LABORATORIO DE REDES Y SERVICIOS I

Práctica 1 INTRODUCCIÓN A LINUX, VIRTUAL BOX Y WIRESHARK

Tiempo Previsto: 3 sesiones

## Obteniendo información de los comandos

1. La mayoría de los comandos admiten el parámetro **--help** o **-h**. Obtener información de los comandos **ls**, **cat** y **tee**.
2. También la mayoría de los comandos disponen de una página de manual a la que se puede acceder ejecutando **man <COMANDO>**. Para salir del mismo hay que pulsar la tecla **q**. Obtener información de los comandos anteriores.

## Gestión de procesos

El comando para visualizar los procesos es **ps**. Con el parámetro **-ef** podremos ver todos los procesos que se ejecutan en el sistema.

1. En una terminal ejecutaremos **sleep 500** (esperar 500 segundos) y en otra ejecutaremos **ps –ef|grep sleep** para localizarlo.

2. ¿Qué procesos está ejecutando nuestro usuario (**ubuntu**)?

Con **pgrep** obtenemos el PID (identificador) del proceso que le pasemos como argumento.

1. ¿Qué PID tiene el proceso asociado al **sleep**? Es posible que haya que volver a ejecutarlo.

El comando **kill** manda por defecto la señal de apagado (equivalente a pulsar **CTRL+C**) al proceso cuyo identificador de proceso (PID) le pasemos.

1. Apagar el proceso del **sleep**.
2. Al comando **pkill** sólo se le debe pasar el nombre del proceso y no su identificador. Volver a ejecutar **sleep 500** y matarlo con **pkill**.

Podemos ejecutar procesos en segundo plano al pasarle al completar el comando con el símbolo **&**.

1. Lanzar **sleep 500** en segundo plano. Ponerlo en primer plano ejecutando **fg**.
2. Lanzar **sleep 500** en primer plano. Parar el proceso con **CTRL+Z** y luego pasarlo a segundo plano con **bg**.

## Sistema de ficheros

El comando **ls** nos permite visualizar el contenido del directorio que le pasemos como parámetro (por defecto, en el que estamos ubicados).

1. Listar el contenido del directorio raíz (**/**).
2. Averiguar qué parámetro se necesita para obtener la mayor información en el listado.
3. Averiguar qué parámetro permite mostrar los archivos ocultos (probar en el directorio del usuario: **/home/ubuntu/**)

El comando **cd** nos desplaza por el árbol de directorios y **pwd** nos muestra en qué directorio estamos.

Las rutas absolutas son aquellas que parten del directorio raíz y por tanto comienzan por **/**. Las relativas lo son desde el directorio en el que nos encontramos y por tanto no empiezan por **/**.

Si ejecutamos **cd** sin suministrarle un parámetro, iremos al directorio del usuario.

1. Desplazarnos al directorio **/etc** dando su dirección absoluta y ejecutar **pwd**.
2. Desplazarnos al directorio raíz y de ahí, al **/etc** usando su dirección relativa. Ejecutar **pwd**.

El comando **mkdir** crea un directorio con el nombre que le facilitemos.

El comando **mv** mueve un archivo/directorio desde un origen a un destino que le pasemos como parámetro. Por otro lado, **cp** copia el archivo/directorio.

El comando **rm** borra archivos.

1. Crear dos directorios en el directorio del usuario llamados **nsa** y **3ech**.
2. Mover **3ech** dentro de **nsa**.
3. Ejecutar **touch** para colocar un archivo llamado **bug** dentro de **nsa**.
4. Ejecutaremos **rm *** (borrar todo) dentro de **nsa**. ¿Qué sucede?
5. Volveremos a colocar el archivo **bug** y esta vez borraremos con los parámetros **-Rf**.

**IMPORTANTE:** El comando **rm * -Rf** ejecutado en un lugar equivocado puede tener consecuencias catastróficas.

## Redirección y encadenamiento de comandos

Con el símbolo **>** redirigimos la salida estándar de un comando a un archivo que indiquemos. Si usamos **2>**, se redirigirá la salida de error.

1. Redirigir la salida del listado de archivos que hay en el raíz a un archivo llamado **output** en el directorio del usuario.
2. Redirigir la salida de error de visualizar el archivo **/etc/shadow** con el comando **cat** a un archivo llamado **error** en el directorio del usuario.

Podemos encadenar la salida de un comando con la entrada de otro empleando el símbolo **|** (conocido también como pipe).

1. Redirigir la salida del listado de archivos que hay en el raíz a un archivo llamado **output** en el directorio del usuario.

