Practica-2-Sesion-3.pdf

SOLUCIÓN Práctica 2 - Sesión 3

3º Ingeniería de Servidores

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
UGR - Universidad de Granada

Ingeniería de Servidores

Práctica 2

Sesión 3

En esta sesión vamos a proceder con la instalación de Apache, la instalación de un servicio de páginas web. Recordemos conceptos como HTTP (HyperText Transfer Protocol) o HTML (HyperText Markup Language).

Con Ubuntu y la red host-only configurada, arrancamos la máquina. Cuando abrimos una web, no sabemos qué pasa debajo. Podemos usar `netcat` para averiguarlo; básicamente, abre un socket, comunicando la entrada y salida estándar y mostrándola por pantalla. Lo comprobamos con `nc www.example.com 80 -v` (80 es el puerto y -v es el verbose, que nos muestra una información más descriptiva sobre sus funciones). Con esto conseguimos conectarnos **al servidor** donde está alojada la web. Ahora hay que poner `GET http://www.example.com HTTP/1.0` (damos dos veces intro) y se descarga la página tras darle esa orden al servidor. Solo vemos el código HTML (dentro de las siglas HTML, "markup" significa que las etiquetas que se usan en el código no son visibles, pero alteran de igual forma el contenido o formato del documento -o web, como lo conocemos comúnmente-).

PHP es un elemento importante que hace la ejecución del código en el servidor (de modo que por ejemplo no se use JavaScript para una tarea como es la comprobación de una contraseña, pues entonces sería código que se ejecutaría en el cliente y esto daría lugar a problemas inimaginables de seguridad).

Para la instalación del **stack**, podemos usar `tasksel` (instala varias cosas de un mismo comando). La desventaja es que hará una instalación genérica y puede instalar más funcionalidades de las que necesitamos. Como hace tiempo que no lo consultamos y no sabemos qué espacio hay libre en el disco, podemos usar `df -k` para comprobarlo. Ejecutaremos `tasksel` con `sudo tasksel` y seleccionamos el paquete **`LAMP`**. Una vez instalado, comprobamos si el daemon (servicio) está ejecutándose con `systemctl status apache2.service` (recordemos que en Ubuntu, al instalar algo, se habilita y ejecuta automáticamente). Hay maneras de comprobar qué servicios han abierto algún puerto y están a la escucha de los mismos, algo que también nos puede ser de utilidad para comprobar exactamente de qué servicio tendríamos que comprobar el estado; anteriormente se usaba `nestat`, pero ahora usaremos `sudo ss -lnp` (-l lista todos los puertos que están escuchando, -n muestra el puerto concreto en número en lugar del nombre del servicio y -p el proceso que ha abierto un puerto en concreto. ss := socket statistics. También nos muestra interconexiones entre los puertos, en general). Vemos que están abiertos los puertos `80` por `apache`, el `22` por `sshd` y `mysqld` el `3306` (este último, al igual que los demás, también podemos comprobar su estado con **systemctl status mysql.service)**. Sin embargo, esto no ocurre con PHP.

Para comprobar la funcionalidad de PHP, podemos usar php -a (abre consola en PHP, el intérprete de órdenes). Los comandos terminan en "`;`". Salimos con Ctrl + D (indica fin de entrada de teclado; con Ctrl + C se aborta la ejecución del programa).

Una vez establecido, comprobamos la dirección IP de `apache` con `ip address`, y si accedemos desde el navegador del anfitrión a dicha dirección (nuestro ordenador, desde donde virtualizamos Ubuntu y CentOS), podemos ver la página por defecto de apache. También podemos hacerlo con `curl http://localhost` (client url)

En CentOS vamos a hacerlo también, pero instalando herramienta a herramienta. Primero instalaremos el servidor de páginas web. Si no sabemos cómo se llama el paquete, podemos usar `yum search httpd` (busca la cadena httpd entre todos los paquetes que tenga) o también podemos buscar por Apache. Instalaremos `httpd` (`yum install httpd`). Aquí, al contrario que en Ubuntu, el proceso no se arranca automáticamente y el servicio viene deshabilitado. Usaremos `systemctl start httpd` y `systemctl enable httpd`. Podemos comprobar su funcionamiento con `systemctl status httpd` y con `ss -lnp`, vemos que `sshd` está a `LISTEN` y con `ip address` vemos que su dirección IP es `192.168.56.110`, pero no funciona porque el cortafuegos nos está limitando la conexión. Vamos a solucionar esto con `firewall-cmd --add-service=http`. Recordemos que cuando se reinicie esto se va a perder, así que también hay que hacer `firewall-cmd --add-service=http --permanent` y, efectivamente, ahora sí funciona y carga la web de prueba. Para la base de datos, instalaremos **MariaDB**, una versión libre de MySQL, con `yum install mariadb-server` (y no `mariadb`, pues lo que necesitamos es un servidor). Una vez instalada, podemos usar de nuevo `ss -lnp` y vemos que no está escuchando por no haber iniciado y habilitado el servicio tras la instalación (puerto `3306` de `mysqld`).

