JAVAID Omer
RT2A1

SAE 302-Développer des applications communicantes

Rapport Test/Validation

1. Introduction

Dans le cadre de la SAE 302, nous avons développé une application de chat sécurisée utilisant Java, Swing pour l'interface graphique, et SSL/TLS pour protéger les échanges. Le stockage des messages se fait dans une base MySQL.

Ce rapport détaille les tests effectués pour valider :

1. **Le lancement du serveur et sa capacité à chiffrer les communications (SSL/TLS).**
2. **La connexion depuis plusieurs clients (interface Swing et openssl s_client).**
3. **L'affichage des messages (en direct et via l'historique) et leur insertion en base de données MySQL.**
4. **La confirmation du chiffrement observée via Wireshark.**

2. Environnement de Test

- **Système : Windows 11**
- **Java : OpenJDK (ou Oracle) 17**
- **Base de données : MySQL 8.0.40 (BD chat_secure, table messages)**
- **Mot de passe MySQL : Vitrygtr94_**
- **Outils : Eclipse, Wireshark, OpenSSL**
- **Ports : Serveur écoutant sur port 1234**
- **Fichiers KeyStore / TrustStore :**
  - **serveurkeystore.jks**
  - **servertruststore.jks**
  - **Mot de passe : vitrygtr**

SAE 302-Développer des applications communicantes

3. Déroulé des Tests

3.1 Lancement du Serveur et Console

1. **Objectif : Vérifier que le serveur démarre et écoute sur le port sécurisé 1234.**
2. **Étapes :**
   - **Exécuter SSLChatServer depuis l'IDE Eclipse.**
   - **Contrôler le message de confirmation dans la console.**
3. **Résultat Attendu : La console affiche « *Serveur chat SSL lancé sur le port 123***

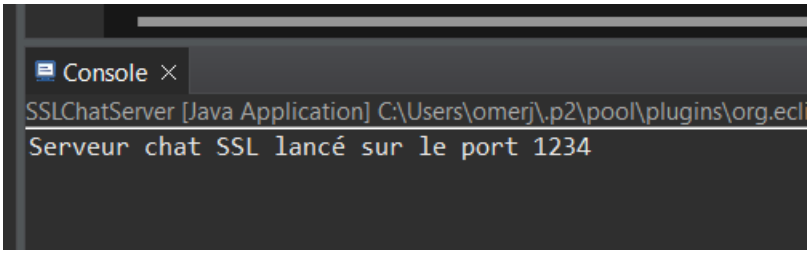

Figure 1- Console du serveur affichant le message de confirmation « Serveur chat SSL lancé sur le port 1234

3.2 Connexion avec OpenSSL - Tests Multi-Utilisateurs

1. **Objectif** : Tester la communication SSL/TLS depuis une console OpenSSL.
2. **Étapes** : Exécuter cette commande

Figure 2 - Test de connexion avec OpenSSL

- Saisir ensuite un nom d'utilisateur et échanger quelques messages.

