Engineer

TLS

🤝 протокол автентифікованих, зашифрованих з'єднань

Що це таке

TLS, Transport Layer Security, — це протокол, за допомогою якого клієнт і сервер захищають дані застосунку від прослуховування й непомітних змін та автентифікують іншу сторону. Зазвичай клієнт автентифікує сервер; автентифікація клієнта необов'язкова. У меню продуктів і в магазинах сертифікатів його досі часто називають SSL, за назвою Secure Sockets Layer, протоколу, на зміну якому прийшов TLS. RFC 7568 у червні 2015 року визнав останню версію SSL застарілою, тож усе, що сьогодні зветься SSL, усередині є TLS або принаймні має ним бути.

TLS розташований між протоколом застосунку й транспортом під ним. HTTPS — це HTTP усередині TLS-з'єднання, а той самий шар захищає надсилання пошти, IMAP, з'єднання з базами даних і протокол Gemini. HTTP/3 іде далі: він працює поверх QUIC, що вбудовує рукостискання TLS 1.3 просто в транспорт, а не кладе його поверх TCP, як описує RFC 9001.

Чинна версія — TLS 1.3. RFC 9846, опублікований у липні 2026 року, оновлює її специфікацію, не змінюючи номера версії; він замінює RFC 8446 від серпня 2018 року. Звичайне з'єднання на основі сертифікатів починається з рукостискання. Клієнт надсилає ClientHello з переліком версій, наборів шифрів і ключових часток, які він підтримує, зазвичай разом із Server Name Indication (SNI) — іменем потрібного хоста, щоб одна адреса могла обслуговувати багато сайтів. Сервер обирає й надсилає свій ServerHello. Подальші повідомлення рукостискання зашифровані, зокрема й сертифікат, який він показує. Ланцюжок сертифікатів пов'язує ключ сервера з центром сертифікації, якому клієнт уже довіряє. Клієнт перевіряє ланцюжок, ім'я та дати, а сервер доводить, що володіє відповідним закритим ключем, підписуючи рукостискання. Відновлене з'єднання натомість може автентифікуватися за допомогою попередньо встановленого спільного ключа.

Версію 1.3 від 1.2 відрізняють три речі. Звичайне повне рукостискання займає один обмін туди й назад замість двох, хоча повторна спроба може додати ще один. Узгодження ключів на основі відкритих ключів забезпечує пряму секретність: запис сьогоднішнього трафіку лишається закритим, навіть якщо довгостроковий закритий ключ сервера витече наступного року. Режим лише з попередньо встановленим спільним ключем — виняток: прямої секретності він не забезпечує. І клієнт, що повертається, може надіслати «ранні дані» вже в першому повідомленні — це називають 0-RTT, — до завершення рукостискання, але RFC прямо називає ціну: такі дані не мають ні прямої секретності, ні захисту від повторного відтворення в іншому з'єднанні.

Застосуйте поняття до рішення

Застосуйте «Що таке TLS?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Облікові записи, паролі й права налаштовані так, щоб доступ відповідав роботі, а звільнені втрачали його того ж дня.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Написала 1 000 правил RBAC для Linux-серверів — 1,000 правил доступу написано, щоб усунути вразливість

Написала 1 000 правил RBAC для Linux-серверів Зв'язатися з нами →

Чому це важливо

Чи використовувати TLS, давно вирішено. Відкритими лишаються два питання: які версії приймати і де закінчується зашифроване з'єднання.

