Administración de Sistemas y Redes
T.P. 2
El protocolo SSH

Introducción

En este reporte plasmaremos toda la información recolectada durante nuestro trabajo de investigación sobre el protocolo SSH. Hablaremos a detalle de la forma en que este protocolo funciona, cuáles son sus rasgos distintivos y qué es lo que provoca que los usuarios lo elijan por sobre otros protocolos. Además, explicaremos en qué áreas del campo de la informática es comúnmente utilizado, cuáles son sus puntos fuertes y sus puntos débiles, de dónde proviene y cuándo fue creado. Para finalizar, destinaremos una sección a la implementación de SSH en una pequeña aplicación de ejemplo, escrita en el lenguaje Python.

Características

SSH como todo protocolo de comunicación basa su funcionamiento en permitir la conexión entre 2 partes. Este se diferencia principalmente de los otros protocolos de comunicación, como Telnet o FTP, mediante la encriptación de la sesión de conexión, haciendo que sea imposible la obtención de contraseñas encriptadas por parte de terceros. Este protocolo fue diseñado para sustituir a los métodos más antiguos y menos seguros, y para permitir la conexión remota en otro sistema mediante la shell de comando.

Ya dijimos que la seguridad que brinda el protocolo SSH es su diferencial frente al resto de protocolos de comunicación y su característica principal, pero es importante destacar que hay ciertas funciones generales que conforman está característica. Estas funciones se dividen en:

- Privacidad de Datos mediante el uso de encriptación.
- Integridad de las Comunicaciones para garantizar su inalterabilidad.
- Autenticación en la Comunicación, es decir, una confirmación de identidad de tanto remitente como destinatario.
- Autorización para el acceso y las funciones de los distintos usuarios.
- Reenvío o Tunelización de sesiones para lograr su cifrado.

Aplicaciones

El protocolo SSH es utilizado principalmente como medio de conexión remota hacia un servidor, está conexión es llevada a cabo mediante la existencia de un usuario y una contraseña ya que es un protocolo que requiere autenticación. Otro uso posible es el de actualizar un dispositivo de forma remota. También se puede hacer cambios en el sistema como instalar aplicaciones o llevar a cabo un reinicio en el sistema, todo esto sin necesidad de estar presente físicamente frente al dispositivo. Por último, una de las funcionalidades principales es la de enviar archivos de un equipo a otro, es decir, es posible trabajar en un archivo desde un dispositivo y luego subir estos archivos a un servidor gracias al protocolo SSH. Asi mismo es posible acceder a un servidor y modificar los archivos que haya en el servidor desde este mismo, evitando tener que descargarlos, modificarlos y enviarlos nuevamente al servidor.

Seguridad

Como ya se mencionó anteriormente, SSH es un protocolo el cual se caracteriza por la encriptación de la sesión de conexión. Está encriptación es posible gracias a técnicas de cifrado las cuales hacen que la información que viaje por el medio de comunicación sea no legible, evitando asi que el usuario y la contraseña de dicha conexión pueda ser interceptada por terceras personas. Es importante destacar que es posible atacar este tipo de sistemas, pero es mucho más difícil que atacar a las otras opciones.

El protocolo SSH brinda distintos tipos de protección, estos son:

- Luego de realizar la conexión, el usuario puede verificar que esté efectivamente conectado al servidor al que se haya conectado previamente.
- La información transmitida por el usuario está cifrada mediante una encriptación robusta de 128 bits. Toda información enviada y recibida durante la sesión es encriptada de está forma, provocando que sea muy difícil descifrar y leer está información.
- El usuario puede hacer uso del reenvío de aplicaciones de ventanas gráfica X11 desde el mismo servidor, brindando un medio seguro para su uso sobre una red.

Otra característica de seguridad del protocolo SSH es su uso para asegurar protocolos inseguros, que es posible gracias a la encriptación de toda información enviada y recibida, y la aplicación de la técnica conocida como “Reenvío por Puerto”.

Por último, falta mencionar que el protocolo SSH presenta tres alternativas de distintas estructuras de criptografía las cuales son aplicadas al momento de su uso. Estas son:

- ***Criptografía Simétrica:*** es realizada mediante una clave secreta, la cual es compartida entre el servidor y el usuario. Su papel es el de encriptar o desencriptar el mensaje que es transferido en ese proceso, sin embargo, el contenido solo puede ser leído mediante la presentación de dicha clave.

- ***Criptografía Asimétrica:*** utiliza 2 claves, una para el cliente y otra para el servidor, para que suceda la criptografía de los datos transferidos. Las claves son llamadas públicas y privadas y forman la combinación necesaria para generar el SSH y su protocolo de seguridad. En este modelo la clave pública es abierta y compartida, ya que a partir de está no se puede conseguir la clave privada. Esto sucede debido a un proceso que funciona de la siguiente forma: mensajes criptografiados por claves públicas solo pueden ser desencriptados por la clave privada de la misma máquina.
- ***Hashing:*** es un método de criptografía unidireccional usado en el SSH. Consiste en crear un hash, por medio de un algoritmo, para garantizar que el mensaje sea protegido en una forma específica de criptografía y códigos de autenticación.

