

OpenSSL\_Communication PHPmyadmin\_mariaDB

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Rédacteur** | **Changements par rapport à la version précédente** |
| 1.0 | 04/04/2025 | KETTERER  Anthony  BTS CIEL | Version initiale |
| 1.1 | 14/04/2025 | KETTERER  Anthony  BTS CIEL | Version corrigée |
| 1.2 | 25/04/2025 | MASSON SCIAUX Romaric  CMIL | Version corrigée |

Table des matières

1. Introduction

2. Prérequis

3. Explication du procédé

4. Conclusion

1. Introduction

Ce document décrit la mise en place de certificats SSL/TLS afin de sécuriser la communication entre la base de données MariaDB et PHPmyadmin.

Cette documentation est destinée au professeurs/jurys notant l’ensemble de mon projet réalisé lors de ma deuxième année de BTS CIEL.

1. Prérequis

Ce document a pour but de sécuriser les communications entre les deux logiciels cité au-dessus sur un **OS Linux (Debian 12)**.

1. Explication du procédé

Nous allons maintenant nous connecter à notre OS Linux et commencer la configuration.

Une image contenant capture d’écran, texte, noir, astronomie

Le contenu généré par l’IA peut être incorrect.

Une fois entré dans notre terminal en mode ‘su -’ (root), créer et accéder au dossier /etc/mysql/ssl car c’est ici que nous voulons créer les clés de certificats pour des raisons d’organisation de fichiers.

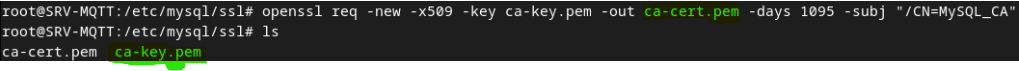
* Mkdir /etc/mysql/ssl : permet de créer le dossier ssl à cette emplacement.
* Cd /etc/mysql/ssl : permet d’aller à cette emplacement.
* openssl genpkey -algorithm RSA -out ca-key.pem :

openssl genpkey : Génère une clé privée.

-algorithm RSA : Utilise l’algorithme RSA pour la clé.

-out ca-key.pem : Stocke la clé privée de la CA dans ca-key.pem.

* ls : permet de voir les fichiers/dossiers présent à l’emplacement actuel.





* openssl req -new -x509 -key ca-key.pem -out ca-cert.pem -days 1095 -subj "/CN=MySQL\_CA"

openssl req -new -x509 : Crée un certificat auto-signé pour la CA.

-key ca-key.pem : Utilise la clé privée ca-key.pem pour signer le certificat.

-out ca-cert.pem : Enregistre le certificat CA dans ca-cert.pem.

-days 1095 : Définit la validité du certificat à 1095 jours.

-subj "/CN=MySQL\_CA" : Définit le nom commun (CN) du certificat comme **MySQL\_CA**.

Une image contenant capture d’écran, texte, astronomie

Le contenu généré par l’IA peut être incorrect.

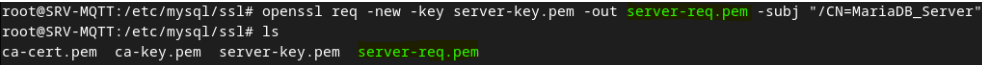


* openssl genpkey -algorithm RSA -out server-key.pem

openssl genpkey : Génère une clé privée.

-algorithm RSA : Utilise l’algorithme RSA pour la clé.

-out serveur-key.pem : Génère une clé privée server-key.pem pour MariaDB.



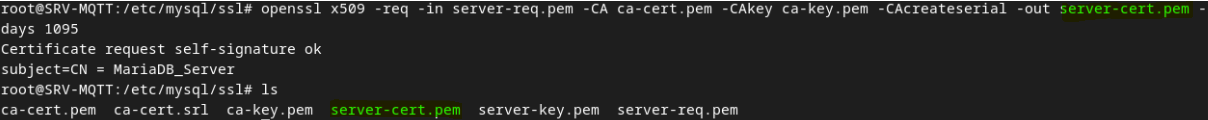
* openssl req -new -key server-key.pem -out server-req.pem -subj "/CN=MariaDB\_Server"

-new : Crée une requête de signature de certificat (CSR).

-key server-key.pem : Utilise la clé privée du serveur.

