Servicios en Windows

Servidor SSH

Pelayo Iglesias Manzano – UO266600

Pelayo Calleja Villafañe – UO292393

Índice

Introducción

Antes de hablar del Servidor SSH, debemos aclarar que es un **protocolo de red** y un **protocolo SSH.**

Los **protocolos de red** establecen de manera detallada el proceso que deben seguir los sistemas informáticos para realizar conexiones y transferencias de información. Gracias a estos protocolos, los sistemas involucrados en una comunicación son capaces de entenderse y proporcionar una interfaz homogénea para el acceso a un tipo de servicio.

Sabiendo lo anterior, **SSH** son las siglas de **Secure Shell** y es un protocolo de red destinado principalmente a la conexión con máquinas a las que accedemos por línea de comandos. En otras palabras, con SSH podemos conectarnos con servidores usando la **red Internet** como vía para las comunicaciones. Su característica más importante es que **siempre se realiza de manera segura**, ya que debido a SSH, la información que viaja por la Red no es legible por terceras personas y, para ello, todo el tránsito de la información se realiza **encriptando los datos**. Esto es importante para garantizar que el tráfico de datos se realice siempre de manera confidencial y que nadie sea capaz de escuchar el canal de comunicaciones para robar información o claves de acceso a los servidores. SSH se trata de una **alternativa** a tecnologías como FTP o Telnet, pero se **diferencia por cifrar los datos.**

El puerto predeterminado para las conexiones SSH es el 22.

Técnicas de cifrado

La ventaja significativa ofrecida por el protocolo SSH sobre sus predecesores es el uso del cifrado para asegurar la transferencia segura de información entre el host y el cliente. **Host** se refiere al servidor remoto al que estás intentando acceder, mientras que el **cliente** es el equipo que estás utilizando para acceder al host. Hay **tres tecnologías de cifrado diferentes** utilizadas por SSH:

- **Cifrado simétrico**
  El cifrado simétrico es una forma de cifrado en la que se utiliza una **clave secreta** tanto para el cifrado como para el descifrado de un mensaje, **tanto por el cliente como por el host.** Efectivamente, cualquiera que tenga la clave puede descifrar el mensaje que se transfiere. El cifrado simétrico a menudo se llama **clave compartida (shared key)** o **cifrado secreto compartido**.
  Tanto el cliente como el servidor derivan la clave secreta utilizando un **método acordado**, y la clave resultante **nunca** se revela a terceros. El proceso de creación de una clave simétrica se lleva a cabo mediante un **algoritmo de intercambio de claves**.
  Lo que hace que este algoritmo sea particularmente seguro es el hecho de que la clave **nunca** se transmite entre el cliente y el host. En lugar de eso, los dos equipos comparten **datos públicos** y luego los manipulan para calcular de forma independiente la clave secreta. Incluso si otra máquina captura los datos

públicamente compartidos, no será capaz de calcular la clave porque el **algoritmo de intercambio de clave no se conoce**.

- **Cifrado asimétrico**

  La diferencia entre el cifrado simétrico y asimétrico es que este último utiliza **dos claves separadas para el cifrado y el descifrado.** Estas dos claves se conocen como la **clave pública (public key)** y la **clave privada (private key).** Juntas, estas claves forman el **par de claves pública-privada (public-private key pair).** La clave pública se distribuye abiertamente y se comparte con todas las partes. Si bien está estrechamente vinculada con la clave privada en términos de funcionalidad, la clave privada no se puede calcular matemáticamente desde la clave pública. La relación entre las dos claves es **altamente compleja**: un mensaje cifrado por la clave pública de una máquina sólo puede ser descifrado por la misma clave privada de la máquina. Esta **relación unidireccional** significa que la clave pública no puede descifrar sus propios mensajes ni descifrar nada cifrado por la clave privada.

  Para que la conexión sea totalmente segura**, ningún tercero debe conocer la clave privada.** Esto se debe a que la clave privada es el único componente capaz de descifrar mensajes que fueron cifrados usando su propia clave pública.

