Engineer

TLS

protocole de connexion authentifiée et chiffrée

Ce que c’est

Transport Layer Security, abrégé TLS, est le protocole qu’un client et un serveur utilisent pour protéger leurs données applicatives contre l’écoute et les modifications inaperçues, et pour authentifier l’autre extrémité. Habituellement, le client authentifie le serveur ; l’authentification du client est facultative. Les menus des produits et les vendeurs de certificats l’appellent encore SSL, du nom du protocole Secure Sockets Layer qu’il a remplacé. La RFC 7568 a déprécié la dernière version de SSL en juin 2015 : ce qui porte aujourd’hui l’étiquette SSL utilise donc TLS en réalité, ou devrait le faire.

TLS se place entre un protocole applicatif et le transport sous-jacent. HTTPS est HTTP transporté dans une connexion TLS ; la même couche protège la soumission de courriels, IMAP, les connexions aux bases de données et le protocole Gemini. HTTP/3 va plus loin : il fonctionne sur QUIC, qui intègre la négociation TLS 1.3 au transport lui-même au lieu de la superposer à TCP, comme le décrit la RFC 9001.

La version actuelle est TLS 1.3. La RFC 9846, publiée en juillet 2026, actualise sa spécification sans changer le numéro de version ; elle remplace la RFC 8446 d’août 2018. Une connexion ordinaire fondée sur un certificat commence par une négociation. Le client envoie un ClientHello énumérant les versions, suites cryptographiques et valeurs d’échange de clés qu’il prend en charge, généralement avec Server Name Indication, ou SNI, le nom d’hôte souhaité, afin qu’une même adresse puisse servir plusieurs sites. Le serveur choisit et envoie son ServerHello. Les messages de négociation suivants sont chiffrés, y compris le certificat présenté. Cette chaîne de certificats relie la clé du serveur à une autorité de certification déjà approuvée par le client. Le client vérifie la chaîne, le nom et les dates ; le serveur prouve qu’il détient la clé privée correspondante en signant la négociation. Une connexion reprise peut plutôt s’authentifier avec une clé prépartagée établie auparavant.

Trois points distinguent 1.3 de 1.2. Une négociation complète ordinaire prend un aller-retour au lieu de deux, même si une nouvelle tentative peut en ajouter un. L'établissement de clés fondé sur des clés publiques assure la confidentialité persistante : un enregistrement du trafic d'aujourd'hui reste protégé même si la clé privée à long terme du serveur fuit l'an prochain. Le mode utilisant seulement une clé prépartagée fait exception et n'assure pas cette confidentialité persistante. Enfin, un client qui revient peut envoyer des « données précoces » dans son tout premier message, ce qu'on appelle 0-RTT, avant la fin de la négociation. Le prix est explicite dans la RFC : ces données ne bénéficient ni de la confidentialité persistante ni d'une protection contre leur rejeu sur une autre connexion.

Utilisez le concept pour prendre une décision

Utilisez « Que signifie TLS? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des comptes, mots de passe et droits attribués selon les besoins exacts de chaque poste, puis retirés le jour du départ.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 1 000 règles RBAC pour sécuriser Linux — 1,000 règles d'accès écrites pour combler une faille de sécurité

1 000 règles RBAC pour sécuriser Linux Contactez-nous →

Pourquoi c'est important

L'utilisation de TLS est acquise. Les décisions restantes portent sur les versions acceptées et sur l'endroit où se termine la connexion chiffrée.

D'abord les versions. La RFC 8996, publiée en mars 2021, interdit l'utilisation de TLS 1.0 et 1.1. La BCP 195 comprend la RFC 9325 et ses mises à jour de juillet 2026. La RFC 9852 impose aux nouveaux protocoles utilisant TLS de prendre en charge 1.3 et de l'utiliser par défaut ; elle autorise 1.2 comme option supplémentaire, non choisie par défaut, pour l'interopérabilité. Ce n'est pas une obligation générale de déployer uniquement 1.3. La RFC 10015 interdit aussi l'échange de clés RSA et l'échange Diffie–Hellman sur corps fini dans TLS 1.2, y compris le DH éphémère sur corps fini ; cela n'interdit pas les signatures RSA des certificats. Pour un service existant qui doit accueillir d'anciens clients, ne conservez aux côtés de 1.3 qu'une version 1.2 configurée de manière sûre, avec ECDHE et un chiffrement authentifié tel qu'AES-GCM. Un service limité à 1.3 présente une surface d'attaque plus petite et une configuration plus courte, mais exclut tout client qui ne parle que 1.2 ou moins : ancien téléphone, terminal de

