Endureciendo nuestro SSH

En otro de nuestros artículos iniciamos con el uso de la línea de comandos segura, en idioma inglés denominado «*Secure Shell*» y abreviado como **SSH.** También escribimos sobre Fail2ban, un sencillo programa para bloquear direcciones IP que realizan intentos fallidos de ingresar a nuestros servidores. Hoy completamos el tema de la seguridad, ¡veamos!

Introducción

Primero debemos saber dónde estamos parados: conozcamos el software que sostiene el mundo del SSH.

Servidores SSH

- Nuestro principal aliado: **OpenSSH** con un uso muy extendido en las distribuciones Linux (considerado una norma) e incluso más allá ya que Windows 10 ahora también lo utiliza *¿Quién lo hubiera creído?*
    - **Copssh** el cual es un software *portado* o mejor dicho **migrado** de OpenSHH al ambiente Microsoft Windows.
- En la Universidad del Oeste de Australia (*The University of Western Australia*) desde 1974 existe el Club de la Computadora de la Universidad (The University Computer Club) y allí labora el sr. Matt Johnston, creador del **Dropbear SSH**. Yo nunca había escuchado de este software (que por cierto aparte de servidor también corre como cliente) pero sí que conocía **OpenWRT**, el *firmware* ligero utilizado en millones de enrutadores alámbricos e inalámbricos (Wi-Fi). Ligero y sencillo de usar, sin mayores pretensiones. En este punto hago notar el caso de la vulnerabilidad **SSHowDowN** cuyo problema esencial fue -y sigue siendo- que los dispositivos para hogares (televisores, enrutadores, repetidores, cámaras de video vigilancia, incluso bombillos, etc.) les dejan las contraseñas por defecto de fábrica *y les permiten conexión al Internet* lo que ocasiona que personas maliciosas se hagan dueños de dichos artefactos para luego efectuar ataques "anónimos" contra sitios web como Twitter (año 2016).

- **Putty** que lo conocemos más en su uso como cliente que como servidor. Ambiente Microsoft Windows.
- **lsh** también como servidor y cliente, provisto por Debian pero muy poco utilizado debido a su complejidad de librerías y dependencias de software.

Existen otros como **CrushFTP Server** que sinceramente porque utilizan SSH para cifrar conexiones no lo alejan de su labor principal como servidor FTP; o el caso del **Apache MINA** que en realidad es un entorno de trabajo en lenguaje Java que utiliza SSH para sus API.

En cuanto a clientes, que ya vimos que también pueden funcionar como servidores, hay mucho software adicional que pueden conocer en este artículo de Wikipedia en inglés (quizás algún día lo traduzcamos allá al castellano y otros idiomas, no se descarta esa idea).

Librería SSH

Del lado del servidor podemos utilizar estas bibliotecas para que nuestros programas reciban órdenes directamente (como los ejemplos de **CrushFTP Server** y **Apache MINA**):

- **libssh** con el protocolo SSHv2 para lenguajes como **R, Perl** y **Python** y muy usado con servidores Git.
- **wolfSSH** creado por la empresa de Daniel Stenberg, el creador de cURL y escrito en lenguaje C ANSI.
- **paramiko** una librería exclusiva para lenguaje Python.

