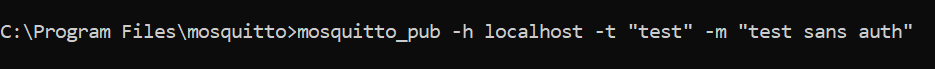
***Compte rendu TP OC02 : Sécurisation d’une plateforme MQTT pour véhicules connectés***

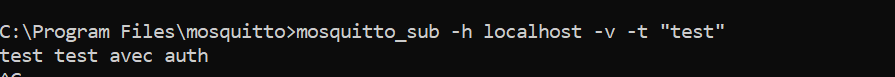
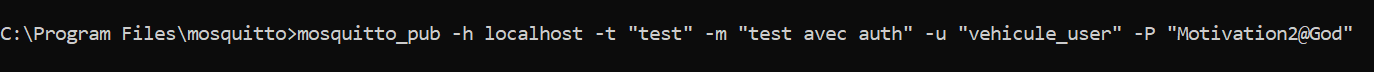
* **Etape 1** : authentification par mot de passe
* Création du fichier de mot de passe



* Configuration de Mosquitto(confert fichier)
* Test de l’authentification
* Sans identifiants



* Avec identifiants



* **Exercice** : Documentation de comment j’integrerai un gestionnaire de secrets dans mon architecture.

**Intégration d’un Gestionnaire de Secrets (AWS Secrets Manager)**

Afin de corriger la faiblesse liée à l’envoi et au stockage non sécurisé des identifiants, nous proposons d’intégrer un gestionnaire de secrets tel qu’**AWS Secrets Manager**. Cette solution permet de **stocker, accéder et faire tourner automatiquement les secrets** (mots de passe, clés API, certificats) de manière sécurisée.

1. **Stockage Sécurisé**

**Principe** : Les identifiants sont chiffrés au repos à l’aide de AWS KMS.

**Implémentation** : Les mots de passe des clients MQTT sont stockés dans AWS Secrets Manager au lieu d’être en clair dans les fichiers.

1. **Accès Contrôlé**

**Principe** : L’accès aux secrets est restreint par des politiques IAM.

**Implémentation** : Seuls les services ou applications autorisés peuvent lire les secrets, via des rôles IAM configurés selon le principe du moindre privilège.

1. **Rotation Automatique**

**Principe** : Les secrets doivent être changés régulièrement pour réduire le risque de compromission.

**Implémentation** : AWS Secrets Manager permet une **rotation automatique des mots de passe** à intervalles définis, sans intervention manuelle.

1. **Intégration Transparente dans l’Architecture**

**Principe** : Les secrets ne doivent jamais être codés en dur dans l’application.

**Implémentation** : Les applications ou scripts clients récupèrent dynamiquement les secrets via l’**API AWS Secrets Manager** ou les SDK, avec une connexion TLS sécurisée.

1. **Audit et Surveillance**

**Principe** : Toute utilisation d’un secret doit être traçable.

**Implémentation** : L’accès aux secrets est journalisé dans AWS CloudTrail, permettant de détecter tout accès non autorisé ou comportement anormal.

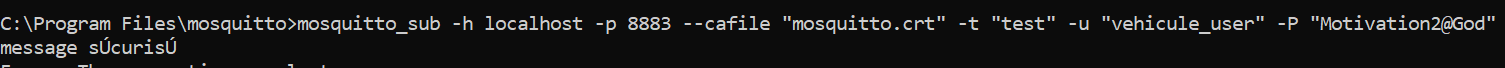
**Conclusion**

L’intégration d’un gestionnaire de secrets comme **AWS Secrets Manager** permet d’éliminer l’envoi de mots de passe en clair et d’améliorer considérablement la gestion sécurisée des identifiants. Cette approche répond directement aux limites identifiées, tout en s’alignant sur les bonnes pratiques de sécurité modernes pour une architecture MQTT robuste et scalable.

* **Etape2**: Chiffrement TLS
* Génération d’un certificat auto-signé



* Configuration de Mosquitto pour TLS(voir fichier)
* Test de communication sécurisée



* Analyse du trafic

### ****Analyse du Trafic avec Wireshark****

### Communication MQTT non sécurisée (port 1883)

### Les messages, noms d’utilisateur et mots de passe sont visibles en clair dans les paquets capturés.Toute personne sur le même réseau peut intercepter et lire ces données.

