	C06 Valider un système informatique R2- Installation et qualification D2- Développement et validation de solutions logicielles	BTS CIEL TP

TP3 : Sécuriser MQTT avec TLS

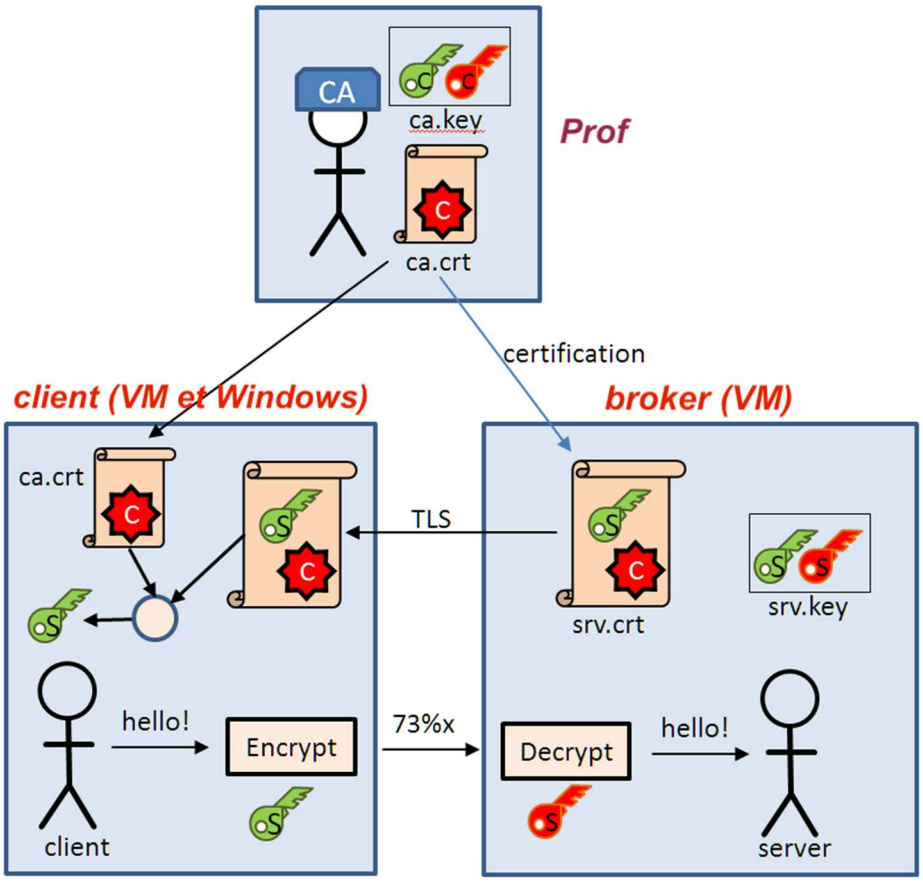
Objectifs

Sécuriser l'architecture MQTT existante avec du chiffrement (TLS)

Pré-requis

- Ce TP utilise l'architecture MQTT mise en place lors des TP précédents
    - broker : mosquitto sur la VM
    - client1 : mosquitto (client) sur la VM
    - client2 : MQTTX sur Windows
- Démarrer la VM.
- Démarrer le broker **sans authentification** (conserver une copie du fichier de configuration sans l'utiliser)
- Vérifier que l'architecture MQTT est fonctionnelle avec un pub / sub sur un topic 'ciel' en utilisant les deux clients

Avant de commencer : certificats et clefs



CA = Autorité de certification, c'est elle qui signe les certificats des clients et du serveur

L'autorité de certification pour le TP sera l'enseignant(e) qui produira les certificats des clients et du broker

Récapitulatif des fichiers

CA Certificate Authority (Autorité de certification) (enseignant)	ca.crt	Certificat fichier public (diffusé aux étudiants)
	ca.key	Clef fichier privé (gardé secret par la prof)
Client (étudiant)	client.key	clef privée
	client.csr	*requête de certificat signée (ne sert qu'à obtenir le certificat)*
	client.crt	certificat (public)
Broker (étudiant)	broker.key	clef privée
	broker.csr	*requête de certificat signée (ne sert qu'à obtenir le certificat)*
	broker.crt	certificat (public)

L'autorité de certification

- Télécharger et enregistrer le certificat du CA (à joindre au TP)
- Le certificat peut être ouvert et visualisé sous Windows
- Relever les informations suivantes

Paramètre	Valeur
Organisation (O)	
Unité d'organisation (OU)	
Nom (Common Name) du serveur	
Algorithme de signature	
Taille de la clé	
Date de validité (valide jusqu'au…)	