paiement ou intégration d'un partenaire sur un ancien environnement Java. Consignez la version négociée pendant quelques semaines, puis décidez selon les chiffres plutôt que selon ce qui paraît moderne.

Ensuite, la terminaison. À l'endroit où TLS se termine se trouve la clé privée, et le trafic y redevient du texte en clair. Trois emplacements sont courants :

- **Un CDN ou un répartiteur de charge cloud.** Le fournisseur détient la clé, ou un certificat qu'il a émis lui-même, absorbe le coût des négociations et voit chaque requête en clair. Le trajet jusqu'au serveur d'origine doit lui aussi être protégé par TLS ; sinon, les données traversent le réseau sans chiffrement.
- **Un proxy inverse sur l'hôte**, comme Nginx, Apache, HAProxy ou Caddy. Un processus détient les clés de tous les sites de la machine et transmet les requêtes en clair aux applications derrière lui, par l'interface de bouclage ou un socket local.
- **L'application elle-même.** Chaque service détient sa propre clé. Cela convient au trafic entre services et au TLS mutuel, où le client présente aussi un certificat, mais multiplie les endroits où renouveler les certificats.

La plupart des infrastructures combinent les trois. L'essentiel est que quelqu'un puisse dire, pour chaque saut, où se trouve le texte en clair et qui pourrait le lire.

À quoi cela ressemble en pratique

La réalisation des services publics de l'entreprise en septembre 2026 montre le peu de moyens nécessaires à la terminaison TLS. Elle utilisait une instance cloud à un cœur avec 512 Mo de mémoire, dont environ 464 Mo utilisables, et 10 Go de disque. Sur cette machine, le serveur web terminait TLS pour sept domaines, à côté de la forge Git, d'un service onion Tor, d'un serveur Gemini, d'un serveur Gopher et du blocage des intrusions. L'automatisation des sauvegardes était réalisée séparément ; cela n'établissait pas que des sauvegardes chiffrées étaient actives. C'était la terminaison sur l'hôte sous sa forme la plus simple : un serveur web répondait pour les sept sites, sans machine ni service supplémentaire pour leur chiffrement. La conception traitait la mémoire comme la contrainte déterminante, et le déploiement consigné tenait dans ces 464 Mo.

La règle suivie par l'entreprise figure sur sa page de sécurité : chiffrement en transit partout, certificats émis et renouvelés automatiquement et renouvellement supervisé. La raison donnée est que TLS échoue le plus souvent

à cause d'un certificat que personne ne surveillait. Les échecs ci-dessous sont ceux qui restent une fois le renouvellement automatisé.

Où les problèmes surviennent

Certains clients échouent à négocier alors que les navigateurs se connectent. Les deux extrémités n'ont aucune version ou suite cryptographique commune. Quelqu'un a limité le serveur à 1.3 ou à une courte liste de suites qu'un ancien client ne propose pas ; ou un serveur ne proposant encore que des suites CBC rencontre un client qui les a abandonnées. L'alerte indique « handshake failure » ou « protocol version », sans guère de précisions. Testez chaque version depuis l'extérieur :

```
openssl s_client -connect example.com:443 \
    -servername example.com -tls1_2 </dev/null
```

Consignez la version et la suite négociées avant de réduire les possibilités. Sur un service existant dont les clients ont encore besoin de 1.2, conservez 1.2 configuré de manière sûre avec ECDHE et AES-GCM à côté de 1.3, tout en préparant la migration. Pour un nouveau protocole utilisant TLS, prenez en charge 1.3 et choisissez-le par défaut ; n'activez 1.2 que si l'interopérabilité l'exige.

