Engineer

TLS

Protocolo de conexiones autenticadas y cifradas

Qué es

Transport Layer Security, abreviado TLS, es el protocolo que utilizan un cliente y un servidor para proteger los datos de sus aplicaciones frente a escuchas y modificaciones inadvertidas, y para autenticar al otro extremo. Normalmente el cliente autentica al servidor; la autenticación del cliente es opcional. Los menús de los productos y los vendedores de certificados siguen llamándolo SSL, por el protocolo Secure Sockets Layer al que sustituyó. RFC 7568 declaró obsoleta la última versión de SSL en junio de 2015, de modo que hoy cualquier producto etiquetado como SSL utiliza TLS por debajo, o debería hacerlo.

TLS se sitúa entre un protocolo de aplicación y el transporte que hay debajo. HTTPS es HTTP transportado dentro de una conexión TLS, y la misma capa protege el envío de correo, IMAP, las conexiones a bases de datos y el protocolo Gemini. HTTP/3 va más allá: funciona sobre QUIC, que incorpora la negociación de TLS 1.3 al propio transporte en lugar de superponerla a TCP, como describe RFC 9001.

La versión actual es TLS 1.3. RFC 9846, publicado en julio de 2026, actualiza su especificación sin cambiar el número de versión y sustituye a RFC 8446, de agosto de 2018. Una conexión normal basada en certificados empieza con una negociación. El cliente envía un ClientHello que enumera las versiones, los conjuntos de cifrado y los parámetros de intercambio de claves que admite, normalmente con Server Name Indication (SNI), el nombre del host al que quiere acceder, para que una dirección pueda servir a muchos sitios. El servidor elige y envía su ServerHello. Los mensajes posteriores de la negociación se cifran, incluido el certificado que presenta. Esa cadena de certificados vincula la clave del servidor con una autoridad de certificación en la que el cliente ya confía. El cliente comprueba la cadena, el nombre y las fechas, y el servidor demuestra

que posee la clave privada correspondiente al firmar la negociación. Una conexión reanudada puede autenticarse, en cambio, mediante una clave previamente compartida.

Tres aspectos distinguen 1.3 de 1.2. Una negociación completa normal necesita un viaje de ida y vuelta en lugar de dos, aunque un reintento puede añadir otro. El establecimiento de claves basado en criptografía de clave pública proporciona secreto hacia adelante: una grabación del tráfico de hoy sigue protegida aunque la clave privada de larga duración del servidor se filtre el año que viene. El modo que utiliza únicamente una clave precompartida es una excepción y no proporciona secreto hacia adelante. Además, un cliente que vuelve puede enviar «datos anticipados» en su primer mensaje, lo que se conoce como 0-RTT, antes de terminar la negociación, con un coste que el RFC explicita: los datos anticipados no ofrecen secreto hacia adelante ni están protegidos contra su reproducción en otra conexión.

Utilice el concepto en una decisión

Aplique «¿Qué es TLS?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Cuentas, contraseñas y permisos organizados para que cada persona tenga exactamente el acceso que necesita y lo pierda el mismo día que deja la empresa.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. 1.000 reglas RBAC para servidores Linux — 1,000 reglas de acceso escritas para cerrar una brecha de seguridad

1.000 reglas RBAC para servidores Linux Póngase en contacto →

Por qué importa

La necesidad de utilizar TLS está resuelta. Las decisiones pendientes son qué versiones aceptar y dónde termina la conexión cifrada.