Спершу версії. RFC 8996 від березня 2021 року забороняє використовувати TLS 1.0 і 1.1. BCP 195 охоплює RFC 9325 та його оновлення від липня 2026 року. RFC 9852 вимагає, щоб нові протоколи, які використовують TLS, підтримували 1.3 і вибирали його за замовчуванням; 1.2 дозволено як додатковий варіант для сумісності, який не використовується за замовчуванням. Це не загальна вимога розгортати лише 1.3. RFC 10015 також забороняє обмін ключами RSA та обмін Діффі–Геллмана над скінченним полем у TLS 1.2, зокрема ефемерний; це не забороняє підписів RSA на сертифікатах. Для наявного сервісу, якому потрібні старіші клієнти, поряд із 1.3 слід залишати лише безпечно налаштований 1.2 з ECDHE та автентифікованим шифруванням, наприклад AES-GCM. Сервіс лише з 1.3 має меншу поверхню атаки й коротшу конфігурацію, але відсікає кожного клієнта, що вміє щонайбільше 1.2: старіший телефон, платіжний

термінал, інтеграцію партнера на старому середовищі Java. Кілька тижнів журналюйте узгоджену версію, а тоді вирішуйте за цифрами, а не за тим, що звучить сучасно.

Потім термінування. Там, де закінчується TLS, лежить закритий ключ, і трафік знову стає відкритим текстом. Є три поширені місця:

- **CDN або хмарний балансувальник навантаження.** Постачальник тримає ключ або сертифікат, який сам випустив, бере на себе навантаження рукостискань і бачить кожен запит відкритим текстом. Ділянці від нього до вихідного сервера потрібен власний TLS, інакше вона йде мережею незашифрованою.
- **Зворотний проксі на хості,** наприклад Nginx, Apache, HAProxy чи Caddy. Один процес тримає ключі всіх сайтів машини й передає запити без шифрування застосункам за ним через loopback або локальний сокет.
- **Сам застосунок.** Кожен сервіс має власний ключ. Це пасує трафіку між сервісами та взаємному TLS (mutual TLS), де сертифікат показує й клієнт, і водночас множить місця, де сертифікат треба поновлювати.

Більшість інфраструктур поєднує всі три. Важливо, щоб для кожної ділянки хтось міг сказати, де лежить відкритий текст і хто здатен його прочитати.

Як це виглядає на практиці

Реалізація публічних сервісів компанії від вересня 2026 року показує, як мало потребує термінування TLS. Вона використовувала хмарний інстанс з одним ядром, 512 МБ пам’яті, з яких придатними були близько 464 МБ, і 10 ГБ диска. На ньому вебсервер термінував TLS для семи доменів поруч із git-форджем, onion-сервісом Tor, сервером Gemini, сервером Gopher і блокуванням вторгнень. Автоматизацію резервного копіювання було реалізовано окремо; це не підтверджувало, що зашифровані резервні копії активно створювалися. Це було термінування на хості в найпростішому вигляді: один вебсервер відповідав за всі сім сайтів, і їхнє шифрування не потребувало ані додаткової машини, ані додаткового сервісу. У схемі пам’ять вважали визначальним обмеженням, а задокументоване розгортання працювало в межах цих 464 МБ.

Правило, за яким працює компанія, записане на її сторінці безпеки: шифрування під час передавання всюди, а сертифікати випускаються й поновлюються без участі людини, і моніторинг стежить за кожним

поновленням. Пояснення там таке: найпоширеніша причина збоїв TLS — сертифікат, строк якого спливає непомітно. Нижче зібрано збої, що лишаються, коли поновлення вже працює саме.

Де виникають проблеми

Частина клієнтів обривається з помилкою рукостискання, хоча браузери під'єднуються без проблем. Сторони не мають спільної версії чи набору шифрів. Хтось звузив сервер до 1.3 або до короткого списку наборів, і старішому клієнту нема чого запропонувати; або сервер, що досі пропонує лише набори CBC, натрапляє на клієнта, який від них відмовився. Повідомлення звучить як «handshake failure» чи «protocol version» і майже нічого не пояснює. Перевірте кожну версію ззовні:

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

Журналюйте узгоджені версію й набір шифрів, перш ніж щось звужувати. На наявному сервісі, клієнтам якого досі потрібен 1.2, тримайте безпечно налаштований 1.2 з ECDHE та AES-GCM поруч із 1.3, плануючи перехід. Для нового протоколу, який використовує TLS, підтримуйте 1.3 і вибирайте його за замовчуванням; умикайте 1.2 лише тоді, коли цього потребує сумісність.

