

# Reto SSH

6 marzo 2022

---

## 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) 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](#) 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](#)
- [Java JDK](#)
- Librería SSH de Java: [JSch](#)
- Cliente SSH
- [NMap](#)

## 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](#) 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:

```
C:\Users\mnava>ssh mnavas@192.168.0.26
The authenticity of host '192.168.0.26 (192.168.0.26)' can't be established.
ECDSA key fingerprint is SHA256:0XCmny0RsV46Nfn0qCeoBiEfKBbECsY3YJ5ZBYxOLT8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.26' (ECDSA) to the list of known hosts.
mnavas@192.168.0.26's password:
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-25-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Se pueden aplicar 0 actualizaciones de forma inmediata.
mnavas@vmubuntu:~$
```

Se nos abre la shell en la máquina remota

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

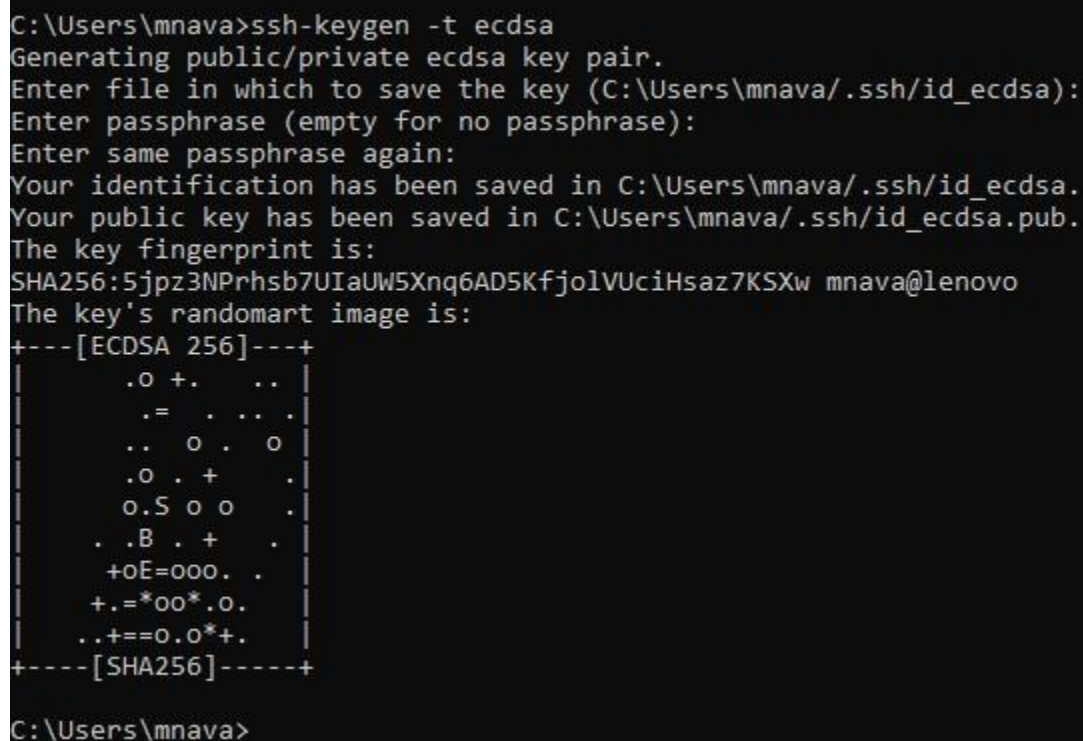
## Autenticación mediante clave pública

Para llevar a cabo el proceso de autenticación mediante clave pública tendremos que

1. Generar una clave privada del cliente.
2. Obtener a partir de dicha clave la clave pública asociada.
3. Copiar esta clave pública de nuestro cliente en el servidor SSH al que queremos conectarnos.

Para generar el par de claves pública/privada en nuestro cliente utilizaremos el comando `ssh-keygen` indicando el tipo de algoritmo que queremos utilizar:

```
ssh-keygen -t ecdsa
```



```
C:\Users\mnava>ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (C:\Users\mnava/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\mnava/.ssh/id_ecdsa.
Your public key has been saved in C:\Users\mnava/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:5jpbz3NPrhsb7UIaUW5Xnq6AD5KfjolVUciHsaz7KSXw mnava@lenovo
The key's randomart image is:
+---[ECDSA 256]---+
|      .o +.  ..  |
|      . =   . . . |
|     .. o .  o   |
|     .o . +    .  |
|     o.S o o    . |
|    . .B . +    . |
|    +oE=ooo.    . |
|    +.=*oo*.o.   |
|    ..+=o.o*+.   |
+-----[SHA256]-----+
C:\Users\mnava>
```

Como se puede observar en la imagen anterior, al ejecutar el comando `ssh-keygen` en primer lugar nos pregunta con que nombre queremos almacenar los archivos de claves que se generarán. Por defecto se utilizará el nombre del algoritmo que hemos seleccionado para generar los archivos de clave pública y privada: `id_<nombre-algoritmo>` y `id_<nombre_algoritmo>.pub`. Por tanto, si no introducimos nada se nos guardarán con estos nombres. Después de esto nos pide

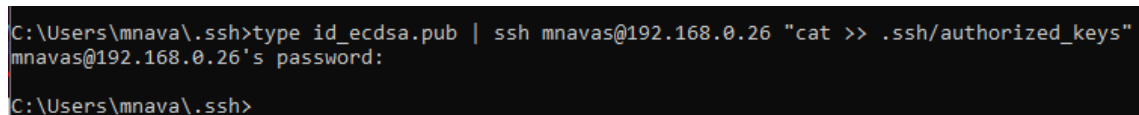
que introduzcamos una clave de paso (o frase de paso) para las claves. Si hacemos esto, a la hora de utilizar la clave para iniciar sesión nos pedirá esta frase de paso.