## Visualización y edición de archivos de texto

El comando **more** se emplea para visualizar archivos de texto largos (que ocupan más de una pantalla). Con **retorno** y **f** avanzamos páginas, con **espacio** avanzamos líneas y con **b** retrocedemos páginas. Para salir, pulsaremos **q**.

1. Visualizar el fichero **/var/log/dmesg** y navegar por él.

Tenemos varios editores para elegir. Entre ellos están **nano** y **vi**. Es fundamental hacernos a uno de estos para las prácticas.

1. Crear un archivo de texto en nuestro directorio de usuario.
2. Introducir nuestros datos personales (uno por línea), guardar los cambios y salir.
3. Volver a abrir el archivo y borrar la penúltima línea. Guardar y salir.
4. Añadir **(RyS)** a continuación del nombre (en la misma línea). Guardar y salir.

## Creación de usuarios

El comando usado para crear usuarios es **adduser**. Se debe ejecutar como **root** a través del comando **sudo**.

Se pide dar de alta a los siguientes usuarios (se suministra el nombre real), asignarles contraseñas distintas, introduciendo lo que se desee en el resto de información:

- kflynn: Kevin Flynn
- abradley: Alan Bradley
- edillinger: Ed Dillinger
- lbaines: Lora Baines

1. Verificar que las entradas correspondientes se han añadido a los ficheros **/etc/passwd** y **/etc/shadow**.
2. Desde la cuenta de **root** cambiar a cualquiera de los usuarios creados usando el comando **su**.
3. Desde la cuenta **ubuntu** cambiar al mismo usuario.

| ¿Qué diferencia se aprecia en el método descrito en el paso 2 y el 3? |
|---|

## Creación de grupos de usuarios

El comando usado para crear grupos de usuarios es **addgroup**. Se pide crear los siguientes grupos:

- employees
- developers
- assistants

1. Verificar que los grupos se han creado en **/etc/group**.
2. Crear un nuevo usuario llamado **sflynn** con las siguientes características:
   - Nombre real: Sam Flynn
   - Grupo: employees

   **CONSEJO:** Para borrar el usuario se usará el comando **deluser**.

3. Cambiar al nuevo usuario y verificar con el comando **id** que efectivamente pertenece al grupo.

| ¿Por qué no hay un grupo llamado **sflynn** en **/etc/group**? |
|---|

## Modificación de usuarios y grupos

Uno de los aspectos que se puede modificar es la contraseña. Esto se realiza con el comando **passwd**. El comando **chage** establece los detalles de expiración de la contraseña.

El comando **usermod** permite mucho más nivel de modificación.

1. Modificar las contraseñas de dos de los usuarios creados con **passwd**. La contraseña de uno de ellos será igual al nombre de usuario.
2. Modificar las contraseñas del resto de usuarios creados con **usermod**. La contraseña de uno de ellos será igual al nombre de usuario.
3. Asignar en los siguientes grupos a los usuarios correspondientes:
   - employees: todos
   - developers: kflynn, abradley, edillinger
   - assistants: edillinger, lbaines
4. Verificar los cambios realizados tanto desde los usuarios (comandos **id** y **groups**) como en el fichero **/etc/group**.
5. Bloquear la cuenta **sflynn**.
6. Establecer la expiración de la contraseña de **kflynn** a mañana.

| ¿En qué archivo se puede verificar que la cuenta **sflynn** ha sido bloqueada? ¿Y la expiración de la contraseña de **kflynn**? |
|---|
| ¿Qué símbolos/campos se suelen emplear y para qué situaciones? |

## Gestión de permisos de archivos y directorios

En esta parte se trabajará en el directorio **/tmp**. Los permisos se cambian con el comando **chmod**. El propietario se cambia con los comandos **chown** y **chgrp**.

1. Como usuario **ubuntu** crear un directorio llamado **Trace**. Verificar los permisos.
2. Dentro del directorio, crear un archivo de texto llamado **data** con el nombre del alumno. Verificar los permisos.
3. Dejar los permisos de lectura del archivo **data** sólo para el usuario propietario.
4. Leer el archivo **data** como los usuarios **edillinger** y **root**.

| ¿El resultado es el mismo? ¿Por qué? |
|---|

5. Crear un archivo de texto llamado **script.pl** con el siguiente contenido:

```
#!/usr/bin/perl

open FILE,"data" or die $!;

$data=<FILE>;

close(FILE);

 chomp($data);

 print "$data is the new Space Paranoids Champion!\n";
```

6. Probar a ejecutar el archivo.

| ¿Qué es lo que falla? Solucionar el problema. |
|---|

7. Cambiar el grupo propietario de **script.pl** a **employees**. Verificar los permisos.
8. Ejecutar **script.pl** como el usuario **edillinger**.