Présentation des étapes de la suite du TP

1. Générer la clef privée et la requête de certificat signé du broker
2. Générer la clef privée et la requête de certificat signé du client

3. Envoyer les fichiers de requêtes de certificats signés à l'autorité de certification (la prof)
4. Récupérer les certificats et copier les fichiers des clients et des brokers sur chaque machine
5. Configurer le broker et démarrer mosquitto avec la configuration TLS
6. Valider l'architecture avec une commande mosquitto_sub
7. Configurer le client MQTTX et valider la communication avec le broker

Générer les clefs et les demandes de signatures

Utilitaire openssl

Openssl est un utilitaire open-source qui permet de :

- Chiffrer et sécuriser les communications via SSL/TLS
- Générer des certificats pour authentifier les serveurs et clients
- Créer et gérer des clés de chiffrement

L'utilitaire 'openssl' est livré en standard avec Debian.

Vérifier que openssl est bien installé et donner son numéro de version :

Commandes principales

Commande	Rôle	Options / exemple
openssl **genrsa**	Crée une clef privée RSA	-out <file> : nom du fichier pour la clef Taille de la clef
Exemple openssl **genrsa** -out private.key 2048		
openssl **verify**	Vérifie qu'un certificat est valide par rapport à une CA	-CAfile <file_certificat_ca>
Exemple openssl **verify** -CAfile ca.crt server.crt		
openssl **req**	Génère des demandes de certificats (CSR) et des certificats auto-signés.	-out <file> : nom du fichier résultat -days <nb_jour> : nb de jours de validité de certificat -key <file> : clef pour signer le certificat ou la demande de certificat
Exemples openssl **req** -new -key private.key -out request.csr (créé une demande de certificate) openssl **req** -x509 -new -key private.key -days 365 -out cert.crt (créé un certificate signé par la CA)		

Fichiers du broker

Ces commandes se font depuis le terminal dans le répertoire utilisateur.

- ☐ Créer un répertoire "`tls`", se placer dans ce répertoire
- ☐ Générer la clef privée du broker (ajouter votre nom dans le fichier)

```
openssl genrsa -out <nom-broker.key> 2048
```

- ☐ Générer la "certificate signing request" pour le broker

```
openssl req -new -out <nom-broker.csr -key>  <nom-broker.key>
```

Informations à entrer :

Organization Name	Ciel <votre nom>
Organizational Unit Name	iot <votre nom>
Nom (Common Name) du serveur	le nom de votre machine (hostname)

Vérifier avec une commande Unix que deux fichiers ont été créés (copie d'écran)

Fichiers du client

- ☐ Générer la clef privée du client (ajouter votre nom dans le fichier)

```
openssl genrsa -out <nom-client.key> 2048
```

- ☐ Générer la "certificate signing request" pour le client

```
openssl req -new -out <nom-client.csr -key>  <nom-client.key>
```

Informations à entrer :

Organization Name	Ciel <votre nom>
Organizational Unit Name	<laisser blanc>
Nom (Common Name) du serveur	localhost

Vérifier avec une commande Unix que deux fichiers ont été créés (copie d'écran)

- ☐ Transmettre les fichiers .csr à l'enseignant (FTP) pour que le CA génère les certificats

Installation des fichiers

- ☐ Récupérer les fichiers .crt qui contiennent les certificats du client et du broker
- ☐ Copier les fichiers du broker dans le répertoire /etc/mosquitto/certs (noter la commande)

- ☐ Copier le fichier ca.crt dans le répertoire /etc/mosquitto/ca_certificates (noter la commande)

Configuration du broker

Cette étape consiste à paramètrer le broker pour qu'il soit en MQTTS (MQTT sécurisé avec TLS).

Sé référer à la documentation https://mosquitto.org/man/mosquitto-conf-5.html

- ☐ Faire une sauvegarde du fichier de configuration du broker (mosquitto-insecure.conf)
- ☐ Modifier le port d'écoute dans fichier de configuration mosquitto.conf (pour mettre le port qui correspond à MQTTS) 8883

Paramètres de sécurisation de MQTT

Trois paramètres permettent de régler la sécurisation de MQTT et peuvent être combinés

- require_certificate true/false
- use_identity_as_username true/false
- allow_anonymous true/false

Paramètre `require_certificate`

require_certificate false	require_certificate true
Le serveur présente son propre certificat auprès du client qui vérifie que le certificat est signé par une CA.. Le client n'a pas besoin de fournir de certificat. → la communication sera chiffrée mais pas de vérification de l'identité du client. Le broker doit avoir un certificat valide mais pas le client.	Le client doit obligatoirement fournir un certificat valide signé par une autorité reconnue. L'authentification repose alors sur ce certificat. Le broker et le client doivent avoir un certificat valide signé par une CA.