Primero, las versiones. RFC 8996, publicado en marzo de 2021, establece que no deben utilizarse TLS 1.0 ni 1.1. BCP 195 incluye RFC 9325 y sus actualizaciones de julio de 2026. RFC 9852 exige que los protocolos nuevos que utilizan TLS admitan 1.3 y la utilicen por defecto; permite 1.2 como opción adicional, no predeterminada, por interoperabilidad. No es una exigencia universal de desplegar únicamente 1.3. RFC 10015 también prohíbe en TLS 1.2 el intercambio de claves RSA y el de Diffie–Hellman de campo finito, incluido DH efímero de campo finito; esto no prohíbe las firmas RSA de los certificados. En un servicio existente que necesite clientes antiguos, conserve únicamente una versión 1.2 configurada de forma segura junto a 1.3, con ECDHE y cifrado autenticado como

AES-GCM. Un servicio que solo admita 1.3 tiene una superficie de ataque menor y una configuración más breve, y excluye a todos los clientes que como máximo hablan 1.2: un teléfono antiguo, un terminal de pago o la integración de un socio que utiliza un entorno Java antiguo. Registre durante unas semanas la versión negociada y decida después a partir de las cifras, no de lo que suene moderno.

Después, la terminación. Donde termina TLS se conserva la clave privada y el tráfico vuelve a ser texto sin cifrar. Hay tres ubicaciones habituales:

- **Una CDN o un equilibrador de carga de la nube.** El proveedor conserva la clave, o un certificado que ha emitido él mismo, absorbe la carga de las negociaciones y ve cada solicitud sin cifrar. El tramo desde allí hasta el servidor de origen necesita su propio TLS o atravesará la red sin cifrado.
- **Un proxy inverso en el host,** como Nginx, Apache, HAProxy o Caddy. Un proceso conserva las claves de todos los sitios de la máquina y pasa las solicitudes sin cifrar a las aplicaciones que tiene detrás mediante la interfaz de bucle local o un socket local.
- **La propia aplicación.** Cada servicio conserva su propia clave. Esto encaja con el tráfico entre servicios y TLS mutuo, en el que el cliente también presenta un certificado, y multiplica los lugares donde hay que renovar un certificado.

La mayoría de las infraestructuras combinan las tres opciones. Lo que cuenta es que alguien pueda explicar, para cada tramo, dónde están los datos sin cifrar y quién podría leerlos.

Cómo se aplica en la práctica

La implementación de los servicios públicos de la empresa de septiembre de 2026 muestra lo poco que necesita la terminación TLS. Utilizaba una instancia en la nube de un solo núcleo, con 512 MB de memoria, unos 464 MB utilizables y 10 GB de disco. En ella, el servidor web terminaba TLS para siete dominios, junto al servidor de repositorios git, un servicio onion de Tor, un servidor Gemini, un servidor Gopher y el bloqueo de intrusiones. La automatización de las copias de seguridad se implementó por separado; eso no demostraba que las copias cifradas estuvieran activas. Era la terminación en el host en su forma más sencilla: un servidor web respondía por los siete sitios y su cifrado no requería otra máquina ni otro servicio. El diseño trató la memoria como la restricción determinante, y el despliegue registrado funcionaba dentro de esos 464 MB.

La norma de trabajo de la empresa está escrita en su página de seguridad: cifrado en tránsito en todas partes, certificados emitidos y renovados automáticamente y renovación monitorizada. La razón que da es que TLS suele fallar por un certificado que nadie vigilaba. Los fallos siguientes son los que quedan una vez que la renovación se ejecuta por sí sola.

Dónde surgen los problemas

Algunos clientes fallan durante la negociación, mientras los navegadores se conectan sin problemas. Los dos extremos no comparten ninguna versión o conjunto de cifrado. Alguien limitó el servidor a 1.3 o a una lista breve de conjuntos, y un cliente antiguo no ofrece ninguno de ellos; o un servidor que sigue ofreciendo solo conjuntos CBC se encuentra con un cliente que ya los ha eliminado. La alerta dice «handshake failure» o «protocol version» y poco más. Pruebe cada versión desde fuera:

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

Registre la versión y el conjunto negociados antes de restringir nada. En un servicio existente cuyos clientes todavía necesiten 1.2, mantenga 1.2 configurada de forma segura con ECDHE y AES-GCM junto a 1.3 mientras prepara la migración. En un protocolo nuevo que utilice TLS, admita 1.3 y utilícela por defecto; habilite 1.2 únicamente cuando la interoperabilidad lo requiera.