| ¿Cuál es el resultado? ¿Por qué? |
|---|
| Solucionar el problema para que **edillinger** pueda ejecutar exitosamente **script.pl** |
| ¿Qué usuarios pueden cambiar los propietarios y permisos de los archivos? |

9. Cambiar el propietario del directorio **Trace** tanto al usuario como al grupo **edillinger**.
10. Dejar los permisos del directorio exclusivamente para el usuario propietario.
11. Con otro usuario, intentar acceder al directorio.
12. Cambiar el grupo propietario del directorio a **developers** y darle permisos de lectura.
13. Como **abradley**, listar el contenido de **Trace** pero **sin acceder a él**.
14. Añadir el permiso de escritura del directorio para el grupo **developers**.
15. Como **abradley**, intentar crear un nuevo archivo de texto con el apellido del alumno.
16. Añadir el permiso de ejecución del directorio para el grupo **developers** y probar a crear el archivo de texto de nuevo.

| ¿Cuál es la traducción de los permisos de los ficheros (rwx) a los directorios? |
|---|

El comando **chmod** permite también asignar, entre otras cosas, el bit de SUID (u+/-s) y GUID (g+/-s).

1. Cambiar el usuario y grupo propietarios de **script.pl** a **root**.
2. Añadir las siguientes líneas al archivo **script.pl**:

```
$ENV{"PATH"} = "/usr/bin";

system ("tail /etc/shadow");

system (id);
```

3. Ejecutar **script.pl** como **edillinger**.

   **CONSEJO:** revisar las opciones del comando **sudo**.

4. Asignar el bit SUID a **script.pl**. Revisar los permisos del archivo.
5. Volver a ejecutar **script.pl** como **edillinger**.

| ¿Qué diferencias se aprecian? |
| --- |
| ¿Qué implicaciones para la seguridad tienen los bits SUID y GUID? |

El comando **umask** permite establecer los permisos por defecto de los archivos y directorios que se crearán. Al igual que **chmod**, este comando tiene dos modos de introducir los parámetros: octal (complemento, permisos que no se establecerán) y texto (ejemplo: u=rwx,g=rwx,o=).

1. Verificar los permisos por defecto actuales. Crear un fichero (con el comando **touch**) y verificar la correspondencia con la máscara.
2. Cambiar la máscara a 0007. Crear un nuevo fichero y verificar los permisos.
3. Cambiar la máscara a 0002. Crear un nuevo fichero y verificar los permisos.
4. Cambiar la máscara a 0077. Crear un nuevo fichero y verificar los permisos.
5. Volver a la máscara original (0022) **pero empleando el texto**.

| ¿En qué situaciones son mejores cada máscara de las vistas anteriormente? |
| --- |

## Gestión de permisos de ejecución con sudo

Se puede definir qué usuario puede ejecutar qué y como qué usuario gracias a **sudo**. Para ello, se tiene que editar el archivo **/etc/sudoers**. Este archivo puede ser editado con una a través del comando **visudo** (en la máquina de las prácticas carga el editor **nano**).

1. Quitar el bit SUID de **script.pl**. Comprobar que el resultado de su ejecución ha cambiado.
2. Ejecutar como **edillinger**: **sudo /tmp/Trace/script.pl** (**./script.pl** si se está en el mismo directorio).
3. En el archivo **/etc/sudoers**:
   - Agregar a **edillinger** al alias ENCOM:

     *# User alias specification*

     User_Alias ENCOM = edillinger

   - Añadir el permiso de ejecución como **root** a **script.pl**:

     ENCOM ALL = /tmp/Trace/script.pl
4. Volver a ejecutar **script.pl** como **edillinger**.
5. Dar permisos a **abradley** para ejecutar **script.pl** como **root**.

| Poner algún ejemplo de cómo **sudo** puede ayudar en las tareas de mantenimiento del sistema. |
| --- |

## Gestión de ACL

La ACL de un archivo o directorio se puede gestionar mediante los comandos **getfacl** y **setfacl**. Además, se cuenta con un entorno gráfico llamado **Eiciel** (invocable con el comando **eiciel**).

1. Por línea de comandos, verificar la ACL de **script.pl**.
2. Abrir **script.pl** en **eiciel**.
3. Quitar todos los permisos a **otros**.
4. Dar permiso sólo de ejecución a **edillinger** y **abradley**.
5. Dar permiso sólo de lectura al grupo **assistants**.
6. Verificar  todos los cambios anteriores.

