Reto SSH

6 marzo 2022

línea corta

# Objetivo

Un CTF es una competición que consiste en capturar una bandera escondida en una máquina expuesta en Internet. Vuestro objetivo será alcanzar un fichero (bandera 1) donde tendréis que escribir vuestro nombre y un servidor web (bandera 2) donde tendréis que subir el archivo *index.html* que queráis.

Con este reto se pretende que los alumnos adquieran algunos conocimientos de seguridad y, más concretamente, de SSH, así como conocimientos de otras ramas mientras superan una serie de pruebas que se han establecido en distintas máquinas virtuales del centro.

# Descripción

El reto consiste en realizar una serie de conexiones SSH en máquinas sucesivas hasta llegar a una máquina final donde poder grabar vuestro nombre y subir el fichero *index.html.* Para conseguirlo tendréis que superar una serie de pruebas que os permitirán acceder a cada máquina.

En [github](https://github.com/navasdamas/CTF-SSH) disponéis de un manual muy completo sobre SSH y las herramientas que se utilizan en este reto. Al final de este documento se os ha incluido un resumen con todo lo que necesitáis para superar las pruebas.

Para comenzar el reto necesitaréis crear una araña en Java que consiga descifrar el usuario y contraseña de la primera máquina, en ella encontraréis las instrucciones para acceder a la siguiente.

El primer alumno que grabe su nombre y suba el fichero index.html será el ganador del reto. Cuando consideréis que habéis terminado el reto, poneros en contacto con algunos de los profesores para que verifique que se ha superado correctamente.

# Instrucciones

En la primera prueba tendréis que acceder mediante SSH a una máquina del centro que se encuentra en la dirección **falcondptoinformatica.synology.me** y en el **puerto 22**. El usuario de la máquina es el nombre de uno de los profesores que os imparte clase: Eladio, Ana, Loli, Guillermo, Antonio, Carlos. La contraseña de este usuario se encuentra entre las 200 contraseñas más utilizadas en España en el año 2020: <https://nordpass.com/es/most-common-passwords-list/>

En [github](https://github.com/navasdamas/CTF-SSH) tenéis un fichero llamado *pass\_csv.csv* donde se encuentra este listado de contraseñas en formato csv.

Para superar esta prueba deberéis de crear una araña en Java que realice un ataque de diccionario probando con cada usuario y cada contraseña hasta que consigáis la combinación correcta. Este programa os imprimirá por pantalla cuáles son las credenciales correctas y deberéis de realizar la conexión SSH desde el terminal de vuestro equipo. Una vez dentro de la máquina, podéis abrir o descargarlos el fichero *instrucciones\_maquina\_2.txt* para conocer las instrucciones de la siguiente prueba.

# Material necesario

* [IDE Java](https://www.jetbrains.com/es-es/idea/download/#section=windows)
* [Java JDK](https://www.oracle.com/java/technologies/downloads/)
* Librería SSH de Java: [JSch](https://mvnrepository.com/artifact/com.jcraft/jsch/0.1.55)
* Cliente SSH
* [NMap](https://nmap.org/download)

# Pistas

## Activar el cliente OpenSSH en Windows 10/11

Para instalar los componentes de OpenSSH:

1. Abre **Configuración**, selecciona **Aplicaciones > Aplicaciones y características** y, a continuación, seleccione **Características opcionales**.
2. Examina la lista para ver si **OpenSSH** ya está instalado. Si no es así, en la parte superior de la página, selecciona **Agregar una característica** y, a continuación:

* Busca **OpenSSH Client** (Cliente OpenSSH) y, a continuación, haga clic en **Instalar**
* Busca **OpenSSH Server** (Servidor OpenSSH) y, a continuación, haga clic en **Instalar**

Una vez completada la instalación, vuelve a **Aplicaciones > Aplicaciones y características y Características opcionales** y debería ver **OpenSSH** en la lista.

**Nota**: La instalación del servidor OpenSSH creará y habilitará una regla de firewall denominada OpenSSH-Server-In-TCP. Esto permite el tráfico SSH entrante en el puerto 22. Si esta regla no está habilitada y este puerto no está abierto, las conexiones se rechazarán por el firewall.

## Conexiones SSH con Java

En [github](https://github.com/navasdamas/CTF-SSH) disponéis de un esqueleto de programa en Java llamado *Java-SSH* donde se muestra cómo se realiza una conexión SSH utilizando la librería JSch. Si queréis ver en más detalle como se instala y utiliza la librería, podéis consultar el ***Anexo III – SSH utilizando Java*** del manual.

## Conexiones SSH desde consola con usuario y contraseña

Para llevar a cabo la conexión SSH mediante consola utilizando usuario y contraseña usaremos el siguiente comando:

ssh **<**usuario**>@<**direccion-ip**>** -p **<**puerto**>**

Donde:

* **Usuario** es el nombre de usuario con el que queremos conectarnos al equipo remoto.
* **Dirección ip** es la dirección del servidor SSH con el que vamos a establecer la conexión.
* **Puerto** es el puerto en el que debe estar en escucha el servicio SSH en la máquina remota, que por defecto y si no se indica nada en el comando es el 22.

A continuación se muestra un ejemplo de conexión SSH mediante el método de autenticación por usuario y contraseña:

Texto

Descripción generada automáticamente

Si queréis saber con más detalle cómo utilizar SSH, podéis consultar el apartado ***5. Utilización de SSH*** del manual.

