**한성시스코**

**TLS/SSL 보안을 적용해**

**MQTT 브로커와 WebSocket 통신**

EoT기술연구소

인턴 최수인



**목 차**

**개요** [3](#Head)

**Npm으로 빠르게 인증서 만들기**[**4**](#Node)

1. 프로젝트 생성[4](#nodeOne)
2. OpenSSL설치[5](#nodeTwo)
3. 프로젝트에 모듈 설치[5](#nodeThree)
4. keygen.ts파일 추가[6](#nodeFour)
5. .babelrc 파일 생성[9](#nodeFive)
6. script 추가[9](#nodeSix)
7. 폴더와 package.json 확인[10](#nodeSeven)
8. 사용법[11](#nodeEight)

**웹서버 구축과 ssl로 mqtt 브로커와 통신하기** [**12**](#Web)

1. MQTT 브로커 configure 설정(mosquitto사용)[12](#webOne)
2. 프로젝트 생성[13](#nodeTwo)
3. 필요한 모듈 설치[13](#webThree)
4. 진행확인[14](#webFour)
5. 웹에서 MQTT와 ssl로 통신하기[14](#webFive)

**개 요**

앞서 이 문서는 node.js를 이용해 웹페이지를 열고 OpenSSL을 사용해 자체 서명한 보안서를 이용해 mqtt 브로커와 websocket으로 통신하는 방식을 설명합니다. Mqtt 브로커는 mosquitto를 사용했습니다. 먼저 아래의 링크로 들어가 mosquitto와 node.js를 설치해주세요.

Mosquitto Broker : <https://mosquitto.org/download/>

Node.js : <https://nodejs.org/ko/>

만약 npm방식이 아닌 OpenSSL에 직접 명령어로 인증서를 만들어 쓰신다면 12p의 [‘웹서버 구축과 ssl로 mqtt 브로커와 통신하기’](#Web) 부터 시작하시면 됩니다. 이 때, 코드 작성 중 인증서들의 파일위치에 주의해 주시기 바랍니다.

cmd 창에 입력하는 명령어들은 아래의 모습과 같습니다

명령어는 회색바탕 입니다

코드의 내용은 아래와 같습니다.

파일명

--------------------------------------------------------------------------------------------------------------

코드내용은 이런 형식으로 적었습니다. 붙여넣기 시 “와 페이지 번호를 주의해 주시기 바랍니다.

--------------------------------------------------------------------------------------------------------------

**Npm으로 빠르게 인증서 만들기**

출처 : <https://salgum1114.github.io/etc/2019-05-20-using-openssl-in-windows/>

OpenSSL을 사용하면 긴 명령어들을 반복해서 실행해야 합니다. 밑의 방법을 실행하면 짧은 명령어 하나만으로 인증서를 만들 수 있습니다.

1. **프로젝트 생성**

원하는 위치에 프로젝트 폴더를 생성 후 프로젝트 안에

package.json 파일을 생성합니다.

package.json

----------------------------------------------------------------------------------------------------

{

"name" : "mqtt-websocket-over-ssl",

"version" : "0.0.1",

"scripts" : {},

"dependencies" : {},

"author" : "Suin Choi"

}

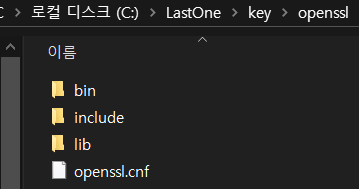
----------------------------------------------------------------------------------------------------

1. **OpenSSL 설치**

프로젝트 폴더에 key폴더를 생성합니다.

OpenSSL : <https://code.google.com/archive/p/openssl-for-windows/downloads>

위의 페이지에서 파일을 다운받고 압축해제후 key폴더에 넣습니다. 이때 폴더명은 openssl로 바꿔주세요.



1. **프로젝트에 모듈 설치**

프로젝트 폴더에 아래 모듈들을 설치합니다.

npm install --save-dev @babel/core @babel/node

npm install --save-dev @babel/plugin-proposal-class-properties

npm install --save-dev @babel/plugin-proposal-object-rest-spread

npm install --save-dev @babel/preset-env

npm install --save-dev @babel/preset-typescript

1. **Keygen.ts 파일 추가**

프로젝트 폴더에 keystore 폴더를 생성합니다. keystore 폴더 안에 ca, server, client 폴더를 생성합니다. keygen 실행 시 각각의 폴더로 인증서가 생성됩니다.

key폴더에 keygen.ts파일을 생성합니다.