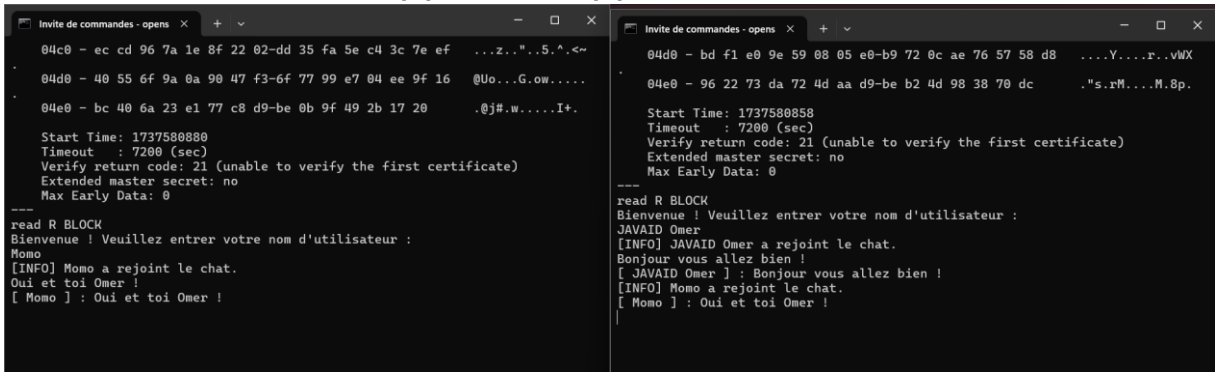

Figure 3 - Tests multi-utilisateurs avec de nouveaux clients rejoignant le chat, affichant leurs messages synchronisés

3. **Résultat Attendu** : La négociation TLS aboutit, et on peut communiquer avec le serveur.

3.3 Interface Client Swing : Tests Multi-Utilisateurs

1. **Objectif** : Vérifier que plusieurs clients peuvent se connecter simultanément et échanger des messages.
2. **Étapes** :
   - Lancer deux (ou plusieurs) fois SSLChatClientGUI.
   - Saisir des noms (ex. : « Abdel », « Mustafa », « Momo »).
   - Envoyer des messages et observer leur diffusion sur chaque instance.

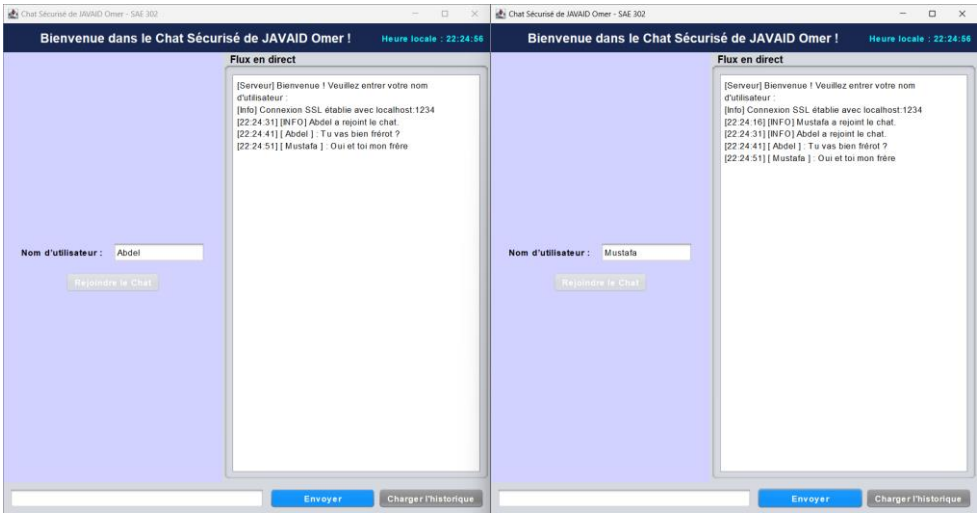

Figure 4 - Deux clients connectés simultanément à l'interface Swing, affichant les messages en temps réel avec horodatage.

3. **Résultat Attendu** : Les messages de l'un apparaissent sur tous les autres clients et le serveur logge les connexions et messages.

3.4 Vérification de l'Insertion en Base MySQL

1. **Objectif** : Confirmer que chaque message est inséré dans la table messages avec user_id, message et timestamp.
2. **Étapes** :
   - Dans MySQL Workbench, exécuter USE chat_secure; SELECT * FROM messages :

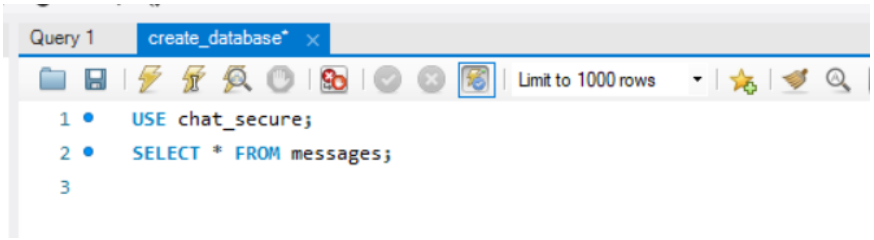

Figure 5 - Commandes SQL pour afficher la table "message"

   - Comparer le contenu avec les messages envoyés depuis l'interface.

