Engineer

HTTPS

🔑 Las mismas peticiones mediante una conexión cifrada

Qué es

HTTPS es HTTP enviado a través de una conexión TLS. Las peticiones y respuestas son los mismos mensajes, con los mismos métodos, códigos de estado y cabeceras; cambia la conexión que los transporta, que ahora está cifrada y autenticada. El esquema de URI `https` se define en RFC 9110, la especificación de semántica HTTP de junio de 2022, que sustituyó a RFC 2818, el breve documento de mayo de 2000 que describió por primera vez HTTP sobre TLS. Su puerto predeterminado es el 443, mientras que HTTP en claro usa el 80.

RFC 9110 impone dos reglas al esquema. El cliente debe (MUST) comprobar que la identidad presentada por el servidor coincide de forma aceptable con el host de la dirección; en la práctica, el certificado debe cubrir ese nombre. Una conexión al nombre equivocado ha fallado, por sólido que sea su cifrado. Además, `http://example.com/` y `https://example.com/` son orígenes distintos, con espacios de nombres separados. Para el protocolo, solo comparten el nombre de host.

HTTPS no está ligado a una versión de HTTP. HTTP/1.1 y HTTP/2 funcionan sobre TLS en TCP. HTTP/3 funciona sobre QUIC, que incorpora el intercambio inicial de TLS 1.3 en el transporte. RFC 9114 descarta usarlo para acceder directamente a un servidor `http://` en claro, por lo que la versión más reciente llega con HTTPS o no llega.

Una dirección permite ver qué cubre el cifrado. En `https://shop.example/orders/1234?ref=mail`, se cifran la ruta, la cadena de consulta, todas las cabeceras, incluidas las cookies y `Authorization`, y los cuerpos de ambos mensajes. La red entre visitante y servidor sigue viendo la dirección IP y el puerto del servidor, los tiempos y tamaños del tráfico y, en la mayoría de las conexiones,

también el nombre de host `shop.example` : el navegador lo consultó en DNS y el cliente lo indica en el primer mensaje TLS para que una dirección pueda servir muchos sitios. HTTPS oculta qué se pidió, no a quién se pidió.

Utilice el concepto en una decisión

Aplique «¿Qué es HTTPS?» 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 pregunta vigente para un sitio público es si algo debe seguir respondiendo en `http://` ; los navegadores ya toman por sí mismos gran parte de esa decisión.

La primera palanca es el contexto seguro, término del W3C para una página entregada por HTTPS o desde la propia máquina. Los navegadores reservan un conjunto creciente de funciones para estos contextos. Los service workers son un ejemplo: `navigator.serviceWorker` se declara para contextos seguros, de modo que, en una página cargada por HTTP en claro, la propiedad sencillamente no existe, ni tampoco las funciones construidas sobre ella, empezando por la caché sin conexión. La especificación Secure Contexts también recomienda que quienes elaboren nuevas especificaciones protejan del mismo modo las funciones sensibles; la restricción es así el punto de partida para nuevos desarrollos, no una excepción decidida caso por caso.

La segunda palanca es el contenido mixto. Cuando la propia página es HTTPS, el navegador clasifica cada recurso que solicita según las reglas Mixed Content del W3C. Las imágenes, el audio y el vídeo solicitados por `http://` pasan automáticamente a `https://` y fallan si esa dirección no responde. Los scripts, las hojas de estilo y las peticiones de datos se bloquean directamente con un error de red. Basta, por tanto, una dirección en claro olvidada en una plantilla para dejar inoperativa una funcionalidad completa, no solo una imagen.

Eso deja muy poco para el puerto 80. Sigue siendo útil un servicio a la escucha que responda a cada petición con una redirección permanente a la misma ruta en `https://` , porque se escriben nombres sin esquema y los enlaces antiguos aún

dicen `http://` . El desafío HTTP de la autoridad certificadora también llega allí al emitir o renovar un certificado. HSTS permite después que un navegador que vuelve al sitio omita incluso la redirección. Cualquier otro uso del puerto 80 sirve una página sin las garantías anteriores.

La interfaz de bucle local es la única excepción permanente para sitios normales. La especificación considera `localhost` , `127.0.0.1` y `::1` potencialmente fiables, por lo que una página servida desde la máquina del desarrollador dispone de funciones de contexto seguro sin certificado. Esta comodidad para el desarrollo origina uno de los fallos descritos más adelante.

Cómo se aplica en la práctica

El sitio de la empresa sigue esa regla. En el host de producción, Caddy responde a todas las peticiones web y renueva por sí mismo el certificado HTTPS, de Let's Encrypt: mantenerlo vigente es tarea del servidor web, no de una persona. Las mismas páginas se publican como servicio onion de Tor y esa dirección empieza por `http://` de forma deliberada. Una dirección onion codifica la clave pública de identidad del servicio. Tor autentica el servicio y cifra la conexión antes de que se intercambie HTTP; por eso este espejo utiliza HTTP sobre Tor. La página de créditos también enlaza a Qualys SSL Labs, que evalúa la configuración HTTPS activa para cualquiera que lo solicite.