-out server-req.pem : Enregistre la requête CSR.

-subj "/CN=MariaDB\_Server" : Définit le **nom commun (CN)** du certificat en **MariaDB\_Server**.



* openssl x509 -req -in server-req.pem -CA ca-cert.pem -CAkey ca-key.pem -CAcreateserial -out server-cert.pem -days 1095

openssl x509 -req -in server-req.pem: Transforme la requête CSR (server- req.pem) en certificat signé.

-CA ca-cert.pem -CAkey ca-key.pem : Utilise le certificat et la clé de la CA pour signer le certificat du serveur.

-CAcreateserial : Génère un numéro de série unique pour le certificat.

-out server-cert.pem : Stocke le certificat signé dans server-cert.pem.

-days 1095 : Valide le certificat pour **1095 jours**.

Une image contenant capture d’écran, texte, astronomie

Le contenu généré par l’IA peut être incorrect.

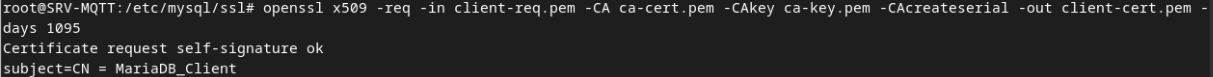
* openssl genpkey -algorithm RSA -out client-key.pem

Génère une clé privée client-key.pem pour le client.



* openssl req -new -key client-key.pem -out client-req.pem -subj "/CN=MariaDB\_Client"

Génère une requête CSR (client-req.pem) pour le client avec CN=MariaDB\_Client.



* openssl x509 -req -in client-req.pem -CA ca-cert.pem -CAkey ca-key.pem -CAcreateserial -out client-cert.pem -days 1095

openssl x509 -req : Transforme la requête CSR (server-req.pem) en certificat signé.

-CA ca-cert.pem -CAkey ca-key.pem : Utilise le certificat et la clé de la CA pour signer le certificat du serveur.

-CAcreateserial : Génère un numéro de série unique pour le certificat.

-out server-cert.pem : Stocke le certificat signé dans server-cert.pem.

-days 1095 : Valide le certificat pour **1095 jours**.



* chmod 600 \*.pem

chmod 600 : Définit les **permissions** des fichiers .pem pour qu’ils ne soient lisibles et modifiables que par leur propriétaire.

\*.pem : Applique cette restriction à tous les fichiers .pem dans le dossier. Cela empêche d'autres utilisateurs d'accéder aux clés privées, garantissant la **sécurité** des connexions SSL.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.



**ssl-ca = /etc/mysql/ssl/ca-cert.pem** : spécifie le **certificat de l'Autorité de Certification (CA)** utilisé pour vérifier les certificats clients.

**ssl-cert = /etc/mysql/ssl/server-cert.pem** : définit le **certificat ssl** utilisé par le serveur MariaDB pour s’authentifier auprès des clients.

**ssl-key = /etc/mysql/ssl/server-key.pem** : correspond à la **clé privée** associée au certificat du serveur.

**require\_secure\_transport = ON** : force l'utilisation de connexions sécurisées (SSL/TLS). si un client tente de se connecter sans SSL, MariaDB refusera la connexion.

Une image contenant texte, capture d’écran, Police, ligne

Le contenu généré par l’IA peut être incorrect.



* **$cfg['Servers'][$i]['ssl'] = true;**

Cette ligne active l'utilisation de SSL pour la connexion au serveur MariaDB. En définissant cette option sur true, phpMyAdmin établira une connexion sécurisée (chiffrée) entre le client (phpMyAdmin) et le serveur MariaDB.

* **$cfg['Servers'][$i]['ssl\_ca'] = '/etc/mysql/ssl/ca-cert.pem';**

Ce paramètre spécifie le chemin du certificat de l'autorité de certification (CA) qui est utilisée pour vérifier la validité du certificat du serveur. Le fichier ca-cert.pem contient la clé publique de l'autorité de certification. Cela permet de garantir que la connexion au serveur est sécurisée et authentique.

* **$cfg['Servers'][$i]['ssl\_cert'] = '/etc/mysql/ssl/client-cert.pem';**

Cette ligne spécifie le chemin du certificat du client (phpMyAdmin) utilisé pour s'authentifier auprès du serveur MariaDB. Le fichier client- cert.pem contient la clé publique du client. Cela permet au serveur de vérifier l'identité du client avant d'accepter la connexion.