  Pese a lo que uno se puede imaginar, el cifrado asimétrico no se utiliza para cifrar toda la sesión SSH. En realidad, **sólo se utiliza durante el algoritmo de intercambio de claves de cifrado simétrico**. Antes de iniciar una conexión segura, ambas partes generan pares de claves públicas-privadas temporales y comparten sus respectivas claves privadas **para producir la clave secreta compartida.**

  Una vez que se ha establecido una comunicación simétrica segura, el servidor utiliza la clave pública de los clientes para generar y desafiar y transmitirla al cliente para su autenticación. Si el cliente puede descifrar correctamente el mensaje, significa que contiene la clave privada necesaria para la conexión. Y entonces **comienza la sesión SSH.**

- **Hashing**

  El hashing unidireccional es otra forma de criptografía utilizada en **Secure Shell Connections.** La diferencia de las funciones de hash unidireccionales respecto a las dos formas anteriores de encriptación es que **nunca están destinadas a ser descifradas.** Generan un valor único de una longitud fija para cada entrada que no muestra una tendencia clara que pueda explotarse. Esto los hace prácticamente **imposibles de revertir.**

  Es fácil generar un hash criptográfico de una entrada dada, pero imposible de generar la entrada del hash. Esto significa que, si un cliente tiene la entrada correcta, pueden generar el hash criptográfico y comparar su valor para verificar si poseen la entrada correcta.

SSH utiliza hashes para **verificar la autenticidad** de los mensajes. Esto se hace usando **HMACs**, o códigos de autenticación de mensajes basados en hash. Esto asegura que el comando recibido no se altere de ninguna manera.
Mientras se selecciona el algoritmo de cifrado simétrico, también se selecciona un algoritmo de autenticación de mensajes adecuado. Esto funciona de manera similar a cómo se selecciona el cifrado, como se explica en la sección de cifrado simétrico.
Todo mensaje transmitido debe contener un **MAC**, que se calcula utilizando la clave simétrica, el número de secuencia de paquetes y el contenido del mensaje. Se envía fuera de los datos cifrados simétricamente como la sección final del paquete de comunicaciones.

SSH en Windows

El protocolo **SSH (Secure Shell)** se ha convertido en una herramienta fundamental para la administración de sistemas debido a su capacidad de establecer conexiones seguras y su flexibilidad en diferentes entornos. **OpenSSH** se integró en Windows a partir de **Windows Server 2019** y **Windows 10**. Como todo, tiene sus limitaciones, así que vamos a hablar de las ventajas y desventajas que estas provocan.

Las **ventajas** que SSH provoca en Windows son:

- **Cifrado de datos**: Todas las comunicaciones se cifran, lo que protege contra escuchas ilegales y ataques de intermediario **(Man-in-the-Middle).**
- **Autenticación robusta**: Soporta claves públicas/privadas, que son más seguras que las contraseñas tradicionales.
- **Compatibilidad con firewalls y restricciones de red**: SSH se puede configurar para funcionar en redes seguras con políticas estrictas.
- La **integración en Windows** permite que los administradores gestionen **entornos mixtos** sin necesidad de software adicional.
- **Facilita la conexión y administración entre servidores Windows y otros sistemas**, reduciendo barreras de compatibilidad.
- SSH está **incluido como una característica opcional** en Windows, eliminando la necesidad de instalar clientes como **PuTTY**, de los cuales hablaremos más adelante.
- **La configuración inicial es sencilla,** y la mayoría de las opciones se pueden ajustar desde un único archivo de configuración **(sshd_config).**
- A través de **SCP** o **SFTP**, se pueden transferir archivos de forma segura entre sistemas Windows y otros dispositivos.
- SSH se puede usar junto con herramientas como **PowerShell** para ejecutar scripts en múltiples servidores de manera remota, facilitando la automatización de tareas administrativas.
- Con **túneles SSH**, puedes redirigir puertos para acceder a **servicios internos de una red** (como bases de datos o aplicaciones) de manera segura, incluso desde redes no confiables.

- SSH es compatible con entornos locales y servicios en la nube, como **Microsoft Azure**.

Pese a que las ventajas son muchas, también hay que destacar sus **desventajas:**

- **Aunque OpenSSH en Windows cubre las funciones principales**, algunas características avanzadas, como ssh-agent para gestionar claves o túneles persistentes, pueden ser **menos flexibles** o estar limitadas en Windows.
- Toda la configuración avanzada se realiza a través de un archivo de texto plano, lo que puede ser **complicado para usuarios nuevos.**
- **No hay una interfaz gráfica nativa** para configurar el servidor SSH.
- La **configuración predeterminada no es la más segura**. Los administradores deben tomar **medidas adicionales** para mejorar la seguridad, como **deshabilitar contraseñas y habilitar solo claves públicas.**
- SSH utiliza permisos de archivos similares a los de Linux (propietarios, grupos, y otros). En Windows, esto puede entrar en **conflicto con el sistema de permisos NTFS,** lo que a veces causa errores o configuraciones incorrectas.
- Algunas herramientas de Windows **no se integran** completamente con SSH.
- El servidor SSH en Windows **no está diseñado** para manejar miles de conexiones simultáneas, lo que puede ser un **problema** en entornos empresariales de **gran escala**.

