SSH

El protocolo Secure Shell (SSH) es el nombre de un protocolo (y del programa que implementa el homónimo) que nos permite acceder de forma remota y ,más importante , segura, a un servidor, copiar datos de forma segura, gestionar claves RSA.
Para asegurarnos de que podemos conectarnos de forma segura con el servidor este protocolo nos conecta mediante un canal seguro, en el cual, toda la información está cifrada. La forma de trabajar de SSH es similar a la de Telnet ,pero a diferencia de Telnet, SSH utiliza técnicas de cifrado para que la información viaje de manera no legible, de esta manera, evitamos que tercero puedan descubrir nuestro usuario o contraseña, ni lo que se escribe durante la sesión.

Un poco de historia

Inicialmente, solo existían los r-commands, basados en el programa rlogin, el cual tenía un funcionamiento muy similar al de Telnet.
La primera versión del protocolo fue creada EN 1995 por Tatu Yiönen, un finlandes que originalmente creo el protocolo de forma que este fuese libre, lamentablemente, la licencia del protocolo cambió y terminó apareciendo la compañía SSH Communications Security, que lo ofrecía gratuitamente para uso doméstico y académico, pero exigía el pago a otras empresas.
A inicios de 1999 se comenzó a escribir una versión de uso libre de SSH conocida como OpenSSH.

Paramiko

Paramiko es una implementación del protocolo SSHv2 en Python, la cual provee funcionalidad cliente - servidor, librería de SSH de alto nivel.

Clave pública y privada

La clave SSH consiste en la generación de un par de claves que proporcionan largas cadenas de caracteres, una pública y otra privada. La pública se instala en cualquier servidor y se desbloquea mediante la conexión con un cliente SSH, el cual, hace uso de la clave privada.
En caso de que ambas clases coincidan, el servidor SSH permite el acceso sin la necesidad de una contraseña.

La implementación en una cáscara de nuez

Primero, se importan 3 librerías, time, paramiko y getpass.
Time se importa con el objetivo de poder medir un intervalo de tiempo entre comando y comando, para que así, se evite un flujo excesivo de comandos que puedan afectar el rendimiento de la terminal, Paramiko se importa con el objetivo de crear un cliente SSH y getpass con el objetivo de ingresar una contraseña que no sea sensible a vulnerabilidades de seguridad humanas.

Se crea el objeto “Host” con la IP del servidor al que nos queremos conectar y el nombre de usuario con el cual nos queremos conectar. Lo próximo que se hace es crear un objeto cliente, que tenga las propiedades de un cliente SSH dadas por la librería paramiko y se setea que para el ingreso se usen los datos del cliente y no otros.

Usando la función “getpass” se solicitará la contraseña, la cual será guardada en la variable “password”, y se realizará la conexión del cliente usando la función “connect”, a la cual, se le ingresarán los parámetros “Host”, “User” y “Password”. Lo próximo que se hará es crear un objeto sftp_client, al cual, a través de la función “.open_sftp”, se le darán las propiedades para cumplir con el protocolo SFTP (Secure File Transfer Protocol) lo cual permitirá, a este objeto, realizar intercambios de archivos entre el cliente y el servidor. Para usarlo se puede utilizar la función “.put” a la cual se le ingresarán como parámetros el path del archivo local que quiero enviar y el path de a donde lo quiero enviar en el servidor, junto con el nombre que deseo darle a dicho archivo, o también, se puede usar la función “.get” a la cual se le tienen que ingresar como parámetros el path del archivo a descargar del servidor y el path de donde quiero alojar ese archivo en mi equipo junto con el nombre que le quiero dar.

Ahora con el uso de la función “exec_command()” a la cual se le ingresara como parámetro, un string con el comando a realizar en terminal, se darán como resultado el objeto “stdin”, “stout” y “sterr”, los cuales serán respectivamente el estándar input que quiero dar a la terminal, el estándar output que me devolverá el comando ingresado y el estándar error que me daría en caso de un error.

Para evitar excesos de inputs seteo un tiempo de espera tras ejecutar un comando, tras esto creo el objeto “result” el cual es una modificación legible y en formato string del resultado “stout”, tras esto, muestro en pantalla la variable “result” y cierro la conexión con el servidor con la función “.close”, esto funciona de forma equivalente a como funciona un script en bash a la cual yo puedo programa desde código, para automatizar las acciones que yo quiera desde el servidor con una variable SSH. Esto puede tener como utilidad, cargar una carpeta local al servidor o descargar una copia de seguridad de todo el servidor a mi equipo.

Otras funciones pueden ser realizar diagnósticos del servidor viendo la cantidad de archivos y viendo el estado en el que se encuentran,

Bibliografía

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

https://pyneng.readthedocs.io/en/latest/book/18_ssh_telnet/paramiko.html

https://pyneng.readthedocs.io/en/latest/book/18_ssh_telnet/paramiko.html