* **$cfg['Servers'][$i]['ssl\_key'] = '/etc/mysql/ssl/client-key.pem';**

Ce paramètre indique le chemin de la clé privée du client. Le fichier client-key.pem contient la clé privée du client qui correspond au certificat spécifié précédemment. Elle est utilisée pour établir la connexion sécurisée, prouvant que le client possède bien la clé privée associée au certificat.

// $cfg['Servers'][$i]['ssl'] = true; pour connexion tcp/ip

// $cfg['Servers'][$i]['ssl'] = false; pour connexion socket unix

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Entrer ensuite dans la base de données avec un utilisateur ayant des droits pour exécuter les commandes suivantes :

* + SHOW VARIABLES LIKE ‘have\_ssl’; : Cette commande SQL permet de vérifier si le serveur MariaDB prend en charge SSL (Secure Sockets Layer).

C’est le cas ici car la valeur est ‘YES’. Si la valeur avait été ‘DISABLED’ cela aurait signifié que SSL est installé sur le serveur MariaDB mais désactivé. Si la valeur avait été ‘NO’ cela aurait signifié que SSL n’est pas installé sur le serveur MariaDB.

Une image contenant texte, capture d’écran, Police, noir

Le contenu généré par l’IA peut être incorrect.

* SHOW STATUS LIKE ‘Ssl\_cipher’ ; : la variable **Ssl\_cipher** montre le chiffrement SSL utilisé pour la connexion **actuelle** entre le client (phpMyAdmin) et le serveur MariaDB. Si phpMyAdmin utilise une connexion sécurisée via SSL, la valeur de **Ssl\_cipher** sera remplie avec le nom de l'algorithme de chiffrement utilisé pour cette connexion (TLS\_AES\_256\_GCM\_SHA384).

Selon des documentations externes :

**TLS** : Cela fait référence au protocole Transport Layer Security utilisé pour sécuriser la communication. TLS est l'évolution de SSL (Secure Sockets Layer) et est utilisé pour chiffrer les données échangées entre un client et un serveur.

**AES** : Cela fait référence à l'algorithme de chiffrement Advanced Encryption Standard. AES est un algorithme de chiffrement symétrique largement utilisé pour garantir la confidentialité des données.

* Le chiffre 256 dans AES\_256 indique la taille de la clé de chiffrement, qui est de 256 bits. Plus la clé est grande, plus l'algorithme est sécurisé (en théorie).

**GCM** : GCM (Galois/Counter Mode) est un mode de fonctionnement pour le chiffrement AES. GCM est un mode qui fournit à la fois chiffrement et intégrité des données. Il est particulièrement utilisé pour offrir à la fois des performances élevées et une résistance à des attaques spécifiques.

**SHA384** : SHA384 fait référence à l'algorithme de hachage SHA-2 (Secure Hash Algorithm 2), mais avec une taille de sortie de 384 bits. SHA-384 est utilisé pour fournir une fonction de hachage cryptographique dans le cadre du processus d'intégrité des données. Il est utilisé dans le processus de vérification de l'intégrité des messages, en s'assurant que les données n'ont pas été altérées en cours de transmission.

1. Conclusion

Dans cette documentation, nous avons abordé l'importance de sécuriser la communication entre le serveur MariaDB et le client phpMyAdmin en utilisant des connexions SSL/TLS. En activant SSL sur MariaDB et en configurant correctement phpMyAdmin pour qu'il utilise SSL lors des connexions, nous garantissons que les données échangées entre le serveur et le client restent confidentielles et intégrales.

Cette approche garantit une protection accrue des données et réduit les risques d'interception ou de modification pendant la transmission. Nous vous encourageons à tester votre configuration en vous assurant que la connexion est bien sécurisée, et à ajuster les paramètres en fonction des besoins spécifiques de votre environnement.

**Ressources supplémentaires** :

**Ressources supplémentaires** :

* Documentation officielle de MariaDB sur SSL (<https://dev.mysql.com/doc/refman/8.4/en/creating-ssl-files-using-openssl.html>)
* phpMyAdmin - Sécurisation des connexions avec SSL

(<https://docs.phpmyadmin.net/fr/latest/config.html>)