¿Qué diferencias tiene la gestión de ACL respecto a la vista con **sudo**?

## VISUALIZACIÓN DE PUERTOS

Abrir tres consolas. Realizar un chat sencillo utilizando NETCAT entre dos de ellas. En la tercera, comprobar la apertura de puertos y procesos utilizando NETSTAT y PS. Apuntar los resultados.

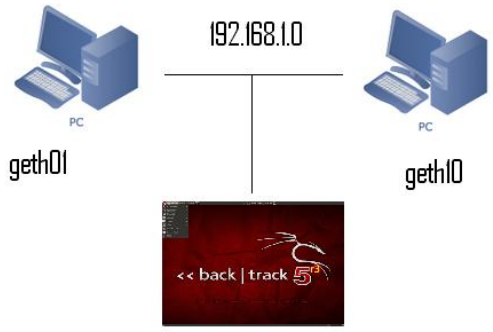
## CREACIÓN DE LOS SIGUIENTES ESCENARIOS EN VIRTUAL BOX

- Configurar los interfaces usando el commando **ifconfig**
- Router es el gateway por defecto del resto de elementos:
  **route add default gw <IP_Router> <Interfaz>**
- Recordar activar el IP forwarding en el router. Para activarlo, ejecutar :
  **sysctl -w net.ipv4.ip_forward=1**
- Comprobar la configuracion utilizando ICMP (ping)
- Backtrack ha de estar configurado en modo promiscuo.
- Utilizar Wireshark (backtrack) para registrar las comunicaciones entre geth01 y geth10.
    - Limpiar la cache ARP de todos los interfaces
    - Realizar un chat sencillo utilizando NETCAT
    - Abrir una sesión TELNET desde geth01 en geth10 y ejecutar algunos comandos
    - Abrir una sesión FTP desde geth01 en geth10 y transferir un archivo de texto
    - Abrir una sesión son SSH.

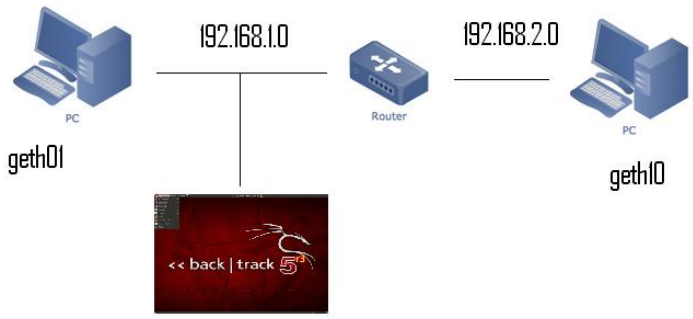
Apuntar para cada escenario:

- Direcciones MAC e IP de los paquetes
- Recomponer el Flujo GENERAL
- Gráfico del flujo TCP
- Conclusiones generales

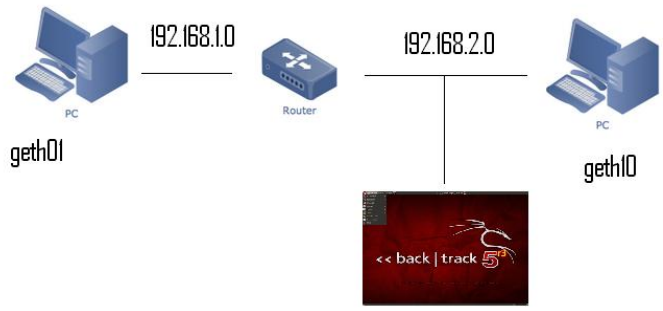
Escenario 1



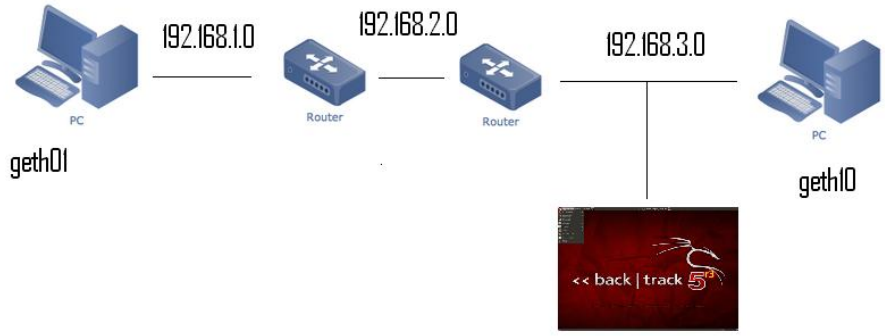
Escenario 2 A




Escenario 2 B




Escenario 3 A




Escenario 3 B