Recomendaciones

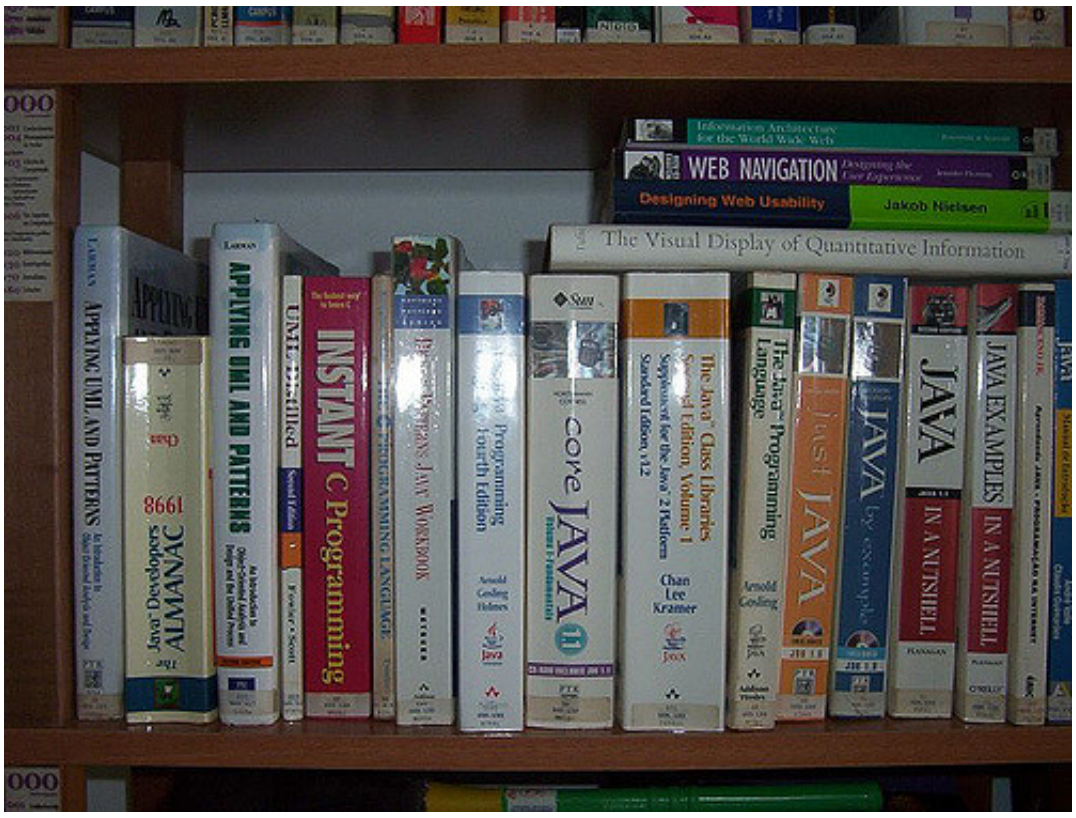
Ir a profundidad en este tema implica mucha, demasiada teoría y en realidad somos promotores de la simplicidad y practicidad. Planteamos, para comenzar, las líneas generales:

- Obvio, primero que nada no dejar usuarios y contraseñas por defecto en nuestros módem, enrutadores y concentradores. Esto también es válido para nuestros servidores y aparatos domésticos, lo cual nos lleva al siguiente punto.
- En nuestros enrutadores o servidores Linux dedicados al DHCP debemos siempre colocar grupos de direcciones IP sin acceso al Internet como regla por defecto, esto es patente en el caso de las redes inalámbricas. Es decir, pongamos como hecho que nos jaquean nuestra contraseña de Wi-Fi (o unos de nuestros usuarios se lo pasa a otra tercera persona): podrán conectarse a la red pero no tendrán salida al Internet, a menos que sean listos y se autoasignen una dirección IP que sí tengan salida a Internet. Pero los

dispositivos como cámaras de videovigilancia, etc. no tendrán este comportamiento. Desde luego, también debemos definir un rango de direcciones IP fijas con acceso a Internet y otra sin acceso a Internet, dependiendo de lo que necesitemos agregaremos en uno u otro grupo.

- Otra cosa que debemos realizar es basar la autenticación en clave pública y eliminar la conexión por contraseña ya que los programas destinados para jaquear SSH miden el tiempo que tarda en devolver respuesta a un usuario especificado: si tarda mucho es porque el usuario existe (y el servidor SSH tarda porque calcula el cifrado de la contraseña enviada) y esa muy pequeñísima diferencia abre el camino para un ataque de fuerza bruta.
- Dado el caso que nos realicen un ataque de fuerza bruta pues para ello Fail2ban, del cual ya comentamos al principio de este artículo.

También podemos usar una herramienta como **ssh-audit** si queremos realizar una práctica, evaluar los consejos anteriores, con la ventaja que podemos analizar el código fuente para mejorar nuestros conocimientos y habilidades.



Fuentes consultadas

En idioma castellano

- «».
- «».
- «».

En idioma francés

- «».
- «».
- «».

En idioma inglés

- «OpenWrt».
- «Dropbear SSH».
- «SSHowDowN: Exploitation of IoT devices for Launching Mass-Scale Attack Campaigns».
- «Copssh - OpenSSH for Windows».
- «SSH Pentesting Guide».

Crédito de la imagen Gerd Altmann, trabajo, licencia de uso:Pixabay