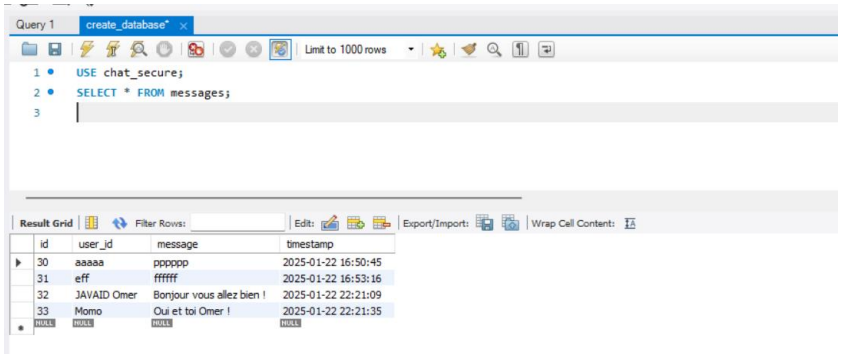

*Figure 6 - Résultat de la commande SELECT * FROM messages en concordance avec messages émis de la **Figure 3***

SAE 302-Développer des applications communicantes

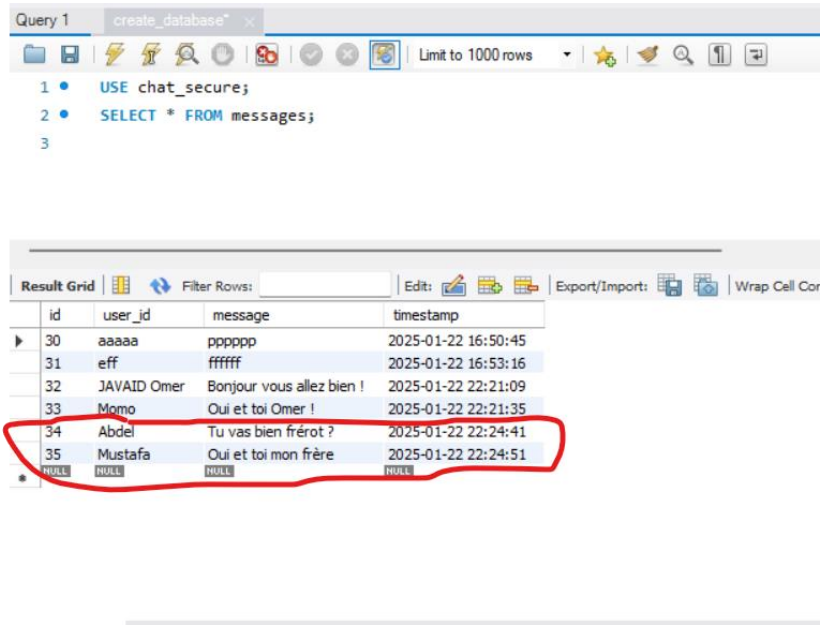

*Figure 7 - Résultat de la commande SELECT * FROM messages en concordance avec messages émis de* ***la Figure 4***

3. **Résultat Attendu** : Les lignes contiennent le nom d'utilisateur, le texte exact, et un horodatage correct.

3.5 Fonction « Charger l'Historique »

1. **Objectif** : Vérifier que l'utilisateur peut récupérer tous les anciens messages en base et les afficher dans la zone de chat.
2. **Étapes** :
   - Cliquer sur « Charger l'historique » dans SSLChatClientGUI.
   - Vérifier que les messages stockés (y compris les plus anciens) s'affichent dans la zone.

