Engineer

SSH

Intérprete de órdenes remoto con conexión cifrada

Qué es

SSH, abreviatura de Secure Shell, es un protocolo de red para iniciar sesión en otro ordenador y ejecutar órdenes en él a través de una red que no es de confianza, con el tráfico cifrado y ambos extremos demostrando primero su identidad. El nombre abarca tres cosas: el protocolo, publicado por el IETF en enero de 2006 como un conjunto de RFC encabezado por el RFC 4251; OpenSSH, la implementación mantenida por el proyecto OpenBSD y utilizada en los ejemplos de este artículo; y la orden `ssh`, que se comunica con un demonio servidor llamado `sshd`.

El RFC 4251 describe tres capas sobre una conexión, normalmente en el puerto TCP 22, el número que le asignó IANA. La capa de transporte negocia las claves, demuestra la identidad del servidor mediante su clave de host y después cifra y comprueba la integridad de todo lo que circula. La capa de autenticación de usuarios funciona dentro de ese túnel y determina quién es el cliente: mediante clave pública, que todas las implementaciones deben admitir; mediante contraseña, que es opcional; o mediante confianza basada en el host. La capa de conexión divide el túnel en canales, de modo que un único inicio de sesión puede transportar a la vez un intérprete de órdenes interactivo, una orden individual, una transferencia de archivos SFTP y varios puertos reenviados.

SSH sustituyó a telnet y rlogin, que enviaban sesiones y contraseñas por la red como texto sin cifrar. La versión del protocolo utilizada hoy es la 2.

Utilice el concepto en una decisión

Aplique «¿Qué es SSH?» 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

En un servidor, SSH es la puerta de entrada. Una máquina en un centro de datos o en la nube rara vez tiene un teclado al alcance de alguien, por lo que la administración, los despliegues, las copias de archivos y la configuración automatizada llegan a través de SSH. Quien inicia sesión con privilegios suficientes tiene el control de la máquina.

La autenticación con clave pública permite que SSH sea cómodo y sólido a la vez. El usuario conserva una clave privada en su propio ordenador y coloca la clave pública correspondiente en el servidor, por defecto en `~/.ssh/authorized_keys`. Al iniciar sesión, el cliente firma datos vinculados a esa sesión, por lo que la propia clave nunca cruza la red. Una contraseña robada se puede reutilizar desde cualquier lugar, y cualquiera que alcance el puerto de un servidor que acepte contraseñas puede intentar adivinarlas; las herramientas automatizadas hacen precisamente eso. OpenSSH acepta contraseñas con su configuración inicial, así que cada servidor tiene que decidir desactivarlas.

La confianza también funciona en sentido contrario, porque el cliente necesita saber que ha llegado al servidor correcto. Cada servidor tiene una clave de host, y el RFC 4251 ofrece dos formas de confiar en ella: una lista local en el cliente que relaciona cada nombre de host con su clave, o una autoridad de certificación que firma las claves de host para que el cliente solo conserve la clave de esa autoridad. La mayoría utiliza la primera opción en su forma menos estricta, que el RFC admite como concesión a la facilidad de uso: aceptar la clave sin comprobarla en el primer contacto, guardarla en `~/.ssh/known_hosts` y compararla en cada conexión posterior. Esto suele llamarse confianza en el primer uso. El primer inicio de sesión solo es tan seguro como la red por la que pasa; por eso OpenSSH pregunta antes de guardar una clave nueva y rechaza la conexión cuando un host conocido presenta una distinta.

SSH también sirve de transporte para otras herramientas. Git lo utiliza para enviar y obtener cambios, `scp` y `sftp` para copiar archivos y Ansible para acceder a sus hosts. El reenvío de puertos permite que un portátil llegue a una base de datos que solo escucha en la interfaz de bucle local de un servidor sin exponerla a internet.

Cómo se aplica en la práctica

El servidor propio de la empresa acepta inicios de sesión interactivos únicamente mediante SSH, y su refuerzo de seguridad se comprueba en tres etapas cada vez que se ejecuta el rol de automatización encargado de la configuración. Las comprobaciones datan de septiembre de 2026, cuando un informe de acceso comparó los ajustes efectivos resueltos por el demonio con los escritos por el rol y encontró dos diferencias: los valores predeterminados de la distribución eran líneas activas que se leían antes del archivo complementario del rol, y sshd conservaba el primer valor. Renombrar ese archivo para que se ordenase antes que el del proveedor cerró la brecha. Ahora una configuración SSH debe superar tres comprobaciones: se valida la sintaxis de la candidata antes de instalarla, se comprueba el conjunto de archivos una vez instalado y se comparan los ajustes efectivos del demonio con 24 directivas esperadas, de modo que un valor desplazado por una línea anterior detiene la ejecución. Se omiten deliberadamente dos directivas: el demonio actual ya no admite ninguna de ellas.

