireless, Mobile and Multimedia Networks. https://doi.org/10.1109/WoWMoM.2016.7523546

Lee, S., Kim, H., Hong, D. K., & Ju, H. (2013). Correlation analysis of MQTT loss and delay according to QoS level. International Conference on Information Networking, 714–717. https://doi.org/10.1109/ICOIN.2013.6496715

Skraba, A., Kolozvari, A., Kofjac, D., Stojanovic, R., Stanovov, V., & Semenkin, E. (2016). Streaming pulse data to the *cloud* with bluetooth le or NODEMCU ESP8266. 2016 5th Mediterranean Conference on Embedded Computing, MECO 2016 - Including ECyPS 2016, BIOENG.MED 2016, MECO: Student Challenge 2016, 428–431. https://doi.org/10.1109/MECO.2016.7525798

Pr., E. (2015). Mengenal MQTT. Retrieved February 4, 2017, from https://jsiot.pw/mengenal-mqtt-998b6271f585

Charoenporn, T. (2016). Selection Model for Communication Performance of the Bus Tracking System.

www.seeedstudio.com. (n.d.). Nodemcu. Retrieved April 2, 2017, from https://www.seeedstudio.com/NodeMCU-v2-Lua-based-ESP8266-development-kit-p-2415.html

TRESNA WIDIYAMAN. (2016). Pengertian Modul Wifi ESP8266. Retrieved March 3, 2017, from http://www.warriornux.com/pengertian-modul-wifi-esp8266/