Configuración de Servidor SSH

En este apartado vamos a profundizar en como **configurar un ordenador concreto como servidor SSH.** Una vez lo hayamos conseguido, podremos conectarnos desde el resto de los dispositivos a este ordenador que hace de servidor. También hay que decir que se trata de un método para profesionales y **para ser utilizado en centros de datos.** Hoy en día vamos a poder recurrir al SSH **prácticamente desde cualquier ordenador** independientemente de su sistema operativo.

Configurar el **SSH** en Windows 10 es relativamente sencillo. Lo primero que tenemos que hacer es activar un ordenador como servidor. Para ello, se enciende el PC que vayas a usar como servidor, **pulsamos Windows+R y escribimos services.msc** en la ventana que nos aparece para ejecutar programas.

A continuación, se abrirá la ventana de servicios de Windows. En esta ventana, tenemos que buscar y **hacer click derecho sobre el servicio OpenSSH SSH Server** que tenemos. En el menú emergente hay dos cosas que podemos hacer. Si solo quiere iniciar el servidor SSH pulsamos en Iniciar, pero si queremos que se inicie automáticamente todas las veces que encendamos el ordenador para no tener que hacerlo manualmente, entra a **Propiedades** y cambiamos el tipo de inicio de **Manual a Automático.**

Una vez hayamos iniciado **OpenSSH Server,** tenemos que hacer **exactamente lo mismo con OpenSSH Authentication Agent.** Este posiblemente nos aparezca como **Deshabilitado**, por lo que tenemos que entrar en sus propiedades sí o sí, y elegir el inicio

manual para iniciarlo después haciendo click derecho sobre él o el automático para que se inicie luego también cada vez que encendamos Windows.

Después de esto, abrimos el menú de inicio y escribimos **PowerShell.** Cuando salgan los resultados, tenemos que entrar a Windows PowerShell **como administrador.** Para lo que vamos a hacer no vale el símbolo de sistema, tiene que ser la consola de comandos PowerShell con permisos de **administrador.** Una vez estemos dentro de PowerShell como administrador tenemos que escribir el comando **New-NetFirewallRule -Name sshd -DisplayName 'OpenSSH Server (sshd)' -Service sshd -Enabled True -Direction Inbound -Protocol TCP -Action Allow -Profile Domain.** Cuando hagamos esto, habremos habilitado todo lo necesario y habremos abierto el puerto 22 para poder acceder a este ordenador con un cliente SSH desde fuera de nuestra red doméstica.

En el ordenador que hace de Servidor, ya habremos terminado, pero **vamos a necesitar su IP**¸ o sea que anotemos la IP de nuestro ordenador y la tenemos que anotar. Ahora, tenemos que ir al otro ordenador, desde el que quieras acceder, y descargamos un programa que se llama **Putty,** el cuál es un cliente sencillo para conectarnos a servidores SSH. Instalar **Putty,** es un proceso sencillo y limpio, sólo tenemos que elegir una carpeta de destino. Una vez instalado el **Putty,** ya sólo tenemos que escribir la **IP** donde nos pone **Host Name y pulsar el botón Open.**

Casos de uso

Este protocolo permite diferentes acciones, entre ellas, conectarte a un servidor de manera remota y, lo mejor de todo, de forma **segura.** Aunque esto es solo una de las distintas opciones que ofrece la opción de utilizar el **protocolo SSH.**

Volviendo a la seguridad, utilizar SSH no solo es bueno, sino que también es un hecho relevante. Especialmente **si se quiere utilizar una conexión cifrada.** De lo contrario, cualquier usuario puede llegar a tener la opción de interceptar esa transmisión de datos, por lo que pondrías en peligro tu información personal, como usuarios, contraseñas, datos bancarios, etc. Para que se pueda tener una idea más clara de para qué sirve el **protocolo SSH,** vamos a hablar de las acciones más comunes que se pueden realizar a través de este protocolo en particular:

Conectar remotamente a un servidor:

El uso más importante del protocolo SSH es el de conectarnos de forma remota a un **servidor.** Esto puede ser de forma gráfica, como sería un programa en Windows, pero también se puede hacer a través de la terminal, como por ejemplo en Linux.