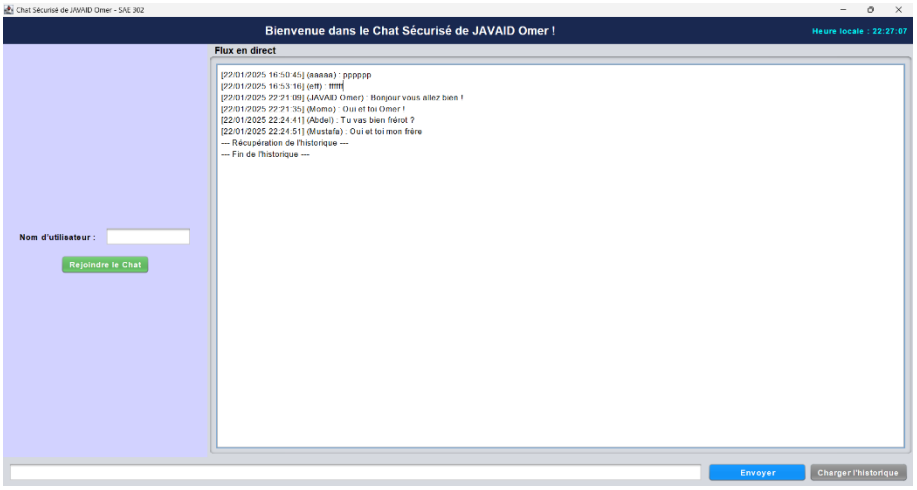

Figure 8 - Affichage de l'historique des messages récupérés depuis la base MySQL dans l'interface graphique Swing

3. **Résultat Attendu** : Le chat affiche la mention « --- Récupération de l'historique --- », puis toutes les lignes existantes.

3.6 Confirmation du Chiffrement (Wireshark)

1. **Objectif** : Vérifier que le trafic réseau est chiffré (SSL/TLS) et que les messages ne sont pas visibles en clair.
2. **Étapes** :
    - Ouvrir Wireshark, filtrer sur tcp.port == 1234 && tls.
    - Démarrer la capture, puis échanger des messages dans le chat.
    - Observer les paquets TLSv1.2 ou TLSv1.3.
3. **Résultat Attendu** : On voit des trames de type *"TLSv1.3 Application Data"* ou *"TLSv1.2 Application Data"*, dont le contenu (payload) est chiffré.

SAE 302-Développer des applications communicantes

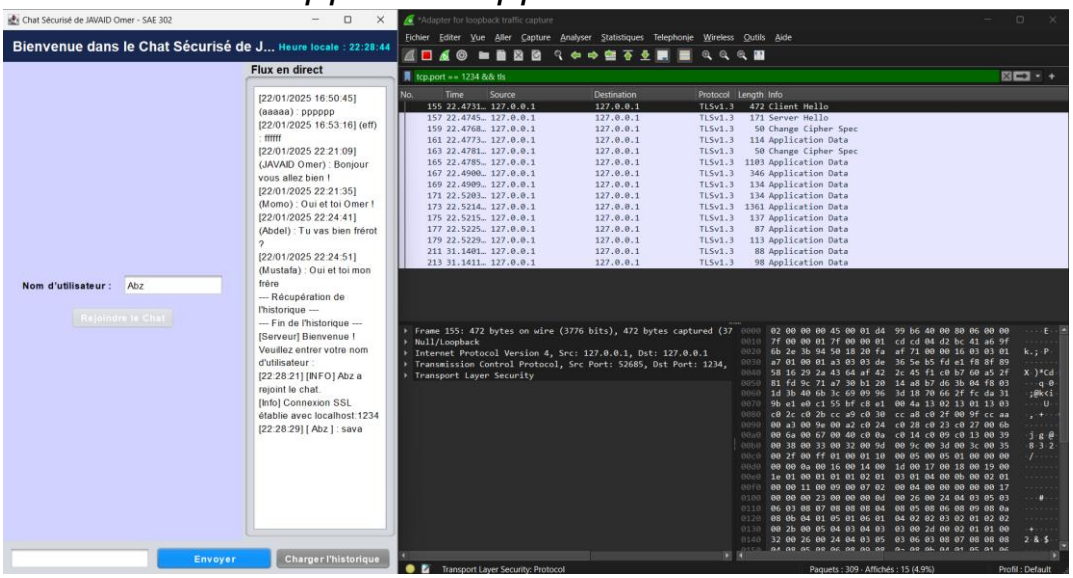

Figure 9 - Capture Wireshark montrant le flux réseau TLSv1.3 suite a la connexion d'un nouvel utilisateur « abz » suivi de messages.