El trabajo para clientes muestra la regla de contexto seguro desde el otro lado. Para una plataforma cuyos usuarios suelen ser marinos y personal de campo con teléfonos, a menudo sin conexión, la empresa entregó soporte completo de aplicación web progresiva, con un service worker que almacena fuentes, imágenes, audio, vídeo y recursos de CDN con una estrategia de «primero la caché» (cache-first). Como los service workers solo funcionan realmente sobre HTTPS, el comportamiento sin conexión y de instalación se probó por HTTPS en dispositivos reales, en lugar de darlo por hecho. La misma necesidad aparece en las herramientas de un monorepositorio políglota con 578 objetivos de tareas: la capa go-task escrita por la empresa ofrece una variante HTTPS para probar PWA y móviles junto a los modos de desarrollo nativo y Docker.

Dónde surgen los problemas

Los scripts dejan de ejecutarse al pasar a HTTPS y desaparecen algunas imágenes. Las páginas aún contienen direcciones `http://` , fijadas en una plantilla, almacenadas en contenido del CMS o pegadas desde un fragmento de

terceros. El navegador bloquea los scripts y las hojas de estilo y cambia las imágenes a HTTPS, que fallan donde el otro host no lo ofrece. La consola señala cada dirección bloqueada. Busque el esquema antiguo en las plantillas y el contenido almacenado y reescriba cada coincidencia como `https://` o como ruta relativa al sitio:

```
grep -rnE '(src|href)="http://' templates/ content/
```

Una cabecera `Content-Security-Policy: upgrade-insecure-requests` ayuda a mantener las páginas en funcionamiento durante la limpieza, pero no modifica las direcciones almacenadas.

El sitio se redirige a sí mismo hasta que el navegador desiste. Una CDN o un equilibrador de carga termina la conexión TLS y reenvía la petición a la aplicación por HTTP en claro. La aplicación ve `http` y redirige a `https` ; el proxy reenvía el nuevo intento como `http` otra vez y el bucle acaba en un error «too many redirects». Esa misma falta de información hace que la aplicación construya enlaces absolutos que empiezan por `http://` . El proxy debe transmitir el esquema original mediante la cabecera estándar `Forwarded: proto=https` de RFC 7239 o la anterior `X-Forwarded-Proto` . La aplicación solo debe confiar en ella si procede de la dirección del propio proxy, ya que cualquier cliente puede enviarla. Asigne después la redirección a una sola capa, normalmente la que posee el certificado.

Una cookie de sesión creada por HTTPS viaja en claro. RFC 9110 advierte de que las cookies no respetan la separación entre los dos orígenes. Según las reglas de RFC 6265, una cookie sin el atributo `Secure` también se envía en peticiones `http://` al mismo host. Basta con que un visitante escriba el nombre sin esquema, o con que un atacante en la red responda a cualquier petición en claro no relacionada con una redirección a la dirección `http://` del sitio. Ponga `Secure` en todas las cookies que emita el sitio y añada HSTS para que el navegador ni siquiera haga la petición en claro.

El service worker funciona en el portátil y falta en el teléfono. El desarrollador abre `http://localhost:3000` , un contexto seguro, y todo funciona. El teléfono en la misma red abre `http://192.168.1.20:3000` , que no lo es: `navigator.serviceWorker` tiene el valor undefined, una biblioteca que lo comprueba omite el registro y el modo sin conexión falta sin avisar. Sirva también por HTTPS la versión de desarrollo. Caddy, por ejemplo, emite certificados para nombres y direcciones IP

locales desde su propia autoridad certificadora local e intenta añadir ese certificado raíz al almacén de confianza de la máquina. Su documentación advierte de que el paso no está garantizado y que corresponde al administrador instalar el certificado raíz en cada almacén; en un teléfono de pruebas hay que instalarlo a mano.

Lecturas adicionales

- RFC 9110, sección 4.2.2: esquema de URI https — Esquema, puerto predeterminado y obligación del cliente de comprobar la identidad del servidor.
- Contextos seguros — Qué páginas se consideran seguras, por qué se admite el bucle local y la recomendación de restringir las funciones nuevas.
- Contenido mixto — Qué peticiones de una página HTTPS se pasan a HTTPS y cuáles se bloquean.
- RFC 7239: extensión HTTP Forwarded — Cabecera estándar con la que un proxy transmite el esquema utilizado por el visitante.
- HTTPS automático — Explicación de Caddy sobre sus certificados, su redirección desde HTTP y su autoridad certificadora local.



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 HTTPS?» 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/https/