Vulnerabilidades

Si se hace uso del protocolo SSH de la forma correcta, sin cometer errores que puedan perjudicar la seguridad de los datos del usuario, se puede afirmar que SSH es de los protocolos más seguros que existen, y mucho más seguro que sus antecesores como Telnet.

Dicho esto, hay algunos errores comunes los cuales pueden hacer al protocolo SSH vulnerable. Estos son:

- ***Débil configuración:*** tanto el cliente como el servidor de SSH incluyen parámetros específicos de configuración los cuales son omitidos por los administradores en ciertas ocasiones. Algunos optan por los ajustes por defecto como el “port forwarding” dado su conveniencia, lo que aumenta el riesgo de seguridad.
- ***Falta de Conciencia:*** por más que el protocolo SSH se basa en la autenticación mediante claves para asi generar una conexión segura entre administradores, aplicaciones y máquinas, las organizaciones muchas veces fallan al proveer un refuerzo en las políticas necesarias para prevenir un acceso no autorizado.
- ***Desorganización en el manejo de Claves:*** dado que se pueden generar tantas claves SSH como sean necesarias, se puede llegar al caso en el que claves descartadas o no utilizadas sean guardadas en múltiples lugares. En caso de que una de estas claves no utilizada sea adquirida por alguna tercera persona, abre la posibilidad de sufrir un ataque de escucha, como puede ser un ataque MITM (el atacante se posiciona en el medio de la comunicación entre el usuario y servidor, teniendo la posibilidad de conseguir datos o alterarlos)

Obviamente que todo protocolo por más seguro que sea siempre puede llegar a estar abierto a sufrir ataques debido a vulnerabilidades de seguridad. SSH no es un protocolo el cual tenga muchas vulnerabilidades, pero no queda exento de tenerlas. Algunas de dichas vulnerabilidades son:

- ***Ataques de fuerza bruta y Malware:*** atacantes apuntas conseguir claves SSH para asi poder lanzar ataques de fuerza bruta hacia el servidor, o peor aún, lanzar ataques de Malware mediante la creación de puertas traseras utilizando las claves SSH.
- ***Acceso no autorizado o Secuestro de sesiones:*** es posible "secuestrar" una sesión de un usuario mediante la explotación de la comunicación establecida entre múltiples sistemas a través de la autenticación de la clave pública. También puede suceder al obtener acceso no autorizado al "socket" del usuario. Al secuestrar una sesión activa, el atacante logra obtener acceso al servidor, teniendo la posibilidad de generar alguna puerta trasera.
- ***Obtención de la Clave Privada por terceros:*** las claves privadas de SSH brindan acceso a funciones críticas del sistema. Cuando una clave privada es comprometida por algún tercero, el atacante puede acceder a todas las cuentas en donde se confíe dicha clave privada.

Como vimos anteriormente, SSH es un protocolo confiable y seguro, pero no está totalmente cerrado a ataques debido a malas prácticas por parte de los usuarios o vulnerabilidades de seguridad propias del protocolo. Por esto es que hay varias prácticas del protocolo las cuales vuelven a SSH mucho más seguro de lo que ya es. Algunas de estas prácticas son:

- ***Llevar un registro de Claves:*** llevar un registro de las claves privadas que dan acceso a SSH es considerado una práctica fundamental. El llevar un registro de estas claves y de cuáles usuarios tienen acceso a estas ayuda mucho a mantener el servidor más seguro.
- ***Creación y Rotación de Claves:*** hay quienes consideran que rotar las claves viejas de los usuarios con claves nuevas es una práctica clave a la hora de mantener un servidor, ya que puede disminuir los riesgos de compromiso con estas claves.
- ***Deshabilitar el "Root Login":*** el "root" es el usuario con el mayor acceso dentro del servidor. Un usuario root tiene todas las funciones como acceso a todo lo que esté en el servidor, por lo que deshabilitar el inicio de sesión en dicho usuario es algo primordial, ya que en caso de que un atacante acceda al servidor no seria a este usuario con acceso ilimitado, sino a uno con menor funcionalidad.

Funcionamiento

El protocolo SSH funciona de la siguiente manera: el cliente inicia una conexión con el servidor por SSH a través del puerto especificado (22 en este caso, por ser el predeterminado). Luego, el cliente y el servidor intercambian claves criptográficas para emplear el método de encriptación asimétrica. Este funciona a partir de que cada usuario tenga una clave pública y una privada. Al principio de la comunicación, ambos usuarios intercambian sus claves públicas, las cuales van a usar para encriptar los mensajes que le envíen al otro, y la clave privada la van a usar para desencriptar estos mensajes. Es decir, la clave pública y privada que generó cada cliente al principio, cumple las características de que con la clave privada se pueden desencriptar mensajes encriptados por la clave pública.