Documentation : By setting `require_certificate` to *true*, a client connecting to this listener must provide a valid certificate in order for the network connection to proceed. This allows access to the broker to be controlled outside of the mechanisms provided by MQTT.

Paramètre use_identity_as_username

`use_identity_as_username` false	`use_identity_as_username` true
L'authentification MQTT se fait comme d'habitude avec un username/password, même si le certificat est utilisé.	Le Common Name (CN) du certificat du client est utilisé comme nom d'utilisateur MQTT. Le mot de passe MQTT est ignoré, car le certificat est censé prouver l'identité du client.

Paramètre allow_anonymous

`allow_anonymous` false	`allow_anonymous_username` true
Permet aux clients sans username/password de se connecter.	Oblige les clients à fournir un username/password. Si la connexion nécessite un certificat, il faut fournir user/password en plus du certificat.

☐ Définir les paramètres à ajouter à la configuration du broker pour le cas d'utilisation suivant :

- Le broker est configuré pour MQTT avec TLS (clef +certificat du broker)
- Les clients s'authentifieront grâce à leurs certificats uniquement (pas de user/mot de passe demandé)

☐ Compléter le fichier en ajoutant l'emplacement des fichiers et le protocole utilisé

cafile #certificat du CA
certfile # certificat du broker
keyfile # clef privée du broker

tls_version tlsv1.2

☐ Recopier le contenu du fichier de configuration de mosquitto

☐ Démarrer le broker avec le fichier de configuration. Vérifier que mosquitto écoute sur le port 8883 (copie d'écran)

Connexion du client mosquitto

Les options utiles des commandes mosquitto_pub et mosquitto_sub sont

-p	port
-cafile	pour spécifier le fichier du certificat du CA
–cert	pour spécifier le fichier du certificat du client
–key	pour spécifier le fichier de clef privée du client
-d	debug (informations détaillées sur le moniteur)

- ☐ Exécuter une commande SUB depuis votre home directory sur le topic “ciel” (commande envoyée à votre broker)
- ☐ Noter la commande

- ☐ Copie d’écran de la commande et du résultat

Connexion du client MQTTX

Avant tout il faut s’assurer que le PC Windows puisse atteindre le hostname de votre VM sur le réseau. Pour cela nous modifierons le fichier “hosts” du PC.

En TCP/IP, une machine cherche en priorité dans ce fichier l’adresse IP d’un host distant avant de lancer une requête de résolution de nom.

- Exécuter NotePad++ en tant qu’administrateur
- Ouvrir le fichier “C:\Windows\System32\drivers\etc\hosts
- Ajouter une ligne pour votre host (hostname indiqué dans le certificat du broker)

Nous utiliserons les mêmes fichiers “client” que pour le client mosquitto.

- Copier les fichiers ca.crt, client.crt et client.key sur le PC
- Configurer une nouvelle connexion sous MQTTX en SSL/TLS
- Copie d’écran de la configuration

- ☐ Lancer une commande PUB depuis MQTTX
- ☐ Valider l'architecture (moniteur broker, moniteur client sub et publish sur MQTTX) en faisant plusieurs publications sur un ou deux topics

Analyse de trames

Capture de trames

- ☐ Installer le package tcpdump sur la VM
- ☐ Capturer les trames sur la VM avec la commande suivante (<interface> est le nom de l'interface réseau visible avec la commande ip a).

tcpdump -i <interface> -s 65535 host <IP> -w mqtt-nom.pcap

- ☐ Effectuer quelques commandes MQTT pendant la capture tcpdump :
  - Connexion du client MQTTX connecté en MQTTS et sub au topic 'ciel'
  - Connexion du client mosquitto connecté en MQTTS (sub au topic 'ciel')
  - MQTTX envoie un publish MQTT sur le topic 'ciel',
  - MQTTX se déconnecte
- ☐ Arrêter la capture avec CTR-C

Analyse des trames

- ☐ Récupérer le fichier de capture avec SCP
- ☐ Ouvrir le fichier dans Wireshark
- ☐ Analyser les trames
- ☐ Est-ce que les données circulent en clair ou sont cryptées ?? (copie d'écran)

- ☐ Comment pourrait-on améliorer la sécurisation de l'application ?