## Instalación y uso de Nmap

Para utilizar Nmap en Windows basta con bajarnos el instalador desde su página web e instalarlo en nuestro equipo: [Nmap](https://nmap.org/download#windows)

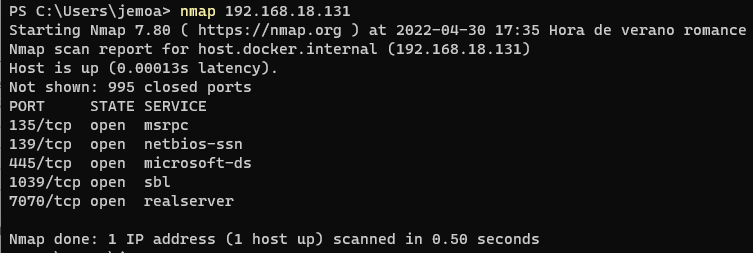
En el caso de Linux podemos instalarlo ejecutando en consola:

sudo apt-get install nmap

Para utilizarlo debemos de abrir un terminal y escribir:

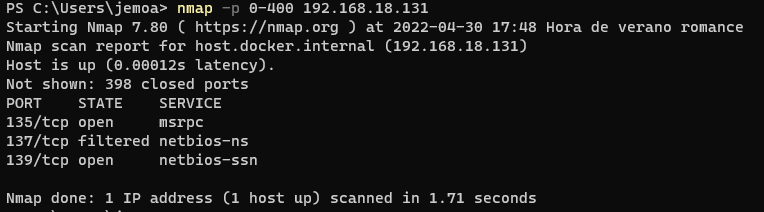
namp **<**dirección-ip**>**

Esto analizará los **puertos más comunes** de la dirección ip introducida y nos dirá cuales tiene abiertos. Además, nos indicará el servicio que **considera** que se está ejecutando en dicho puerto.



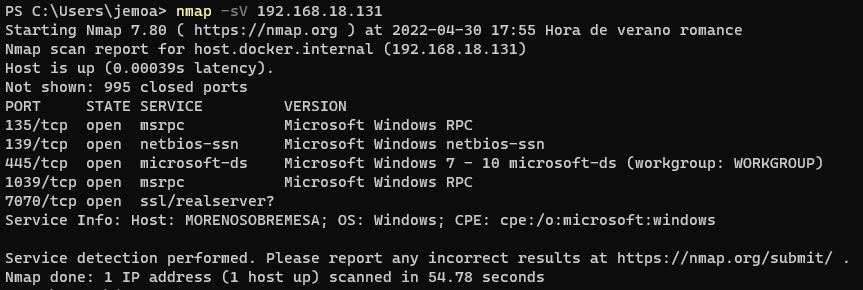
Si queremos forzar que Nmap busque en todos los puertos de un rango dado, podemos hacerlo con el siguiente comando:

namp -p **<**puerto-inicio**>-<**puerto\_fin**> <**dirección-ip**>**



Por último, podemos forzar a Nmap a que nos diga exactamente qué servicio se está ejecutando en cada puerto. Este comando tarda más en ejecutarse, pero nos puede ser útil si necesitamos encontrar un servicio en concreto que sabemos que no se está ejecutando en su puerto habitual.

namp -sV **<**dirección-ip**>**



Si queréis más información acerca de Nmap, podéis consultarlo en el ***Anexo VI – Instalación y uso de Nmap*** del manual

## Transferencia de ficheros mediante SCP

El comando *scp* nos permite transferir un fichero entre un origen y un destino con una sintaxis sencilla.

scp **<**usuario**>@<**direccion-ip-origen**>**:**<**ruta-fichero**> <**usuario**>@<**direccion-ip-destino**>**:**<**ruta-guardado**>**

Nota: Si el origen o el destino es nuestra máquina local, no tenemos que especificar ni el usuario ni la dirección ip, bastaría con especificar la ruta del fichero,

Veamos el siguiente ejemplo, supongamos que tenemos un archivo pelota.txt en la carpeta personal del usuario mnavas de nuestra máquina remota, y queremos transferir dicho archivo al directorio actual de nuestra máquina local. Para ello ejecutaremos el siguiente comando:

scp mnavas**@**192**.**168**.**0**.**26**:**pelota.txt **.**

Texto

Descripción generada automáticamente

Si queréis más información sobre como utilizar *scp* podéis consultar el punto *7 – Transferencia de ficheros* del manual. También podéis consultar el ***Anexo IV – Transferencia de ficheros mediante SFTP en Java*** para ver una forma de realizar la transferencia de ficheros segura utilizando Java y la librería JSch.

## Peticiones GET

Con el método GET, los datos que se envían al servidor **se escriben en la misma dirección URL**.

Para enviar una petición GET a través del navegador hay que especificar:

* La URL donde queremos mandar la información.
* El puerto donde se está ejecutando el servidor web.
* La ruta del servidor donde queremos mandar la petición.
* Parámetro/s (información queremos enviar.

Si en el servidor, la ruta especificada está configurada para recibir un parámetro, este procesará la petición y enviará al cliente una respuesta 200 (OK)

**<**protocolo**>**://**<**URL**>:<**puerto**>/<**ruta**>/<**parámetro**>**

Por ejemplo, supongamos que tengo un servidor cuya URL es **mi-server.com** y está corriendo en el puerto **8080** de mi ordenador. En este servidor he creado la ruta **/saludo** que espera recibir un **string** por parte del usuario. Para enviar esta petición a través del navegador se haría de la siguiente manera:

http://mi-server.com:8080/saludo/hola\_server

También se pueden realizar peticiones GET a través de Java. Para ello, en [github](https://github.com/navasdamas/CTF-SSH) disponéis de un proyecto de ejemplo llamado Java\_GET que la librería Apache HttpClient que se utiliza para realizar peticiones Http desde Java