

**SECURISATION DE FORMULAIRES DE CONNEXION ET DE CONTACT**

Réalisé par : FOBASSO FOKOU Carole



Table des matières

[INTRODUCTION 3](#_Toc197985426)

[I. Contexte et objectifs 3](#_Toc197985427)

[II. Mise en œuvre 3](#_Toc197985428)

[III. Test 1 : Analyse des cookies via DevTools 4](#_Toc197985429)

[IV. Test 2 : Vérification de la configuration HTTPS via SSL Labs 5](#_Toc197985430)

[V. Test 3 : Résistance au XSS 7](#_Toc197985431)

[CONCLUSION 8](#_Toc197985432)

# INTRODUCTION

Dans ce travail pratique, nous avons conçu une application web minimale offrant une page de connexion et un formulaire de contact sécurisés. L’enjeu était de déployer un serveur HTTPS local, d’intégrer Google reCAPTCHA, de protéger les sessions et les cookies, et de stocker de manière simple les données utilisateur et les messages sans exposition à des vulnérabilités courantes (XSS, mauvaises configurations TLS).

## Contexte et objectifs

L’objectif de ce TP était de mettre en place un formulaire de connexion et un formulaire de contact protégés par :

* Authentification par session Express
* reCAPTCHA v2
* HTTPS complet (certificat self-signed pour le développement)
* Journalisation des accès et des erreurs
* Stockage local des utilisateurs et des messages

Nous avons également ajouté des headers de sécurité (Helmet et CSP) et vérifié la robustesse de la configuration TLS.

## Mise en œuvre

1. Serveur HTTPS

* Utilisation d’OpenSSL pour générer localhost.key et localhost.crt.
* Configuration d’Express pour écouter en HTTPS sur le port 3000, avec redirection automatique de HTTP vers HTTPS.

1. Gestion des sessions

* Express-session avec cookies sécurisés (Secure, HttpOnly).

1. reCAPTCHA

* Clés publique/privée injectées via le fichier .env.
* Vérification du token côté serveur avant chaque action sensible (connexion, envoi de message).

1. Stockage JSON local

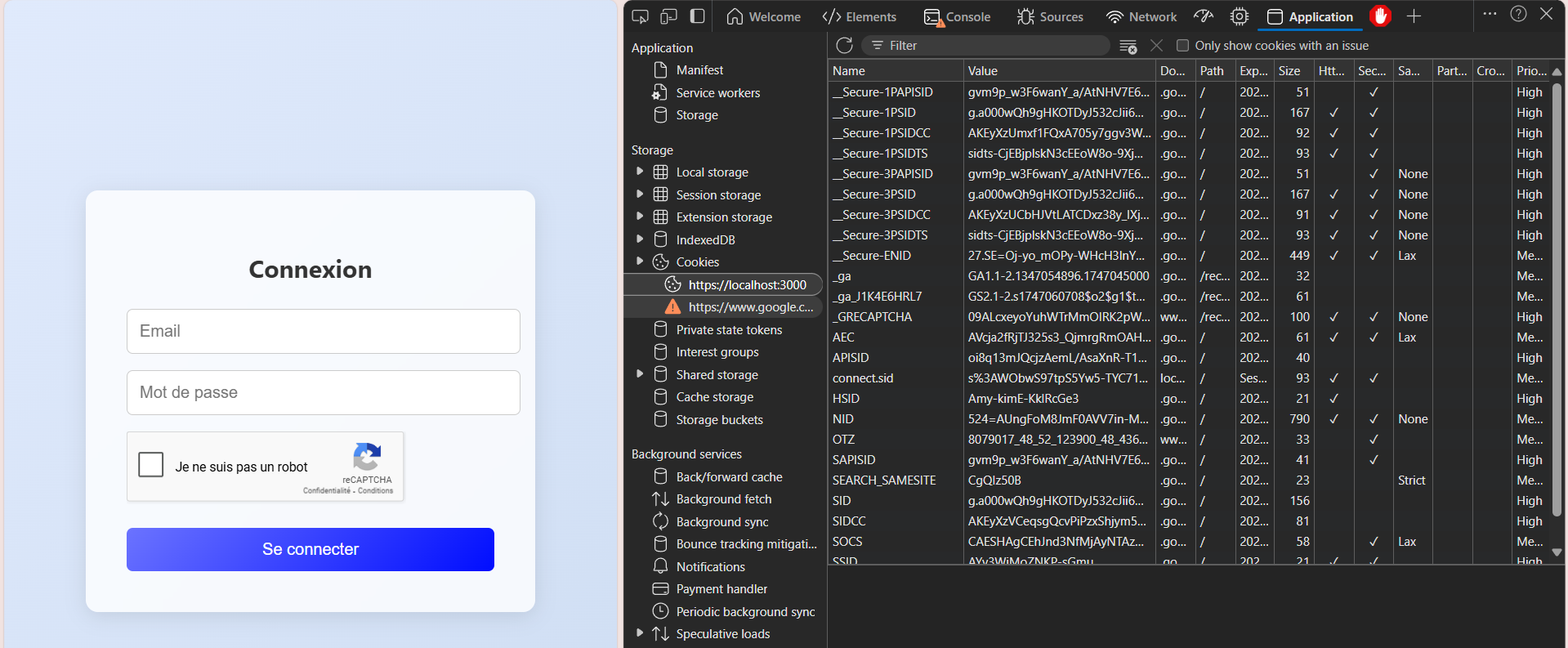
* data/users.json pour les utilisateurs (hachage bcrypt).
* data/messages.json pour conserver les messages soumis.