keygen.ts

----------------------------------------------------------------------------------------------------

import { execSync, ExecSyncOptions } from 'child\_process';

import path from 'path';

type Status = 'all' | 'ca' | 'server' | 'client' | 'verify';

const argsLength = process.argv.length;

let status: Status = 'all';

if (argsLength > 2) {

status = process.argv.slice(2)[0] as Status;

}

const opensslPath = 'openssl/bin/openssl';

const opensslConfPath = `openssl/openssl.cnf`;

const binPath = path.resolve(\_\_dirname, opensslPath)

const confPath = path.resolve(\_\_dirname, opensslConfPath);

const execSyncOption: ExecSyncOptions = {

stdio: 'inherit',

}

/\*\*

\* Create CA

\*/

const caKeystorePath = 'keystore/ca';

const caKeyPath = `${caKeystorePath}/ca.key`;

const caCrtPath = `${caKeystorePath}/ca.crt`;

if (status === 'all' || status === 'ca') {

execSync(`${binPath} req -new -x509 -days 1024 -extensions v3\_ca -keyout ${caKeyPath} -out ${caCrtPath} -config ${confPath}`, execSyncOption);

}

if (status === 'verify') {

execSync(`${binPath} x509 -text -in ${caCrtPath}`, execSyncOption) ;

}

/\*\*

\* Create server key

\*/

const serverKeystorePath = 'keystore/server';

const serverKeyPath = `${serverKeystorePath}/server.key`;

const serverCsrPath = `${serverKeystorePath}/server.csr`;

const serverCrtPath = `${serverKeystorePath}/server.crt`;

if (status === 'all' || status === 'server') {

// execSync(`${binPath} genrsa -des3 -out ${serverKeyPath} 2048`, execSyncOption);

execSync(`${binPath} genrsa -out ${serverKeyPath} 2048`, execSyncOption);

execSync(`${binPath} req -out ${serverCsrPath} -key ${serverKeyPath} -new -config ${confPath}`, execSyncOption);

execSync(`${binPath} x509 -req -in ${serverCsrPath} -CA ${caCrtPath} -CAkey ${caKeyPath} -CAcreateserial -out ${serverCrtPath} -days 1024`, execSyncOption);

}

/\*\*

\* Create client key

\*/

const clientKeystorePath = 'keystore/client';

const clientKeyPath = `${clientKeystorePath}/client.key`;

const clientCsrPath = `${clientKeystorePath}/client.csr`;

const clientCrtPath = `${clientKeystorePath}/client.crt`;

if (status === 'all' || status === 'client') {

// execSync(`${binPath} genrsa -des3 -out ${clientKeyPath} 2048`, execSyncOption);

execSync(`${binPath} genrsa -out ${clientKeyPath} 2048`, execSyncOption);

execSync(`${binPath} req -out ${clientCsrPath} -key ${clientKeyPath} -new -config ${confPath}`, execSyncOption);

execSync(`${binPath} x509 -req -in ${clientCsrPath} -CA ${caCrtPath} -CAkey ${caKeyPath} -CAcreateserial -out ${clientCrtPath} -days 1024`, execSyncOption);

}

----------------------------------------------------------------------------------------------------

1. **.babelrc 파일 생성**

프로젝트 폴더에 .babelrc 파일을 만듭니다.

.babelrc

----------------------------------------------------------------------------------------------------

{

"presets": [

"@babel/preset-env",

"@babel/preset-typescript"

],

"plugins": [

"@babel/proposal-class-properties",

"@babel/proposal-object-rest-spread"

]

}

----------------------------------------------------------------------------------------------------

1. **Script 추가**

package.json 의 "scripts" : {} 안에 밑의 내용을 추가합니다.

package.json

----------------------------------------------------------------------------------------------------

"keygen": "babel-node --extensions \".ts\" key/keygen.ts",

"keygen:ca": "babel-node --extensions \".ts\" key/keygen.ts ca",

"keygen:server": "babel-node --extensions \".ts\" key/keygen.ts server",

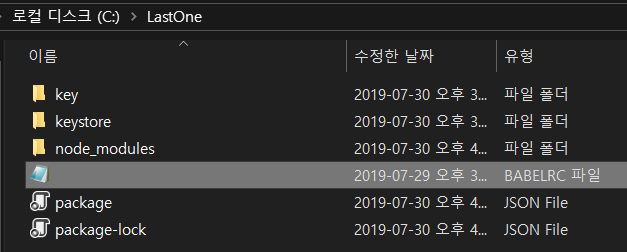
"keygen:client": "babel-node --extensions \".ts\" key/keygen.ts client",