Un mes antes, una revisión de agosto de 2026 encontró tres protecciones contra fuerza bruta que nunca habían funcionado desde que se creó el servidor. El mecanismo de bloqueo (jail) vigilaba el puerto 22 mientras el demonio escuchaba en el 1986. El límite de frecuencia del cortafuegos iba después de una regla más amplia que permitía el mismo tráfico y, como gana la primera regla coincidente, el límite nunca veía un paquete, aunque la salida de estado seguía mostrándolo. La acción de bloqueo dependía de una herramienta de filtrado de paquetes que el gestor de paquetes solo recomienda, por lo que el mecanismo detectaba a los atacantes y después fallaba cada vez que intentaba bloquearlos. La revisión dejó dos reglas permanentes: cada control de seguridad cuenta con una comprobación que demuestra que actúa, y una regla del cortafuegos se confirma tanto por su posición como por su presencia.

La forja Git propia de la empresa utiliza SSH como transporte, no como vía para iniciar sesión. Se eligió porque se maneja por SSH y no tiene un panel de administración web que proteger. Nadie accede de forma anónima ni sin clave, y todos los repositorios son privados. Su servicio SSH solo acepta conexiones en la interfaz de bucle local, en el puerto 23231, y desde fuera se llega a él a través de un servidor de salto, por lo que el cortafuegos no abre ningún puerto adicional para ese servicio.

Dónde surgen los problemas

Una directiva de refuerzo está en el archivo y el servidor la ignora. Para la mayoría de las palabras clave, `sshd` conserva el primer valor que lee, y `Include` expande los archivos en orden lexicográfico. Por tanto, cuando el archivo principal incluye primero una carpeta de archivos complementarios, un archivo del proveedor que se ordena antes prevalece sobre el propio, aunque ambos parezcan correctos. Buscar en la configuración solo demuestra que se escribió una línea. Pregunte al demonio: `sshd -t` comprueba la sintaxis y `sshd -T` muestra los ajustes que realmente utilizará.

```
/etc/ssh/sshd_config.d/10-vendor.conf:     PasswordAuthentication yes
/etc/ssh/sshd_config.d/50-hardening.conf:  PasswordAuthentication no

$ sudo sshd -T | grep -i '^passwordauthentication'
passwordauthentication yes
```

Se borra y se acepta una clave de host que ha cambiado. Cuando un servidor conocido presenta una clave de host diferente, `ssh` muestra una advertencia y rechaza la conexión. La reacción automática es ejecutar `ssh-keygen -R` con el nombre del host, volver a conectar y responder afirmativamente: justo lo que necesita un intermediario malicioso. Confirme primero la nueva huella por otro canal, por ejemplo ejecutando `ssh-keygen -lf` sobre la clave pública de host del servidor desde su consola. En scripts, `StrictHostKeyChecking no` es peor de lo que parece: además de guardar las claves nuevas sin preguntar, permite, con ciertas limitaciones, continuar las conexiones a hosts cuyas claves hayan cambiado. El valor `accept-new` guarda los hosts desconocidos y sigue rechazando las claves modificadas.

El acceso con contraseña sigue abierto a todo internet. `PasswordAuthentication` tiene por defecto el valor yes, al igual que la autenticación interactiva mediante teclado, que puede pedir la misma contraseña por otra vía. Cuando todos los usuarios dispongan de una clave, configure ambas opciones como no. `PermitRootLogin` ya tiene por defecto el valor `prohibit-password`, que rechaza las contraseñas de root, pero sigue permitiendo una clave de root. Trasladar el demonio fuera del puerto 22 cambia dónde escucha, no quién puede intentar adivinar las credenciales, y todos los controles que indiquen el puerto deben cambiar con él, como muestra el mecanismo de bloqueo anterior.

El reenvío del agente presta sus claves al servidor. Con `ForwardAgent` activado, o con `ssh -A`, cualquiera que pueda eludir los permisos de archivos del host remoto, incluido root, puede utilizar su agente para autenticarse como usted en otros equipos mientras siga conectado. No puede copiar la clave, pero sí firmar con ella. Para llegar a una máquina detrás de un bastión, utilice un servidor de salto: `ssh -J bastion.example.com app.internal` reenvía una conexión TCP a través del bastión y negocia la sesión de extremo a extremo con el destino, de modo que el bastión retransmite bytes que no puede leer y la clave permanece en su portátil.

Lecturas adicionales

- RFC 4251: The Secure Shell (SSH) Protocol Architecture — las tres capas, los dos modelos de confianza en claves de host y las consideraciones de seguridad.
- RFC 4253: The Secure Shell (SSH) Transport Layer Protocol — intercambio de claves, verificación de la clave de host y cadena de versión que cada extremo envía primero.
- RFC 4252: The Secure Shell (SSH) Authentication Protocol — cómo funcionan en el protocolo los accesos con clave pública, contraseña y confianza basada en el host.
- sshd_config(5) — todas las palabras clave del servidor con sus valores predeterminados y la regla de utilizar el primer valor obtenido.
- ssh_config(5) — el lado del cliente, incluidos StrictHostKeyChecking, ProxyJump y ForwardAgent.
- sshd(8) — los modos de comprobación `-t` y `-T` y los formatos de authorized_keys y known_hosts.

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 SSH?» 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/ssh/