1. Sécurisation renforcée

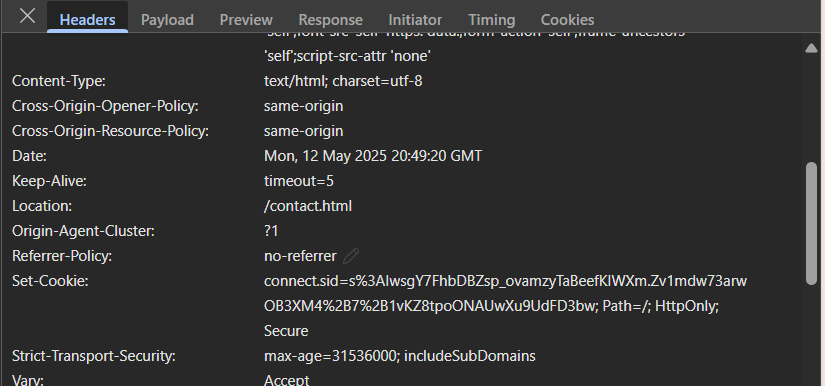
* Helmet pour appliquer un CSP strict n’autorisant que les scripts de reCAPTCHA et du domaine local.
* Validation et échappement des données côté serveur et client.

## Test 1 : Analyse des cookies via DevTools

Nous avons ouvert les outils de développement (onglet Application → Cookies) pour observer les cookies définis par notre serveur et par reCAPTCHA.



*Figure 1 : Affichage des flags « secure » des cookies*



*Figure 2 : Affichage du flag HttpOnly des cookies*

1. **Constats :**
2. connect.sid : cookie de session Express

* Attributs Secure et HttpOnly actifs → inaccessible au JavaScript côté client, transmis uniquement en HTTPS.
* SameSite Lax par défaut → protection mitigée contre CSRF tout en permettant la redirection OAuth éventuelle.

1. \_GRECAPTCHA et autres cookies Google

* Positionnés sur www.google.com pour reCAPTCHA, avec SameSite None; Secure (exigé pour iframe inter-domaines).
* Absence de cookies non sécurisés (aucun cookie transmis en clair ou sans l’attribut Secure).

1. **Conclusion :** La politique de cookie côté serveur est conforme aux bonnes pratiques de sécurité par défaut.