### ****Risques :**** interception, usurpation d’identité, modification des messages.

#### Communication MQTT sécurisée par TLS (port 8883)

#### Le trafic capturé est **chiffré** : aucun message ni identifiant n’est lisible.

#### TLS garantit **confidentialité**, **intégrité** et **authentification** des échanges.

#### Pourquoi TLS protège contre les attaques MitM

#### **Chiffrement** : empêche l’espionnage du trafic.

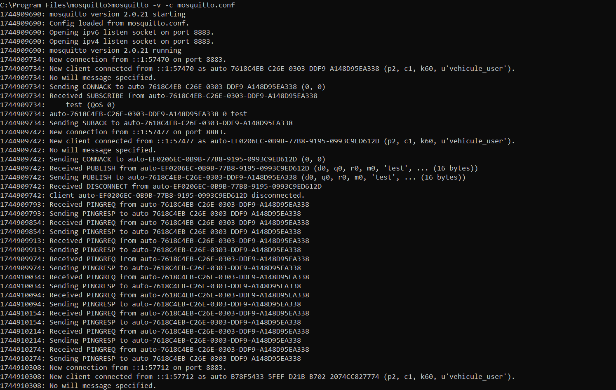
#### **Authentification** : vérifie que les deux parties sont légitimes.

#### **Intégrité** : les données ne peuvent être altérées en transit sans être détectées.

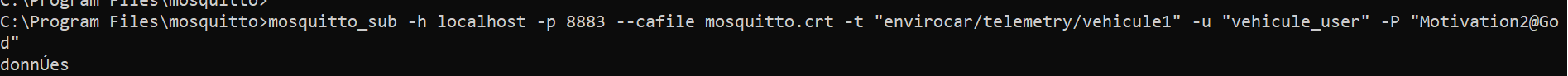
#### **Conclusion**

#### L’analyse montre clairement que **TLS est indispensable** pour sécuriser MQTT : il protège contre les interceptions, les falsifications et les attaques de type **Man-in-the-Middle**.

* Communication MQTT sécurisée par TLS(port 8883)



* **Etape 3)** Contrôle d’accès ACL
* Test de restriction d’accès
* Topic autorisé



* Topic non autorisé





* Exercice d’analyse

### Rôle des ACL dans le Diagramme AWS (Points 2, 5 et 6)

### Les **Access Control Lists (ACL)** assurent la **compartimentation des communications MQTT** et limitent l’impact d’une compromission en restreignant précisément qui peut publier ou s’abonner à quels topics.