El siguiente paso, como hemos comentado es el de copiar la clave pública que nos ha generado el comando `ssh-keygen` en el servidor SSH. Para llevar a cabo esta tarea de una forma remota, sencilla y rápida `ssh` nos proporciona un comando que nos permite copiar nuestra clave pública autenticándonos de forma remota:

```
ssh-copy-id -i id_ecdsa <usuario>@<direccion-ip> -p <puerto>
```

**Importante:** para usuarios de Windows la utilidad `ssh-copy-id` no está disponible. Por lo tanto para llevar a cabo la copia de la clave pública al fichero correspondiente del servidor de la manera más cómoda posible debemos utilizar el siguiente comando:

```
type id_ecdsa.pub | ssh <usuario>@<direccion-ip> -p <puerto> "cat >> .ssh/authorized_keys"
```



```
C:\Users\mnava\.ssh>type id_ecdsa.pub | ssh mnavas@192.168.0.26 "cat >> .ssh/authorized_keys"
mnavas@192.168.0.26's password:
C:\Users\mnava\.ssh>
```

Una vez tenemos la clave pública copiada en el servidor y nosotros disponemos de la clave privada, podremos establecer la conexión SSH sin necesidad de introducir la contraseña utilizando el comando:

```
ssh -i ~/.ssh/miclaveprivada <usuario>@<direccion-ip> -p <puerto>
```

Si queréis más información sobre la autenticación mediante clave pública podéis consultar el punto **5.3 Autenticación mediante clave pública** 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](#)

En el caso de Linux podemos instalarlo ejecutando en consola:

```
sudo apt-get install nmap
```

Para utilizarlo debemos de abrir un terminal y escribir:

```
nmap <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.

```
PS C:\Users\jemoa> nmap 192.168.18.131
Starting Nmap 7.80 ( https://nmap.org ) at 2022-04-30 17:35 Hora de verano romance
Nmap scan report for host.docker.internal (192.168.18.131)
Host is up (0.00013s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1039/tcp  open  sbl
7070/tcp  open  realserver

Nmap done: 1 IP address (1 host up) scanned in 0.50 seconds
```

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

```
nmap -p <puerto-inicio>-<puerto-fin> <dirección-ip>
```

```
PS C:\Users\jemoa> nmap -p 0-400 192.168.18.131
Starting Nmap 7.80 ( https://nmap.org ) at 2022-04-30 17:48 Hora de verano romance
Nmap scan report for host.docker.internal (192.168.18.131)
Host is up (0.00012s latency).
Not shown: 398 closed ports
PORT      STATE SERVICE
135/tcp   open  msrpc
137/tcp   filtered netbios-ns
139/tcp   open  netbios-ssn

Nmap done: 1 IP address (1 host up) scanned in 1.71 seconds
```

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.

```
nmap -sV <dirección-ip>
```

```

PS C:\Users\jemoa> nmap -sV 192.168.18.131
Starting Nmap 7.80 ( https://nmap.org ) at 2022-04-30 17:55 Hora de verano romance
Nmap scan report for host.docker.internal (192.168.18.131)
Host is up (0.00039s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE      VERSION
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds (workgroup: WORKGROUP)
1039/tcp  open  msrpc        Microsoft Windows RPC
7070/tcp  open  ssl/realserver?
Service Info: Host: MORENOSOBREMESA; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 54.78 seconds

```

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 .

```

```
C:\Users\mnava\TransferenciaFicheros>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 54B4-35FF

Directorio de C:\Users\mnava\TransferenciaFicheros
25/04/2022  20:25    <DIR>        .
25/04/2022  20:25    <DIR>        ..
               0 archivos             0 bytes
               2 dirs  189.445.820.416 bytes libres

C:\Users\mnava\TransferenciaFicheros>scp mnava@192.168.0.26:pelota.txt .
mnava@192.168.0.26's password:
pelota.txt                                           100%  54  32.6KB/s  00:00

C:\Users\mnava\TransferenciaFicheros>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 54B4-35FF

Directorio de C:\Users\mnava\TransferenciaFicheros
25/04/2022  20:25    <DIR>        .
25/04/2022  20:25    <DIR>        ..
25/04/2022  20:25    54 pelota.txt
               1 archivos             54 bytes
               2 dirs  189.445.812.224 bytes libres

C:\Users\mnava\TransferenciaFicheros>
```

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](#) disponéis de un proyecto de ejemplo llamado Java\_GET que la librería Apache HttpClient que se utiliza para realizar peticiones Http desde Java.