"keygen:verify": "babel-node --extensions \".ts\" key/keygen.ts verify"

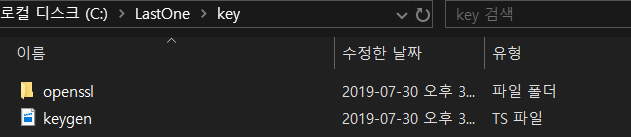
----------------------------------------------------------------------------------------------------

1. **폴더와 package.json 확인**

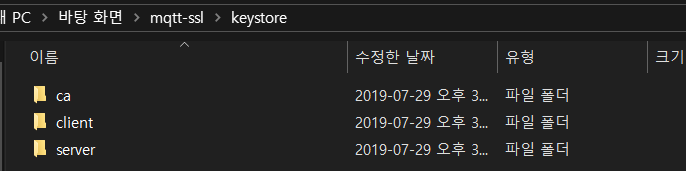
**프로젝트 폴더**



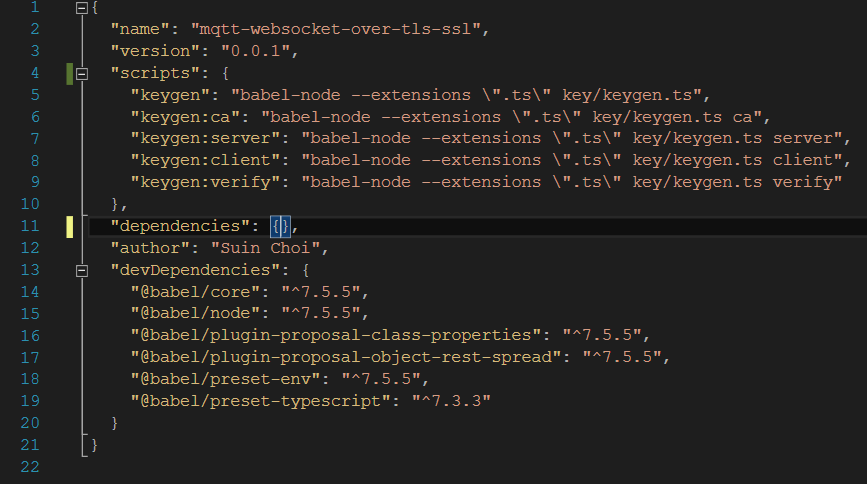
**프로젝트/key 폴더**



**프로젝트/keystore 폴더**



**package.json**



1. **사용법**

cmd의 프로젝트 위치에서

npm run keygen

을 사용하면 keystore의 ca, server, client에 자동으로 인증서가 생성됩니다.

npm run keygen:ca

npm run keygen:server

npm run keygen:client

를 사용하면 각각만 만들 수 있습니다.

**웹서버 구축과 ssl로 mqtt 브로커와 통신하기**

1. **MQTT 브로커 configure 설정(mosquitto사용)**

Mosquitto 브로커에서 Websockects over TLS/SSL을 사용하려면 configure 파일을 수정 또는 생성해야 합니다. '프로젝트파일경로'에는 프로젝트 주소를 추가해주세요.

myconf.conf

----------------------------------------------------------------------------------------------------

# WebSockets over TLS/SSL

listener 9883

protocol websockets

cafile 프로젝트파일경로\keystore\ca\ca.crt

keyfile 프로젝트파일경로\keystore\server\server.key

certfile 프로젝트파일경로\keystore\server\server.crt

tls\_version tlsv1.2

require\_certificate true

# End of WebSockets over TLS/SSL

----------------------------------------------------------------------------------------------------

mosquitto broker을 실행시 configure 파일 설정은

mosquitto -v -c configure파일경로

입니다. 이 명령문은 mosquitto가 설치된 폴더에서 실행합니다.

1. **프로젝트 생성**

원하는 위치에 프로젝트 폴더를 생성 후 프로젝트 안에 package.json 파일을 생성합니다.

만약 앞의 npm으로 빠르게 인증서 만들기를 실행했다면 기존 package.json에 빨간 부분만 추가해주세요.

package.json

----------------------------------------------------------------------------------------------------

{

"name" : "mqtt-websocket-over-ssl",

"version" : "0.0.1",

"main" : "index.js",

"scripts" : {

"start" : "node index.js"

},

"dependencies" : {},

"author" : "Suin Choi"

}

----------------------------------------------------------------------------------------------------

1. **필요한 모듈 설치**

프로젝트에 필요한 모듈들을 설치합니다.