Siempre vamos a tener que utilizar el nombre de usuario y contraseña correspondiente, ya que este protocolo requiere autenticación. Esto es lo que nos permitirá acceder de forma remota a un servidor y poder controlarlo o llevar a cabo diferentes acciones sin necesidad de estar físicamente delante

Actualizar un dispositivo o realizar cambios:

De forma remota y a través de **SSH** vamos a poder **actualizar un dispositivo.** Por ejemplo, podemos acceder a un NAS para actualizar a una nueva versión del firmware y enviar los archivos que sean necesarios para ello.

También podemos realizar cambios en la configuración de forma remota. Por ejemplo, instalar una aplicación o incluso reiniciar el dispositivo si hubiera algún error. Todo ello, una vez más, sin necesidad de estar físicamente delante de ese dispositivo.

Modificar o copiar archivos:

También podremos **enviar archivos** de un equipo a otro a través del protocolo SSH. Esto significa que podremos estar trabajando desde un ordenador, por ejemplo, y posteriormente subir esos archivos a un servidor mediante este protocolo.

De la misma manera, podemos acceder a un servidor y modificar los archivos que ya hay. Esto evita que tengamos que descargar los archivos, modificarlos y posteriormente volverlos a enviar. Lo que hacemos es modificarlos directamente en el servidor.

Transferencia de archivos segura

SSH no solo se limita al acceso remoto, sino que también ofrece un protocolo específico para la transferencia segura de archivos, conocido como **SFTP.** Este protocolo proporciona una capa adicional de seguridad en comparación con los métodos tradicionales como **FTP,** ya que toda la comunicación, incluida la autenticación y la transferencia de datos, **está cifrada.** Esto asegura que la información sensible, como nombres de usuario y contraseñas, estén **protegidas** durante la transmisión.

La principal **ventaja de SFTP** es su capacidad para transferir archivos de manera segura a través de conexiones encriptadas. Al utilizar el mismo mecanismo de **autenticación y seguridad** que SSH, SFTP garantiza la confidencialidad en integridad de los datos, convirtiéndose en una opción preferida para la transferencia de archivos en entornos donde la seguridad es primordial. Además, SFTP suele ser compatible con **firewalls,** ya que utiliza el mismo puerto que SSH.

Gestión de dispositivos en red

Muchos administradores de red utilizan SSH para acceder y administrar dispositivos de red, como routers y switches, de manera segura.

La gestión segura de dispositivos de red es una parte fundamental en entornos de red y SSH tiene un papel crucial en este tema. Los administradores de red utilizan SSH para acceder y administrar dispositivos de manera remota y segura. La clave de esta función está en la capacidad de SSH para **proporcionar una conexión cifrada,** lo que garantiza que la información crítica transmitida entre el administrador y el dispositivo de la red esté protegida contra posibles amenazas.

El uso de SSH en la gestión de dispositivos de red también permite a los administradores realizar tareas de configuración, monitoreo y resolución de problemas de manera eficiente. La encriptación de extremo a extremo proporcionada por SSH

asegura que los comandos y la información confidencial no sean vulnerables a ataques de interceptación, lo que es muy importante para mantener la integridad y la seguridad de la infraestructura de la red

Clientes SSH

Los clientes SSH, al tratarse de herramientas que se utilizan para establecer conexiones con servidores SSH, deben cumplir algunas funciones. Para que estos sean considerados buenos, deben dispones de un conjunto de funciones y características, que nos ayudarán a mantener la seguridad adecuada para cada caso. A la vez que dispondremos del rendimiento óptimo en todos los casos. Estas funcionalidades y características son:

- **Seguridad:** Es imprescindible en prácticamente cualquier sistema cumplir con unos mínimos de seguridad. Se debe disponer de un cifrado fuerte que proteja las conexiones, así como las credenciales de acceso del usuario. Esto debe ir acompañado de un sistema de autenticación que impida accesos no autorizados.
- **Interfaz:** Disponer de una interfaz sencilla e intuitiva, es un punto importante para optimizar el uso que le damos al cliente. Actualmente, la mayoría cuentan con interfaces gráficas que permiten que los usuarios se puedan conectar de una forma rápida y sencilla.
- **Soporte:** La compatibilidad del cliente es vital para empresas y usuarios que dan uso de diferentes sistemas operativos. Lo mejor, es buscar opciones que sean compatibles con los sistemas más conocidos actualmente.
- **Funciones avanzadas:** Más allá de las funciones de conexión y autenticaciones básicas, un buen cliente SSH debe contar con características añadidas. La posibilidad de guardar conexiones, configurar otros parámetros, transferencia de archivos o la capacidad de ajustar velocidades de conexión, son buenos añadidos dentro de cualquier cliente SSH.
- **Actualizaciones:** El cliente debe recibir actualizaciones periódicas del desarrollador. Esto no solo mejorará el rendimiento del sistema, sino que nos dará una protección mayor, y en algunos casos, nuevas funciones para el cliente.
- **Protocolos:** Que sea compatible con muchos protocolos es una buena forma de ver si estamos ante un buen cliente SSH. Esto facilitará la transferencia de archivos, así como nos ayudará a mantener la seguridad.

A continuación, os hablaremos de algunos clientes SSH

PuTTY

Es sin duda el mejor cliente SSH que podemos encontrar para Windows. Liviano y muy sencillo, este cliente cumple con su función: permitirnos conectar a cualquier servidor Windows o Linux de forma remota para controlarlo de forma segura a través de Internet. Además, cuenta con una serie de opciones que nos permite configurar, por ejemplo, la apariencia o guardar una serie de sesiones para restaurarlas fácilmente.

CMDER

Mientras que las aplicaciones anteriores están pensadas especialmente para permitirnos conectarnos a servidores SSH remotos, **CMDER** busca llevarnos una experiencia similar a la terminal de Linux a Windows mezclando el potencial de dos herramientas: **Clink y ConEmu.** De esta manera, con esta alternativa a CMD vamos a poder ejecutar muchos más comandos en nuestro sistema operativo, como, por ejemplo, SSH, para conectarnos de forma remota a cualquier servidor

Solar Putty

Solar Putty es un cliente SSH que destaca por traer una serie de mejoras muy demandadas por los usuarios sobre el famoso programa Putty. Una de las más interesantes es tener una interfaz que utiliza múltiples pestañas para poder utilizar varias sesiones SSH desde una sola ventana, pero eso no es todo ya que este cliente SSH te permite una serie de características tan importantes como:

- Puedes guardar las contraseñas o las claves privadas en cualquier sesión.
- Puedes automatizar una serie de comandos, cuando se inicia sesión con el servidor SSH.
- Puedes buscar en Windows la sesión guardada gracias a la integración de Windows Search
- No requiere ningún tipo de instalación
- Solar Putty también es compatible con los protocolos telnet, SCP, SFTP TFP.
- Puede auto conectarse al servidor SSH en caso de desconexión.

Como podemos ver, añade una serie de características que hacen que Solar Putty sea una opción para tener en cuenta antes que Putty.

SecureCRT

SecureCRT es un cliente SSH disponible para Windows, Mac y Linux que es compatible con los protocolos SSH1, SSH2, Telnet, Rlogin. Las características principales de SecureCRT es que nos proporciona acceso remoto de forma segura, transferencia de archivos, nos permite modificar por completo la apariencia del programa, podemos trabajar mediante pestañas en varias sesiones a la vez.

Su interfaz gráfica permite asignar acciones predefinidas para poder enviar comandos, ejecutar scripts o iniciar programas externos, además, incorpora un servidor TFTP para poder transferir ficheros usando SFTP, Xmodem, Ymodem, Zmodem o Kermit. Por último, SecureCRT es compatible con scripts VBScript, JScript, PerlScript o Python.

mRemoteNG

Es un fork del famoso programa mRemote, que agrega una serie de mejoras para que lo hacen mucho mejor que su antecesor. Este cliente SSH admite múltiples protocolos como RDP, VNC, ICA, SSH, telnet, HTTP, HTTPS, RLogin y raw sockets.

Una de sus características estrella es que tiene la capacidad de poder trabajar con pestañas, para así poder tener a la vez varias sesiones SSH abiertas. Además, incorpora un gestor para poder guardar las sesiones SSH y cargarlas rápidamente.