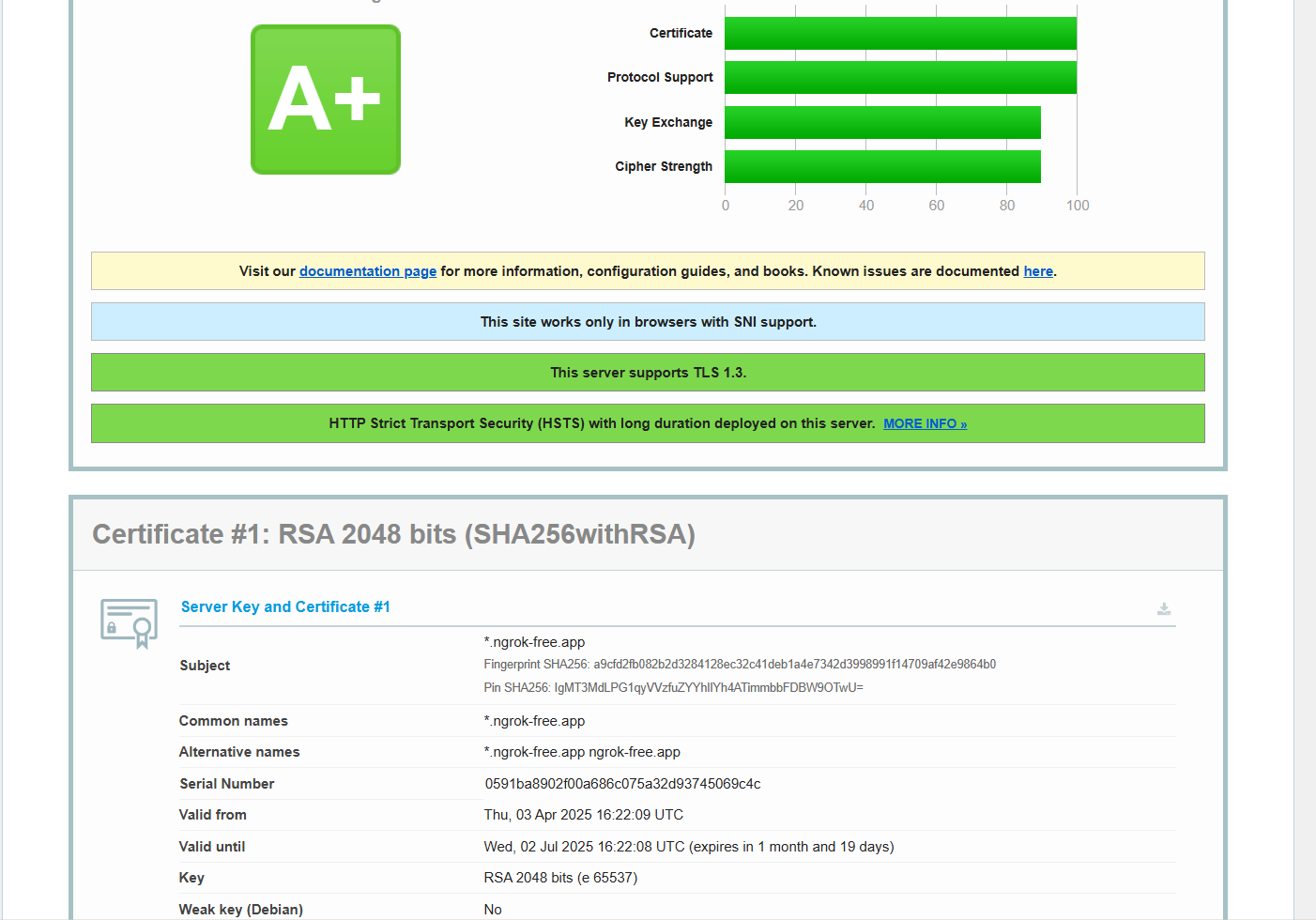
## Test 2 : Vérification de la configuration HTTPS via SSL Labs

Pour cela, nous avons utilisé l’outil Qualys SSL Labs. Il n’était cependant pas possible de l’utiliser avec notre site en local. Nous avons donc dû utiliser ngrok, pour avoir un lien nous permettant de tester notre site avec SSL Labs.

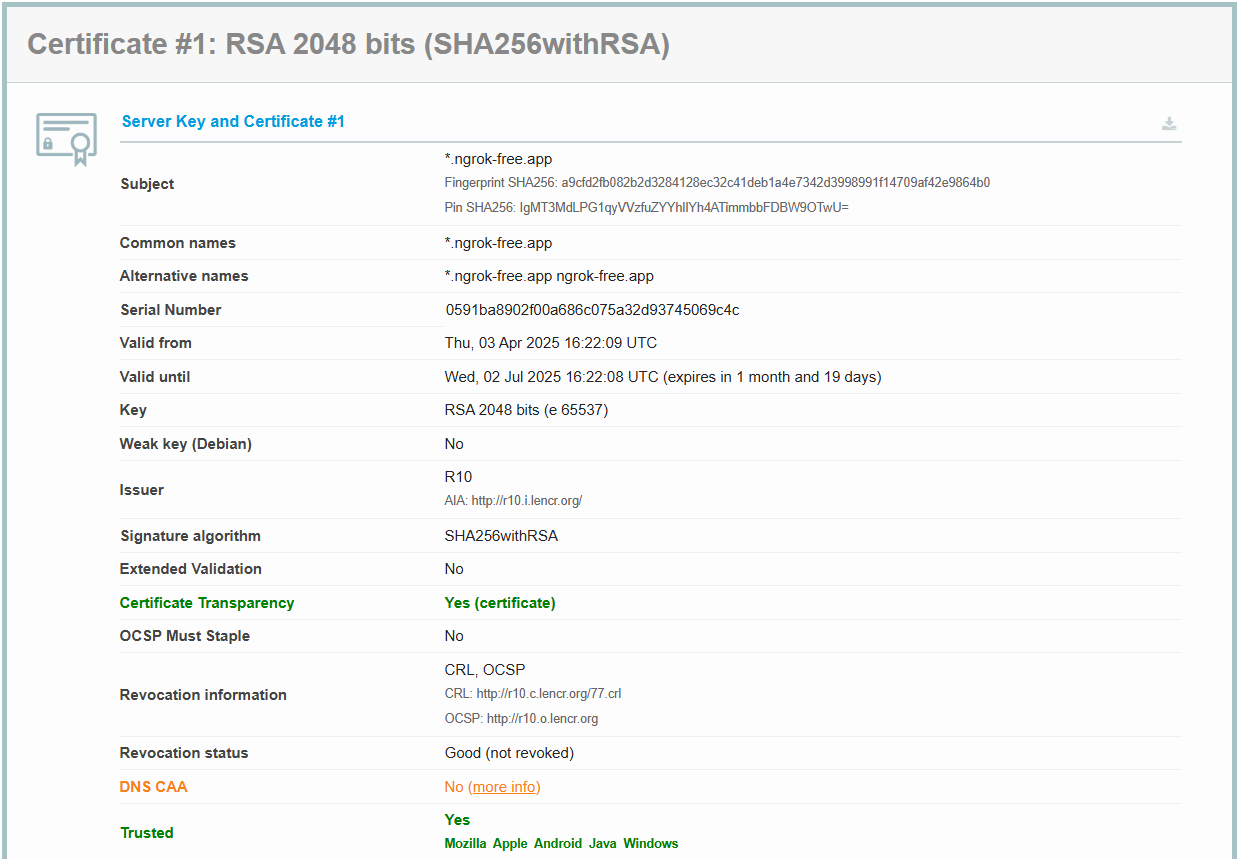
1. **Exposition temporaire du serveur local en HTTPS public à l’aide de ngrok** :