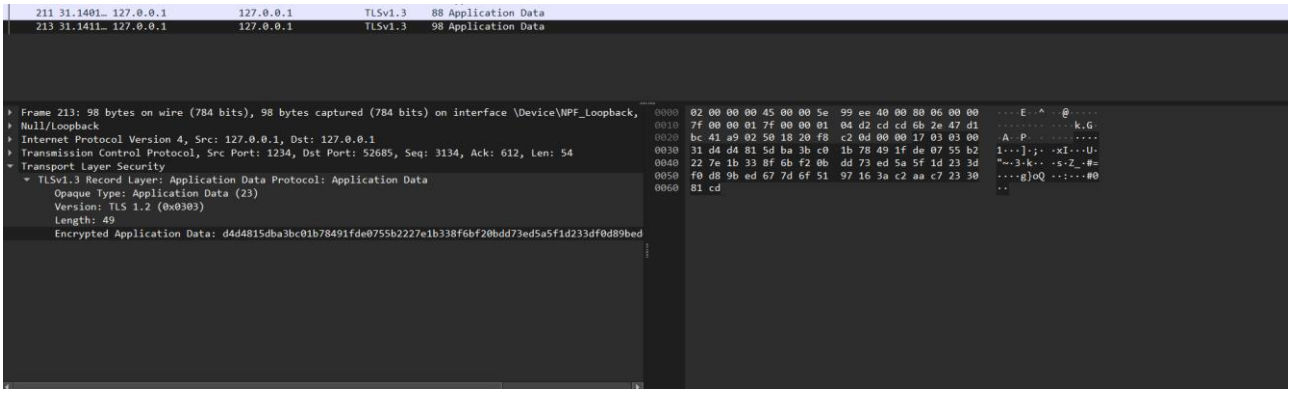

Figure 10 - Confirmant le chiffrement des données via SSL/TLS.

4. Résultats Synthétisées

Résultats Des Tests Du Projet Chat Sécurisé (Final)

	Test	Résultat Attendu	Résultat Observé	Statut
1	Serveur SSL	Affiche « Serveur chat SSL lancé… »	OK	Réussi
2	Connexion OpenSSL	Négociation TLS, session chiffrée	OK	Réussi
3	Multi-clients	Chaque client reçoit les messages en temps réel	OK	Réussi
4	Insertion MySQL	Table messages peuplée correctement	OK	Réussi
5	Historique	Bouton « Charger l'historique » affiche tous les anciens messages	OK	Réussi
6	Capture Wireshark	Paquets TLS chiffrés, contenu non lisible en clair	OK (TLS 1.3)	Réussi
7	Nouvel utilisateur "Abz"	Connexion dynamique, envoi de message, logs côté serveur	OK	Réussi

Tableau 1 - Tableau récapitulatif des résultats des tests du projet Chat Sécurisé.

5. Conclusion

Les différents scénarios de test montrent que :

- Le **chiffrement** via SSL/TLS fonctionne correctement (vérifié par OpenSSL et Wireshark).
- Les **échanges** entre clients sont synchronisés, quel que soit le nombre de participants.
- Les **messages** sont correctement **horodatés et stockés** en base MySQL, puis récupérés via l'interface (historique).

- L'**interface Swing** offre un retour clair à l'utilisateur (connexion, chat en direct, historique).

Ces résultats **valident** la conformité de l'application vis-à-vis des exigences de la SAE 302 : sécurisation, persistance, et interface graphique conviviale.