프로젝트 위치의 cmd에

npm install --save express

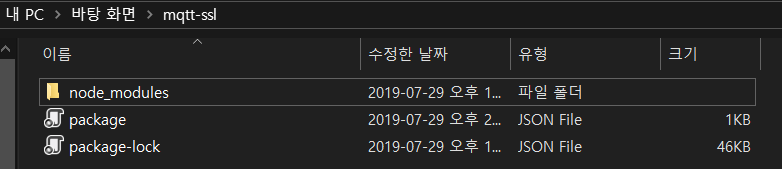
npm install --save socket.io

npm install --save mqtt

를 설치합니다.

1. **진행 확인**

현재 프로젝트 안에는 node\_modules 폴더와 package.json package-lock.json파일이 있어야 합니다.



1. **웹에서 mqtt와 ssl로 통신하기**

Node.js 를 이용해 localhost:3000 웹을 열 수 있습니다.

먼저 간단한 html을 프로젝트 폴더에 만듭니다.

Index.html

----------------------------------------------------------------------------------------------------

<!DOCTYPE html>

<html>

<head>

<title>Test</title>

<script src="/socket.io/socket.io.js"></script>

</head>

<body>

<button onclick = "onClick()">CONNECT</button>

<script>

function onClick(){

let socket = io();

socket.emit('new:connectMQTT');

console.log("clicked");

}

</script>

</body>

</html>

----------------------------------------------------------------------------------------------------

이 html에는 버튼이 하나 있고, 이 버튼을 클릭할 시 웹서버에 요청을 보냅니다.

이 html을 localhost:3000의 웹서버에 파싱 합니다. 프로젝트에 index.js 파일을 생성 후 아래 코드를 적어주세요. option에 host에 mosquitto ip를 넣어주세요.

index.js

----------------------------------------------------------------------------------------------------

const express = require('express');

const app = express();

const server = require('http').Server(app);

const io = require ('socket.io')(server);

const path = require('path');

const port = 3000;

const fs = require ('fs');

const mqtt = require('mqtt');

var client;

app.use(express.static(path.join(\_\_dirname, 'public')));

app.get('/', function(req, res){

res.sendFile(\_\_dirname + '/index.html');

});

server.listen(port, function(){

console.log('listening on port ' + port);

io.on('connection', function (socket) {

console.log(port + " CONNECTED");

socket.on('new:connectMQTT', function(){

console.log("connectMQTT start");

var option = {

host: "192.168.0.9",

port : 9883,

protocol: 'ws',

protocolId: 'MQIsdp',

protocolVersion: 3,

secureProtocol: 'TLSv1\_method',

reconnectPeriod: 5 \* 1000,

connectTimeout: 10 \* 1000,

rejectUnauthorized: false,

ca: [fs.readFileSync('keystore/ca/ca.crt')],

key: fs.readFileSync('keystore/client/client.key'),

cert: fs.readFileSync('keystore/client/client.crt')

};

client = mqtt.connect(option);

client.on('connect', function () {

client.subscribe('#', function (err){

if(!err) {

io.emit('new:mqttconnected');

console.log("MQTT\_CONNECTED");

}

});

});

});

});

});

---------------------------------------------------------------------------------------------------

이제 프로젝트 폴더에서

npm run start

를 치면 localhost:3000가 열립니다. 크롬 주소창에 localhost:3000으로 연결하면 index.html로 코딩한 화면이 보이고 버튼을 클릭하게 되면 서버가 요청을 받아 MQTT서버와 WebScoket으로 연결합니다. Index.html 과 index.js에 여러가지 함수 들을 추가하면 broker와 실시간으로 통신하며 업데이트 되는 동적인 페이지도 가능합니다. 아래 사이트들을 이용하면 위 코드들에 대한 이해와 기능 추가 시 유용할 것 같습니다.

Node.js tls 이용법

<http://nodejs.sideeffect.kr/docs/v0.10.35/api/tls.html#tls_tls_createsecurepair_credentials_isserver_requestcert_rejectunauthorized>

Node.js express 정적파일 사용법

<https://expressjs.com/ko/starter/static-files.html>

Node.js express 기본 라우팅 사용법

<https://expressjs.com/ko/starter/basic-routing.html>

Node.js socket.io 사용법

<https://github.com/socketio/socket.io-client>