MobaXterm

MobaXterm es un cliente SSH que tiene soporte con los protocolos SSH, telnet, Rlogin, RDP, VNC, XDMCP, FTP y SFTP. Las características que hacen muy diferente a MobaXterm respecto a otros clientes SSH es que incluye “X server”, y tiene soporte de plugins, addons y macros. Otra característica estrella de MobaXterm es que no requiere instalación y es portable, lo que es ideal para ser guardado en un pendrive y ejecutarlo desde el mismo.

Otro punto muy interesante es que el terminal de MobaXterm puede resaltar o usar diferentes colores en las palabras clave, y por supuesto tiene soporte para trabajar con varias sesiones SSH a la vez usando pestañas

BITVISE

Es una herramienta que solo funciona en Windows, es una de sus pocas desventajas, sin embargo, admite todas las versiones hasta la última versión de Windows 11. Es gratuito y admite un número ilimitado de conexiones de usuario. Además, tiene capacidades de cliente SFTP seguras y funciones SSH. Es compatible con todos los principales servidores SSH/SFTP y tiene un proceso de instalación y configuración sencillo. Con una interfaz fácil de usar, también cumple con los requisitos de HIPAA, FIPS y PCI.

La aplicación, nos brinda conexiones con secuencias de comandos y admite la autenticidad a través de un directorio activo. Permite la entrada remota segura a través de la GUI y la consola. Tiene un modo de transferencia de archivos bloqueado, así como autenticación de dos factores. La aplicación también admite cuentas virtuales y ayuda a limitar la capacidad disponible para grupos de usuarios.

Xshell

Es uno de los clientes SSH más potentes. Esta herramienta permite a los usuarios crear, iniciar y editar sesiones fácilmente con el Administrador de sesiones y las Propiedades de sesión heredables. Además, la herramienta tiene varios métodos de autenticación, algoritmos y protocolos diferentes para manejar cualquier situación. También tiene acceso a herramientas como WSL, Powershell, CMD y más directamente dentro de Xshell. Además, la aplicación admite alias personalizados para los comandos que se usan con frecuencia. La interfaz de Xshell es intuitiva y con pestañas, y se pueden crear botones de comando rápido. La mejor parte de Xshell es que puede redactar y editar

varias líneas de cadena antes de enviarlo. También tiene la función de tunneling instantánea para crear túneles mientras tiene una sesión en ejecución.

OpenSSH

OpenSSH para Windows es una adaptación de la popular implementación de código abierto del protocolo SSH para sistemas Unix/Linux, ahora **disponible para entornos Windows**. Aunque tradicionalmente asociado con sistemas Unix/Linux, OpenSSH se ha vuelto cada vez más relevante en el ecosistema de Windows gracias a herramientas como Git for Windows o Windows Subsystem for Linux, que permiten a los usuarios de Windows acceder a las funcionalidades de Unix/Linux. Esto significa que los usuarios de Windows pueden instalar y utilizar OpenSSH de la misma manera que lo harían en un entorno Unix/Linux, lo que les da acceso a un conjunto completo de herramientas de línea de comandos para conectarse de forma segura a servidores remotos a través de SSH.

Termius

Termius es un cliente SSH multiplataforma que ofrece una experiencia moderna y fácil de usar para los usuarios de Windows, macOS, Linux, iOS y Android. Una de las características destacadas de Termius es su capacidad para **sincronizar en la nube configuraciones y sesiones entre diferentes dispositivos**, lo que permite a los usuarios acceder a sus conexiones desde cualquier lugar y en cualquier momento. Además, Termius ofrece herramientas avanzadas de gestión de equipos, que permiten a los usuarios organizar y administrar sus conexiones en grupos, lo que facilita la gestión de varios servidores remotos.

Termius también soporta autenticación de clave pública, lo que proporciona una capa adicional de seguridad al iniciar sesión en servidores remotos. Los agentes SSH y los túneles SSH dinámicos son otras características avanzadas que ofrece Termius, permitiendo a los usuarios gestionar y utilizar de forma eficiente sus conexiones SSH.