Браузери довіряють сертифікату, а скрипти, curl і мобільні застосунки його відхиляють. Сервер надсилає власний сертифікат без проміжного, що пов'язує його з кореневим. Браузери часто заповнюють прогалину проміжними сертифікатами, які вже бачили, або самі завантажують відсутній; більшість бібліотек TLS не робить ні того, ні іншого й повідомляє про невідомого видавця. Налаштуйте повний файл ланцюжка — спочатку сертифікат сервера, потім проміжні — і перевіряйте клієнтом без кешу: рядок з `openssl s_client` вище показує ланцюжок, який сервер справді надіслав.

Перевірку сертифікатів вимикають «тимчасово», і це налаштування потрапляє в робоче середовище. Тестовий сервер із самопідписаним сертифікатом або корпоративний проксі, що перепідписує трафік, — і хтось додає `verify=False`, `curl -k`, `InsecureSkipVerify` чи `NODE_TLS_REJECT_UNAUTHORIZED=0`. З'єднання лишається зашифрованим, але з будь-ким, хто відповість: будь-хто посередині може показати власний

сертифікат і прочитати все. Додайте потрібний сертифікат центру сертифікації до сховища довіри клієнта або вкажіть його клієнту явно, а в CI зробіть ці прапорці причиною падіння збірки.

Частина відвідувачів отримує сертифікат зовсім іншого сайту. Сервер показує типовий сертифікат, коли ClientHello називає хост, для якого в нього немає конфігурації, або взагалі не називає хоста. Так буває, коли новий домен спрямовують на адресу раніше, ніж з'явився його серверний блок, і коли зонд моніторингу чи старий клієнт не надсилає SNI. Браузер показує помилку імені, а на спільному хості заодно повідомляє відвідувачу, які ще сайти там живуть. Дайте кожному публічному імені власний блок і сертифікат, що його покриває, а типовий блок нехай відхиляє рукостискання замість показувати чужий сертифікат.

Платіж чи інший запит виконується двічі. Термінатор приймає ранні дані 0-RTT, усередині них приходить POST, і зловмисник або повторна спроба в мережі надсилає ті самі байти знову в новому з'єднанні. TLS 1.3 дозволяє ранні дані, але не може гарантувати захисту від повторного відтворення між з'єднаннями. Або вимкніть ранні дані, або відкладіть дії над ними до завершення рукостискання, або відхиляйте запити, які не можна безпечно повторити, статусом 425 Too Early, як визначає RFC 8470. Шлюз також має передавати ознаку Early-Data далі й знати, що його сервер-джерело її розуміє: очікування на пізнішому стрибку не усуває ризику повторного відтворення з попереднього.

Що почитати далі

- RFC 9846: The Transport Layer Security (TLS) Protocol Version 1.3 — специфікація від липня 2026 року, що замінює RFC 8446 без зміни номера версії протоколу 1.3; рукостискання, записи та обмеження ранніх даних.
- RFC 9325: Recommendations for Secure Use of TLS and DTLS — рекомендації щодо розгортання з BCP 195, які слід читати разом із наведеними далі оновленнями.
- RFC 9852: New Protocols Using TLS Must Require TLS 1.3 — обов'язкова підтримка 1.3 і його вибір за замовчуванням у нових протоколах із необов'язковою сумісністю з 1.2; оновлення не стосується DTLS.
- RFC 10015: Deprecating Obsolete Key Exchange Methods in TLS 1.2 and DTLS 1.2 — заборонені обміни ключами RSA та Діффі–Геллмана над скінченним полем, відмежовані від підписів сертифікатів і від узгодження ключів у TLS 1.3.

- RFC 8996: Deprecating TLS 1.0 and TLS 1.1 — формальне завершення двох старіших версій і його причини.
- RFC 8470: Using Early Data in HTTP — заголовок Early-Data, координація зі шлюзами та статус 425.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке TLS?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/tls/