Podemos conectarnos con `mysql -u root -p`, y dejamos la contraseña vacía. Y listo, ya estamos dentro. Pero esto es terrible, pues no hemos configurado nada y MariaDB nos está permitiendo el acceso. Justo después de su instalación, hay que cambiar la configuración de MariaDB (quitar el acceso a root, cambiar la contraseña, etc., porque aunque el cortafuegos de CentOS no deje pasar a nadie, esto sería muy peligroso). `mysql-secureinstallation` nos pone las cosas fáciles con la configuración de la base de datos. Configuración: no ponemos contraseña inicial, aunque sí lo hacemos para el root (de la base de datos, no del sistema; nada tiene que ver con lo que hablamos sobre el acceso remoto al root en la sesión 1 de esta misma práctica). Aceptamos que elimine los usuarios anónimos, el acceso a root, las tablas de prueba y que se recarguen las tablas (damos intro a todo salvo para introducir la contraseña). Entramos `systemctl restart mariadb` para hacer efectivos los cambios.

Ya hemos instalado el servidor apache, la base de datos MariaDB y tan solo nos falta instalar PHP con `yum install php`. Para comprobar que funciona, crearemos una pequeña página en php (podemos copiarla de internet, como la que aparece en https://www.php.net/manual/es/function.mysql-connect.php). Necesitamos un ejemplo de código en PHP, pero que también se conecte a la base de datos, como este:

```php
<?php
$enlace =  mysql_connect('localhost', 'usuario_mysql', 'contraseña_mysql');
if (!$enlace) {
    die('No pudo conectarse: ' . mysql_error());
}
echo 'Conectado satisfactoriamente';
mysql_close($enlace);
?>
```

Vamos a copiar ese código en un archivo de texto en el anfitrión, y ahora lo pasaremos a la máquina virtual con `netcat`, que también lo podemos usar para transferir ficheros a un socket. Primero abriremos un puerto con el cortafuegos `firewall-cmd --add-port=8080/tcp` (no hace falta poner `--permanent` porque es solo para esta vez). Nos movemos al directorio `/var/www/html/`, instalamos `netcat` con `yum install nmap-ncat` y luego ejecutamos `nc -l -p 8080 > myscript.php` para que comience la escucha desde ese puerto. Desde el anfitrión, ejecutamos `nc 192.168.56.110 8080 < myscript.php` y finalizamos la transmisión, una vez tenemos el archivo en el servidor. Hay que tener en cuenta que queremos que se interprete el código, no que se muestre, como pasa cuando nos conectamos con `192.168.56.110/myscript.php` (desde el anfitrión). Lo solucionamos reiniciando el servicio `httpd`. Si queremos ver los fallos de `httpd`, podemos hacerlo en `/var/log/httpd/error_log`. Nos dice que no reconoce la función `mysql_connect`; y es que no se conecta realmente con el intérprete, sino que hay que instalar unas librerías de funciones que nos permite acceder a la funcionalidad de la base de datos remota, y es en ellas donde están las funcionalidades que ejecuta el código. En concreto, el nombre que éstas reciben es `php-mysql`, y lo instalamos con `yum install php-mysql`. También hay que cambiar el nombre de usuario y contraseña que viene en el archivo de código (con `vi`, por ejemplo). Ponemos de usuario `root`, y la contraseña de siempre (aquí sí podemos loguearnos como root porque estamos en el localhost y no estamos tratando de acceder remotamente). Guardamos, volvemos a reiniciar el servicio `httpd` y vemos que ya sí que muestra, tal y como el código especifica, el mensaje “Conectado satisfactoriamente”.

Si quisiéramos que esta fuera la página de inicio de nuestro servidor, en el archivo de configuración en `/etc/httpd/httpd/conf/httpd.conf`. Cambiaremos el parámetro abajo en el documento que se llama `DirectoryIndex`, que indica al servidor qué fichero usará por defecto cuando en la URL que queremos mostrar indiquemos una carpeta. Podemos especificar que el fichero por defecto sea `myscript.php` en lugar de `index.html`. Lo modificamos y reiniciamos el servicio para que los cambios surtan efecto.