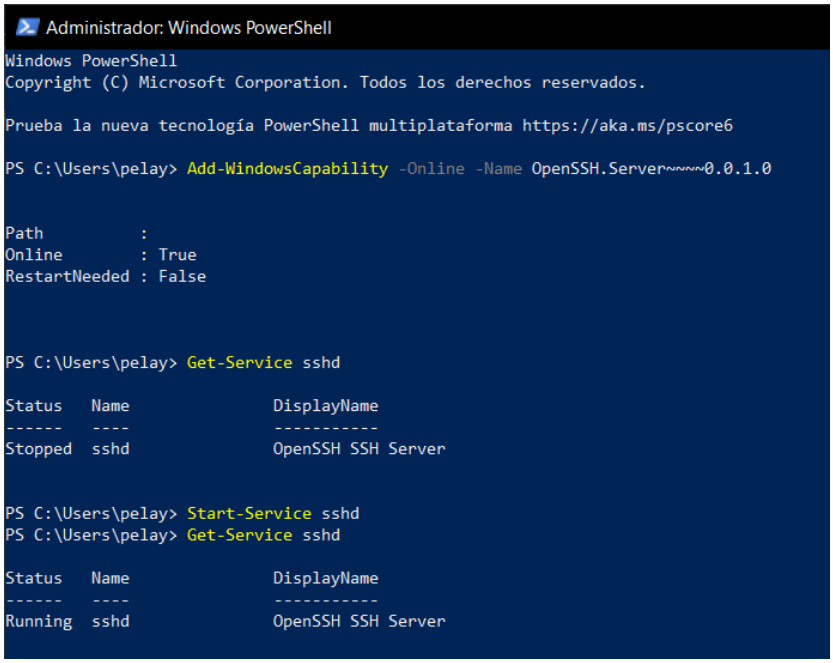
SuperPuTTY

Funciona para Windows y lo cierto es que funciona de la manera más sencilla. Permite ejecutar no solo comandos básicos, sino también cuenta con un administrador de ventanas.

Hay que tener en cuenta que esta aplicación en concreto permite la opción de transferir archivos a través de SCP entre lo que es el sistema local y el remoto, de esta manera se consigue que los usuarios puedan subir archivos de manera segura mediante SFTP. Y como no podía ser de otra forma, es **compatible con el protocolo SSH**, al igual que con Telnet, RAW y RLogin. Por lo tanto, se trata de una de las diferentes posibilidades que podemos tener sobre la mesa en Windows.

Prueba experimental

En la parte experimental de este trabajo hemos querido realizar la instalación completa de un **servidor OpenSSH** en Windows 10. Para llevar esta tarea a cabo se ha utilizado la aplicación Powershell, la cual viene de forma nativa en este sistema operativo.



En esta captura podemos ver que se han realizado 4 acciones: **instalación del servidor OpenSSH, verificación de la instalación, inicio del servicio y verificación del inicio.**

Finalmente, se han llevado a cabo 2 pasos más: **configuración del servicio para que se inicie automáticamente y permiso para realizar conexiones SSH en el firewall de Windows.**

Conclusión

Configurar un servidor SSH en Windows es una forma sencilla y efectiva de mejorar la seguridad y facilitar la administración remota.

SSH proporciona un canal cifrado que permite a los administradores y usuarios conectarse de forma segura a sistemas, transferir archivos y ejecutar comandos sin exponer la red a riesgos innecesarios. Aunque al principio puede parecer complicado, el proceso de instalación y configuración es bastante directo, y una vez configurado, ofrece muchas ventajas.

En resumen, el servidor SSH en Windows es una herramienta clave que ayuda a proteger las conexiones y facilita el trabajo remoto, haciendo que la administración de sistemas sea más segura y eficiente.

Bibliografía

https://www.arsys.es/blog/ssh#tree-2

https://www.hostinger.es/tutoriales/que-es-ssh#Comprendiendo_las_diferentes_tecnicas_de_cifrado

https://es.wikipedia.org/wiki/Secure_Shell

https://www.paessler.com/es/it-explained/ssh#:~:text=La%20principal%20ventaja%20de%20SSH,sentados%20f%C3%ADsicamente%20frente%20al%20dispositivo.

https://sshgrauvicofernandez.wordpress.com/2014/11/27/ventajas-y-desventajas-del-ssh-2/

https://www.godaddy.com/resources/es/seguridad/ssh-secure-shell-que-es-como-funciona

https://nordvpn.com/es/blog/ssh-windows/

https://www.xataka.com/basics/ssh-windows-10-que-como-configurarlo

https://www.redeszone.net/tutoriales/internet/protocolo-ssh-usos/#448821-para-que-se-puede-utilizar

https://www.redeszone.net/tutoriales/servidores/mejores-clientes-ssh-windows/