### ****Point 2 : AWS IoT Core****

**Rôle** : Broker central qui gère les communications MQTT et applique les politiques d’autorisation.

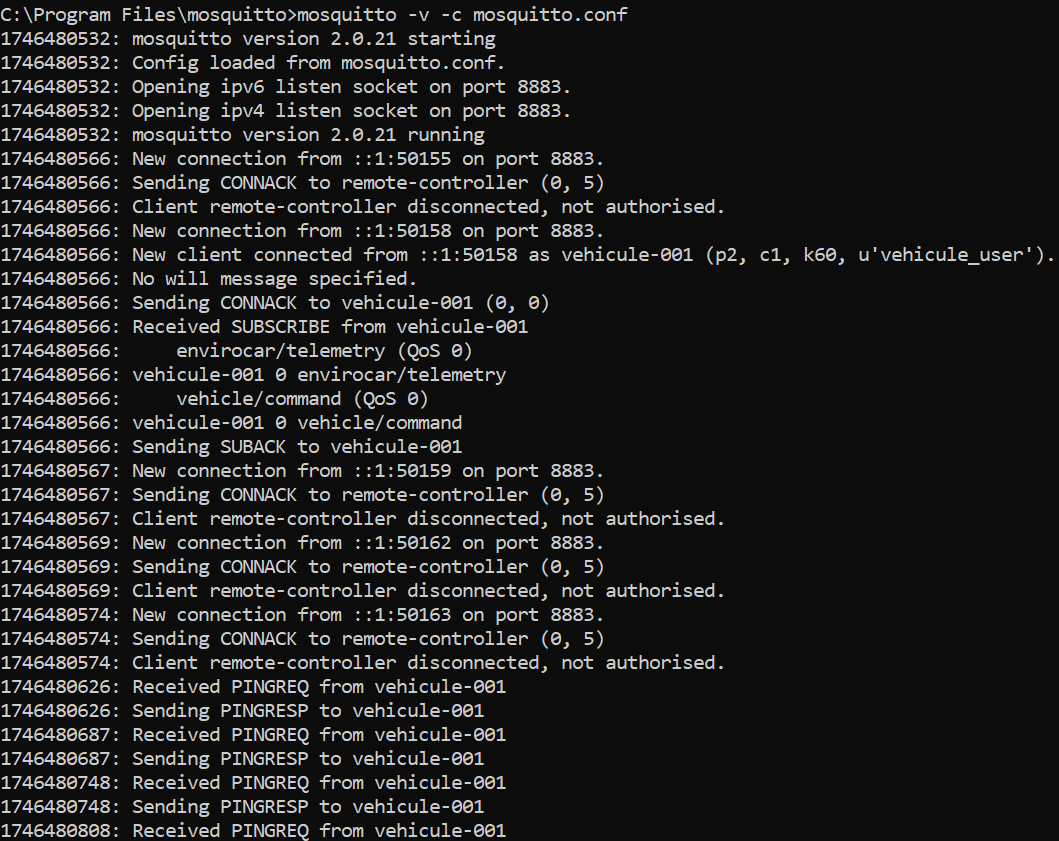
* **Compartimentation** :
  + Segmentation des topics selon les utilisateurs ou types de données.
  + Isolation des dispositifs pour empêcher qu’une compromission ne s’étende.
* **Impact en cas de compromission** : L’accès reste limité au périmètre autorisé par les ACL.
* **Point 5 : Lambda Function Receive Telemetry**
* **Rôle** : Traite les données de télémétrie des véhicules.
* **Compartimentation** :
  + Accès restreint uniquement aux topics de télémétrie nécessaires.
  + Contrôle précis du flux de données.
* **Impact en cas de compromission** : L’attaquant ne peut pas accéder à d’autres types de données.
* **Point 6 : Lambda Function Remote Command**
* **Rôle** : Envoie des commandes aux véhicules via MQTT.
* **Compartimentation** :
  + Seules les fonctions autorisées peuvent publier sur les topics de commande.
  + Séparation stricte entre types de commandes et cibles.
* **Impact en cas de compromission** : Empêche l’envoi de commandes non autorisées à d’autres véhicules.

Conclusion

Les ACL sont essentielles pour :

* **Isoler les composants** de l’architecture,
* **Limiter l’accès aux topics sensibles**,
* **Réduire l’impact d’une compromission**,  
  assurant ainsi une sécurité robuste pour les communications dans un environnement de véhicules connectés.
* **Etape 4** : Simulation d’un système requête –réponse sécurisée

Avec mon implémenation j’obtiens



* Analyse de sécurité :

**Point 6 : Lambda Function Remote Command**

**FonctionnalitésAWS:**  
AWS permet l’envoi sécurisé de commandes à distance via Lambda, avec des politiques d’autorisation fines et un chiffrement/signalisation des messages.

**Notre Implémentation :**

* **Signature HMAC** : garantit l’intégrité et l’authenticité des commandes.
* **Chiffrement TLS** : sécurise les données en transit.
* **Vérification du timestamp** : évite les attaques par rejeu.

**Améliorations possibles :**

* **Authentification par certificat client** : sécurité renforcée par rapport aux mots de passe.
* **Politiques d’autorisation granulaires** : gestion des rôles et permissions selon les types de commandes.

**Point 9 : Lambda Find Remediation**

**FonctionnalitésAWS:**  
AWS utilise Lambda avec Security Hub et IoT Device Defender pour détecter les anomalies et lancer des actions correctives automatiques.

**Notre Implémentation :**

* Prévention via **TLS** et **HMAC**.
* Absence actuelle de détection d’anomalies ou de remédiation automatique.

**Améliorations possibles :**

* **Surveillance comportementale** : détection d’activités anormales (connexions, commandes suspectes).
* **Réponse automatisée** : scripts Lambda pour bloquer, alerter ou révoquer des accès en cas d’incident.
* **Etape 5)** Analyse de vulnérabilité et améliorations

### ****Évaluation de la Sécurité MQTT – Test de Pénétration Simplifié****

Afin d’évaluer la robustesse de notre implémentation MQTT sécurisée, nous avons mené un test de pénétration simplifié. L’objectif était d’identifier d’éventuelles vulnérabilités et d’observer le comportement du système face à des tentatives d’intrusion.

