OpenSSL

This guide provides instructions on using OpenSSL to:

1. Test SSL/TLS connections with a server.
2. Generate a new private key and a self-signed certificate for client authentication.

1. Testing SSL/TLS Connection with `openssl s_client`

The `openssl s_client` command is a tool for checking SSL/TLS connections with servers. It's commonly used to verify if a server's certificate is valid and to view the server's full certificate chain.

Command:

```
openssl s_client -connect {Host}:{Port} -showcerts
```

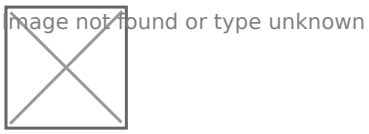
Explanation:

- **`s_client`**: Starts an SSL/TLS client connection.
- **`-connect {Host}:{Port}`**: Connects to the specified server and port (replace `{Host}` and `{Port}` with the appropriate values for your server, e.g., `broker.hivemq.com:8883` or `www.typicode.com:443`).
- **`-showcerts`**: Displays all certificates in the server's certificate chain.

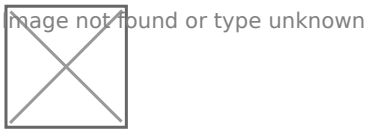
Steps:

1. **Finding Server Common Name**

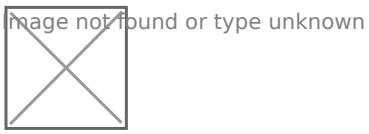
Open your browser and find the settings like below then click on "Connection is secure".



Click on "Certificate is valid"



Here you can find the server's common name which will be our host which is typicode.com. If it is a web server, then you need to add www. in front of the CN. Hence, our host is www.typicode.com



2. **Finding your Port**

For the port, you need to know which protocol you are using.

Ports that we will be using in this module :

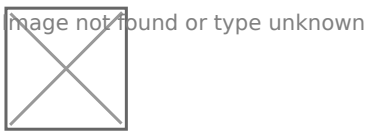
- HTTPS → 443
- MQTTS → 8883

So in this example, if we want to connect and hit the API from https://jsonplaceholder.typicode.com, our {Host}:{Port} combination will be **www.typicode.com:443**

3. **Run the Command**

```
openssl s_client -connect www.typicode.com:443 -showcerts
```

Then you need to scroll down and get the **LAST** certificate.



This will be your server's root CA.

2. Generating a New RSA Key and Self-Signed Certificate with openssl req

This command generates a new private key and a self-signed certificate, which can be used for client authentication.

Command:

```
openssl req -newkey rsa:2048 -nodes -keyout client_key.pem -x509 -days 365 -out client_cert.pem
```

Explanation:

- **-req**: Starts a new certificate request or generates a self-signed certificate.
- **-newkey rsa:2048**: Creates a new RSA key with a size of 2048 bits.
- **-nodes**: Ensures the private key is created without password encryption.
- **-keyout client_key.pem**: Saves the generated private key in client_key.pem.
- **-x509**: Generates a self-signed certificate instead of a certificate signing request (CSR).
- **-days 365**: Sets the certificate to be valid for 365 days.
- **-out client_cert.pem**: Outputs the certificate to client_cert.pem.

Steps:

1. **Run the Command**

Enter the command in your terminal to generate the private key and certificate.

2. **Provide Certificate Details**

Note, for testing purposes of this module, you can skip this step and fill blanks in all details.

You'll be prompted to enter information like Country, State, Organization, Common Name (CN), and Email Address. These fields will be included in the certificate's Subject field. The Common Name (CN) field is important, as it typically contains the hostname of the server or a unique identifier for the client in a client certificate setup. Check Output Files:

3. **After completion, you should find two new files**

client_key.pem: This is your private key. Keep it secure, as it identifies you in a client-server interaction.

client_cert.pem: This is your self-signed certificate, which can be provided to a server for authentication if client certificate verification is set up.

You can open them with notepad to see the certificate

Revision #2
Created 3 November 2024 16:37:08 by Giovan Christoffel Sihombing
Updated 7 November 2024 03:08:06 by Giovan Christoffel Sihombing