Les navigateurs approuvent le certificat, mais les scripts, curl et les applications mobiles le rejettent. Le serveur envoie son certificat sans l'intermédiaire qui le relie à la racine. Les navigateurs comblent souvent le manque avec des intermédiaires déjà rencontrés ou récupèrent eux-mêmes celui qui manque ; la plupart des bibliothèques TLS ne font ni l'un ni l'autre et signalent un émetteur inconnu. Configurez le fichier de chaîne complète, avec le certificat du serveur d'abord, puis les intermédiaires, et testez avec un client sans cache : la ligne `openssl s_client` ci-dessus affiche la chaîne réellement envoyée par le serveur.

La vérification du certificat est désactivée « pour l'instant », et ce réglage est livré. Un serveur de test a un certificat autosigné, ou un proxy d'entreprise signe à nouveau le trafic, et quelqu'un ajoute `verify=False`, `curl -k`, `InsecureSkipVerify` ou `NODE_TLS_REJECT_UNAUTHORIZED=0`. La connexion reste chiffrée, mais avec quiconque y répond : un intermédiaire peut présenter son propre

certificat et tout lire. Placez le bon certificat d'autorité dans le magasin de confiance du client, ou indiquez-lui explicitement où le trouver, et faites échouer la CI sur ces options.

Certains visiteurs reçoivent le certificat d'un autre site. Le serveur présente son certificat par défaut chaque fois qu'un ClientHello nomme un hôte sans configuration correspondante, ou n'en nomme aucun. Cela arrive lorsqu'un nouveau domaine est dirigé vers l'adresse avant que son bloc de serveur existe, ou lorsqu'une sonde de supervision ou un ancien client n'envoie pas de SNI. Le navigateur signale un nom qui ne correspond pas et, sur un hôte partagé, révèle aussi au visiteur les autres sites hébergés là. Donnez à chaque nom public son bloc et un certificat qui le couvre, puis faites refuser la négociation au bloc par défaut au lieu de proposer le certificat d'un autre site.

Un paiement, ou toute autre requête, est exécuté deux fois. Le point de terminaison TLS accepte les données précoces 0-RTT, une requête POST y arrive, puis les mêmes octets sont renvoyés sur une nouvelle connexion, par un attaquant ou lors d'une nouvelle tentative d'envoi sur le réseau. TLS 1.3 autorise ces données, mais ne peut garantir la protection contre le rejeu entre connexions. Désactivez-les, attendez la fin de la négociation avant d'agir sur elles, ou rejetez les requêtes qui ne peuvent pas être rejouées sans risque avec le statut 425 Too Early, comme le prévoit la RFC 8470. Une passerelle doit aussi transmettre l'indication Early-Data et savoir que son serveur d'origine la comprend : l'attente sur un saut ultérieur n'efface pas le risque de rejeu associé à un saut précédent.

Pour aller plus loin

- RFC 9846: protocole Transport Layer Security (TLS), version 1.3 — la spécification de juillet 2026, qui remplace la RFC 8446 sans changer le numéro 1.3 ; négociation, enregistrements et limites des données précoces.
- RFC 9325: recommandations pour une utilisation sûre de TLS et DTLS — les recommandations de déploiement de la BCP 195, à lire avec les mises à jour suivantes.
- RFC 9852: les nouveaux protocoles utilisant TLS doivent imposer TLS 1.3 — prise en charge obligatoire et choix par défaut de 1.3 pour les nouveaux protocoles, avec interopérabilité facultative en 1.2 ; la mise à jour ne s'applique pas à DTLS.

- RFC 10015: dépréciation des méthodes obsolètes d’échange de clés dans TLS 1.2 et DTLS 1.2 — interdiction des échanges RSA et DH sur corps fini, à distinguer des signatures de certificats et de l’établissement de clés en TLS 1.3.
- RFC 8996: dépréciation de TLS 1.0 et TLS 1.1 — la fin officielle des deux anciennes versions et ses raisons.
- RFC 8470: utilisation des données précoces en HTTP — en-tête Early-Data, coordination des passerelles et statut 425.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Que signifie TLS? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/tls/