### ****1. Tentative d'Interception****

**Objectif :**  
Tester la confidentialité des communications en interceptant le trafic réseau avec Wireshark.

**Procédure :**

* Capture du trafic réseau sur les ports 1883 (non sécurisé) et 8883 (sécurisé).
* Envoi de messages MQTT avec et sans TLS.

**Résultats :**

* **Sans TLS** : Identifiants et messages visibles en clair. Risque élevé d’interception (attaque Man-in-the-Middle).
* **Avec TLS** : Trafic chiffré et illisible. TLS protège efficacement les données échangées.

**Conclusion :**  
L’usage de TLS est indispensable pour garantir la confidentialité et l’intégrité des communications MQTT.

### ****2. Tentative d’Accès Non Autorisé****

**Objectif :**  
Tester le contrôle d’accès via les ACL du broker MQTT.

**Procédure :**

* Configuration d’un fichier ACL restreignant l’accès à certains topics.
* Tentative de publication sur un topic non autorisé via un client non privilégié.

**Résultats :**

* **Comportement du broker** : Rejet immédiat de la publication avec message d’erreur "accès refusé".
* **Rôle de TLS** : Ne contrôle pas l’accès, mais empêche l’usurpation ou la falsification de messages.

**Conclusion :**  
Les ACL sont efficaces pour contrôler l’accès aux topics. Associées à TLS, elles assurent une protection complète contre les accès non autorisés.

* **Améliorations possibles**

### ****Améliorations de la Sécurité MQTT****

Pour renforcer la sécurité de notre implémentation MQTT, trois axes d'amélioration principaux sont identifiés : l’authentification par certificat client, la protection contre les attaques DoS, et la surveillance des comportements anormaux.

### ****1. Authentification par Certificat Client****

**Description :**  
Remplace l’authentification par mot de passe par des certificats X.509 uniques pour chaque client, délivrés par une autorité de certification.

**Avantages :**

* **Sécurité accrue** : les certificats sont plus difficiles à falsifier que les mots de passe.
* **Gestion centralisée** : facilité de révocation et de rotation des identifiants.

**Implémentation :**

* Génération de certificats via une CA interne ou tierce.
* Configuration du broker (Mosquitto) pour n’accepter que les clients certifiés.
* Stockage sécurisé des certificats côté client.

### ****2. Protection contre les Attaques DoS****

**Description :**  
Les attaques DoS visent à saturer le broker avec des connexions ou messages massifs (ex. : attaque SlowITe).

**Contremesures :**

* **Limiter les connexions par IP** : via règles pare-feu ou configuration du broker.
* **Limiter le taux de publication** : quota de messages ou bande passante par client.
* **Utiliser un proxy ou pare-feu applicatif** : comme NGINX ou HAProxy pour filtrer et contrôler le trafic entrant.

### ****3. Surveillance et Détection d’Anomalies****

**Description :**  
Permet d’identifier les comportements suspects ou malveillants en temps réel, à l’image d’AWS IoT Device Defender.

**Implémentation :**

* **Monitoring des clients** : collecte de logs et visualisation avec Prometheus/Grafana.
* **Détection d’anomalies** : analyse de schémas inhabituels via règles ou machine learning.
* **Alertes et remédiation** : notification des incidents et réponse automatique (blocage ou isolement du client).

### ****Conclusion****

### En combinant **certificats clients**, **protection DoS** et **détection comportementale**, nous augmentons significativement la résilience et la sécurité de notre infrastructure MQTT. Ces mesures sont essentielles pour garantir la fiabilité des communications dans un environnement de véhicules connectés exposé à des menaces potentielles.

Conclusion et Synthèse

### ****Synthèse de la Sécurisation de la Plateforme MQTT****

Dans le cadre de ce projet, nous avons sécurisé une plateforme MQTT dédiée aux véhicules connectés, en traitant plusieurs vulnérabilités critiques et en intégrant des mécanismes de sécurité essentiels.

### ****1. Vulnérabilités Adressées****

* **Absence d’authentification** : Introduction d’une authentification par mot de passe pour bloquer les accès non autorisés.
* **Communications non sécurisées** : Intégration de **TLS** pour chiffrer les échanges et prévenir les interceptions.
* **Accès libre aux topics** : Mise en place de **ACL** pour restreindre les publications et abonnements aux topics sensibles.
* **Commandes non vérifiées** : Utilisation de **HMAC** pour garantir l’intégrité et l’authenticité des commandes.