ngrok http https://localhost:3000

1. **Soumission de l’URL ngrok à SSL Labs (https://www.ssllabs.com/ssltest/)**
2. **Résultat de l’analyse : A+**



*Figure 3 : Résultat de l’analyse du site avec Qualys SSL Labs*



*Figure 4 : Suite du résultat*

1. **Points forts soulignés par SSL Labs**

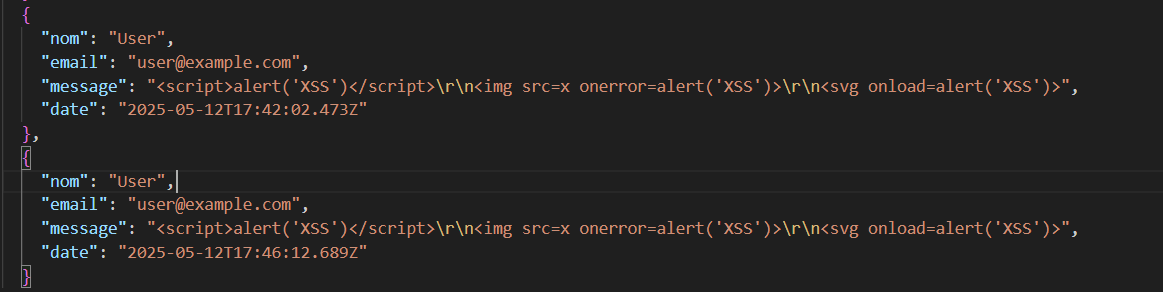
* Support de TLS 1.3
* HSTS Activé
* Certificat valide et approuvé
* Révocation vérifiée

En conclusion, on peut dire que la configuration TLS/HTTPS répond aux recommandations de sécurité actuelles.

## Test 3 : Résistance au XSS

Nous avons tenté d’injecter des payloads JavaScript dans les champs de formulaire (par ex. <script>alert('XSS')</script>).

**Résultat** : les scripts apparaissent littéralement comme texte dans messages.json et, lorsqu’on consulte la page d’affichage des messages, ils ne s’exécutent pas du tout.



*Figure 5 : Affichage du script injecté dans le fichier de stockage des messages*

**Mécanisme** : Toutes les valeurs sont échappées avant injection dans le DOM, et la CSP empêche l’exécution de scripts inline non autorisés.

# CONCLUSION

L’application répond à toutes les exigences : login sécurisé, reCAPTCHA, HTTPS strict, journalisation et stockage local. Les tests manuels et automatiques démontrent une bonne hygiène de sécurité : cookies durcis, CSP, chiffrement solide. Le site a également obtenu la note A+ sur SSL Labs et nous avons vérifié l’absence de cookies vulnérables.

Le TP illustre une approche « Security by Design » pour un formulaire web sans base de données lourde, tout en respectant les standards actuels de sécurité.