Los navegadores confían en el certificado, pero los scripts, curl y las aplicaciones móviles lo rechazan. El servidor envía su certificado sin el intermedio que lo vincula con la raíz. Los navegadores suelen completar el hueco con certificados intermedios que ya han visto u obtener por sí mismos el que falta; la mayoría de las bibliotecas TLS no hacen ninguna de las dos cosas y notifican un emisor desconocido. Configure el archivo de cadena completa, primero el certificado del servidor y después los intermedios, y pruebe con un cliente que no mantenga caché: la línea de `openssl s_client` anterior muestra la cadena que el servidor envió realmente.

La verificación de certificados se desactiva «de momento» y así llega a producción. Un servidor de pruebas con un certificado autofirmado o un proxy corporativo que vuelve a firmar el tráfico lleva a alguien a añadir `verify=False`, `curl -k`, `InsecureSkipVerify` o `NODE_TLS_REJECT_UNAUTHORIZED=0`. La conexión sigue

cifrada, pero con quien responda: cualquiera que se interponga puede presentar un certificado propio y leerlo todo. Incorpore el certificado de la CA correcta al almacén de confianza del cliente, o indíquele explícitamente dónde está, y haga que CI falle ante esas opciones.

Algunos visitantes reciben el certificado de otro sitio. El servidor presenta su certificado predeterminado cuando un ClientHello nombra un host para el que no tiene configuración o no nombra ninguno. Ocurre cuando un dominio nuevo apunta a la dirección antes de que exista su bloque de servidor y cuando una sonda de monitorización o un cliente antiguo no envía SNI. El navegador muestra que el nombre no coincide y, en un host compartido, también revela al visitante qué otros sitios se alojan allí. Dé a cada nombre público su propio bloque y un certificado que lo cubra, y haga que el bloque predeterminado rechace la negociación en lugar de ofrecer el certificado de otro sitio.

Un pago, o cualquier otra solicitud, se ejecuta dos veces. El terminador acepta datos anticipados 0-RTT, llega un POST dentro de ellos y un atacante o un reintento de red vuelve a enviar los mismos bytes en otra conexión. TLS 1.3 permite los datos anticipados, pero no puede garantizar protección contra su reproducción entre conexiones. Desactívelos, espere a que termine la negociación antes de actuar sobre ellos o rechace con 425 Too Early las solicitudes que no puedan reproducirse de forma segura, como especifica RFC 8470. Una pasarela también debe transmitir la indicación Early-Data y saber que el origen la entiende: esperar en un tramo posterior no elimina el riesgo de reproducción de uno anterior.

Lecturas adicionales

- RFC 9846: protocolo Transport Layer Security (TLS), versión 1.3 — especificación de julio de 2026 que sustituye a RFC 8446 sin cambiar el número de versión 1.3 del protocolo; negociación, registros y limitaciones de los datos anticipados.
- RFC 9325: recomendaciones para el uso seguro de TLS y DTLS — recomendaciones de despliegue de BCP 195, que deben leerse junto a las actualizaciones siguientes.
- RFC 9852: los protocolos nuevos que utilizan TLS deben exigir TLS 1.3 — soporte obligatorio y uso predeterminado de 1.3 en protocolos nuevos, con interoperabilidad opcional mediante 1.2; la actualización no se aplica a DTLS.

- RFC 10015: retirada de los métodos obsoletos de intercambio de claves en TLS 1.2 y DTLS 1.2 — intercambios de claves RSA y DH de campo finito prohibidos, diferenciados de las firmas de certificados y del establecimiento de claves de TLS 1.3.
- RFC 8996: retirada de TLS 1.0 y TLS 1.1 — fin formal de las dos versiones antiguas y sus motivos.
- RFC 8470: uso de datos anticipados en HTTP — cabecera Early-Data, coordinación de pasarelas y estado 425.

Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es TLS?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

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

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

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

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/tls/