### ****2. Mécanismes de Sécurité Implémentés****

* **Authentification par mot de passe** : Chaque client doit fournir des identifiants valides.
* **Chiffrement TLS** : Toutes les communications sont protégées contre les écoutes et modifications.
* **Contrôle d’accès (ACL)** : Règles strictes limitant les opérations selon les utilisateurs.
* **Vérification des commandes** : Signature HMAC + horodatage pour bloquer les messages falsifiés ou expirés.

### ****3. Limites par Rapport au Modèle AWS****

* **Authentification par certificat** : Non utilisée, alors qu’AWS l’impose pour une sécurité renforcée.
* **Surveillance & remédiation automatique** : Absent, contrairement aux solutions AWS comme IoT Device Defender.
* **Politiques d’autorisation avancées** : Moins granulaires et moins dynamiques que celles d’AWS (IAM, roles, etc.).

**4. Améliorations Futures Possibles**

* **Authentification par certificat client** : Remplacer les mots de passe pour renforcer l’identification des clients.
* **Protection DoS** : Limiter les connexions/messages, filtrer via proxy ou pare-feu applicatif.
* **Surveillance & détection d’anomalies** : Intégrer un système de monitoring avec alertes et réponse automatique.
* **Politiques d’autorisation fines** : Définir des règles plus précises selon les types d’utilisateurs et d’actions.

### ****Conclusion****

Notre implémentation constitue une base solide pour sécuriser une plateforme MQTT dans un environnement de véhicules connectés. Pour atteindre un niveau de sécurité équivalent à celui d’AWS, des évolutions sont nécessaires, notamment en matière d’authentification, de surveillance et de contrôle d’accès. Ces améliorations renforceront la résilience du système face aux menaces et garantiront une communication fiable et sécurisée.

* **Question de Réflexion**

### ****Gestion de la Sécurité pour une Flotte de Véhicules Connectés****

### Dans le cadre de notre projet, nous avons envisagé les défis liés à la sécurisation d’une flotte de milliers de véhicules connectés. Voici les principales approches retenues et considérations stratégiques.

### ****1. Gestion de la Sécurité à Grande Échelle****

**Authentification centralisée**

* Utilisation d’une autorité de certification (CA) pour délivrer et gérer les certificats clients de manière centralisée.

**Surveillance et analyse**

* Déploiement d’outils de monitoring pour détecter les comportements anormaux.
* Analyse des logs pour identifier les menaces émergentes.

**Mises à jour sécurisées**

* Intégration d’un mécanisme de mise à jour OTA (Over-The-Air) avec signature numérique.

**Segmentation du réseau**

* Isolation des flux critiques pour limiter la propagation des attaques.

### ****2. Compromis Sécurité vs. Performances et Complexité****

**Authentification par mot de passe**

* Simplicité d’implémentation, mais sécurité moindre par rapport aux certificats.

**Chiffrement TLS**

* Ajoute une charge de traitement, mais essentiel pour la protection des données en transit.

**ACL et HMAC**

* Apportent une sécurité supplémentaire, au prix d’une complexité de gestion accrue.

### ****3. Adaptation aux Contrainte de Connectivité****

**Stockage local**

* Enregistrement temporaire des données en cas de perte de connexion.

**Communication asynchrone**

* Envoi différé des messages selon la disponibilité du réseau.

**Optimisation des messages**

* Réduction de la taille des paquets pour s’adapter aux faibles débits.

**Résilience réseau**

* Reprise automatique des opérations à la reconnexion.

### ****4. Risques Persistants Malgré les Mesures****

**Vulnérabilités inconnues**

* Risque de failles non découvertes dans les logiciels ou bibliothèques utilisées.

**Attaques sophistiquées**

* Nécessité d’une surveillance constante pour faire face à des menaces avancées.

**Erreurs humaines**

* Configurations incorrectes ou fuites d’identifiants pouvant compromettre la sécurité.

**Complexité de gestion**

* Risques opérationnels accrus en cas de mauvaise gestion de l’infrastructure sécurisée.

### ****Conclusion****

La sécurisation d’une infrastructure MQTT à grande échelle repose sur un équilibre entre **robustesse, efficacité et adaptabilité**. En anticipant les limites techniques et opérationnelles, et en intégrant des mécanismes de sécurité solides, il est possible de bâtir une solution fiable pour les véhicules connectés tout en maintenant une gestion soutenable et réactive face aux menaces évolutives.