Historia

El protocolo SSH fue desarrollado en 1995 por Tatu Ylonen, un investigador en la Universidad de Tecnología de Helsinki, Finlandia. Fue desarrollado gracias a que la universidad habría sufrido un ataque cibernético a principios de año. En Julio del mismo año fue lanzado al público como un software gratuito de código abierto. Para finales de año se estima que el protocolo contaba con 20000 usuarios repartidos en 50 países, por lo que Ylonen recibía aproximadamente 150 emails diarios solicitando soporte técnico. Debido a esto Ylonen fundó "SSH Communications Security Corp" en donde continuará el desarrollo del protocolo y su comercialización hasta el día de hoy.

En 1996 lanzó el protocolo SSH 2.0 debido a problemas y limitaciones descubiertas las cuales no podrían haber sido resueltas sin causar incompatibilidad con SSH 1. Este nuevo protocolo incorporaba nuevos algoritmos incompatibles con SSH 1. En 1998 se lanzó SSH Secure Shell (SSH 2) basado en el protocolo SSH 2.0, sin embargo no logró reemplazar a SSH 1 ya que le faltaban algunas funcionalidades del protocolo anterior y contaba con una mayor restricción en sus licencias, por lo que los usuarios no sentían una necesidad a cambiar aunque el protocolo SSH 2 sea un protocolo mejor y más seguro.

Está situación cambió con el surgimiento de OpenSSH, una implementación gratuita del protocolo SSH 2. OpenSSH surgió como un software basado en la última versión gratuita del protocolo SSH 1.2.12 y se convirtió en una de las mayores implementaciones de SSH en el mundo. OpenSSH fue desarrollada por mucha gente debido a que es una alternativa libre y gratuita, pero su mayro contribuidor fue Markus Friedl. OpenSSH fue portada con éxito a Linux, Solaris, AIX, Mac OS entre otros sistemas operativos.

Implementación

A modo de demostración y experimentación es que desarrollamos un programa cuya misión es ingresar vía SSH a los servidores de OverTheWire para resolver de forma automática el tercer nivel del juego *Bandit*. El código que compone al ejercicio se encuentra en un repositorio privado de GitHub.

Este programa fue desarrollado en el lenguaje de programación Python, y hace uso de una biblioteca llamada Paramiko, la cual sirve precisamente para realizar conexiones por SSH. Esta biblioteca es realmente potente, y es gracias a ello que ha sido usada como base para construir, por ejemplo, otras bibliotecas de más alto nivel como puede ser Fabric (no aplicada en nuestra implementación). Asimismo, la misma tecnología fue desarrollada, en parte, con herramientas provistas por Cryptography, otra biblioteca de Python muy conocida que facilita algunos algoritmos de encriptado. Cabe decir que Paramiko es una biblioteca comúnmente utilizada (según la página web oficial de la organización) por usuarios avanzados o por aquellos que quieren crear un SSHD (Secure Shell Daemon).

Ahora bien, esto es, en resumen, lo que hace nuestro breve algoritmo paso a paso: Lo primero que se hace es crear el objeto *SSHClient*, que será nuestro *handler* de Paramiko para

todas las operaciones remotas. A continuación, se usa un método denominado *set_missing_host_key_policy*, cuyo propósito es pre-establecer un comportamiento para aquellas situaciones en las que el *host* al que nos queremos conectar no es reconocido. La política *AutoAddPolicy* automáticamente añade como conocido a cualquier *host* al que nos estemos conectando por primera vez.

Luego, guardamos en cuatro constantes los respectivos valores (datos necesarios para establecer la conexión, que son el nombre de *host*, puerto, nombre de usuario, contraseña), y establecemos la conexión al servidor con el método *connect*.

Lo que sigue es simplemente ejecutar los correspondientes comandos para resolver el desafío: Listar los contenidos de la carpeta *home*, encontrar la carpeta *inhere*, entrar a ella, listar los contenidos ocultos y no ocultos, hallar que existe *.hidden* y leerlo en consola.

NOTA: Para ejecutar el archivo, utilizar el comando “python3 main.py”. Previamente a la ejecución, debe ser instalada la biblioteca Paramiko: “pip3 install paramiko”.

Bibliografía

Artículos

https://es.wikipedia.org/wiki/Secure_Shell
Identifying and Mitigating Secure Socket Shell (SSH) Key Security Vulnerabilities
History of SSH - SSH, The Secure Shell: The Definitive Guide, 2nd Edition [Book] (oreilly.com)
Protocolo SSH: cómo funciona y cuáles son sus usos (redeszone.net)

Organizaciones

OverTheWire - https://www.overthewire.org/
Repositorio en GitHub: https://github.com/naranjujeli/tp-ssh/
Página web oficial de Python: https://www.python.org/
Biblioteca Paramiko: https://www.paramiko.org/
Biblioteca Fabric: https://www.fabfile.org/
Biblioteca Cryptography: https://cryptography.io/en/latest/