Practica-2-Leccion-3-Resuelta.pdf

Zukii

Ingeniería de Servidores

3º Doble Grado en Ingeniería Informática y Administración y Dirección de Empresas

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

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Práctica 2 – Lección 3 - Resuelta

Vídeo 1: SSH parte 2

/-----Aspectos Avanzados de SSH-----\

En la P2-L1 configuramos el servicio sshd en el puerto 22022 y deshabilitamos el acceso mediante el acceso root.

Hoy: 1) permitiremos acceso sin contraseña (por rapidez, seguridad de que no nos vean tecleando la contraseña). Esto lo haremos con una llave privada en el cliente, y una llave pública en el servicio.

Para generar una llave: **ssh-keygen**
Nos dirá si queremos guardar la llave en la carpeta por defecto, (**enter** para aceptarlo)
Nos dice que pongamos una contraseña a la llave (no la añadimos en este caso, aunque si tuviésemos la llave en un USB, no sería buena idea) (**enter**).

ls -la .ssh/ y vemos como hay dos llaves, la pública (.pub que es la que distribuimos en las máquinas que nos queramos autenticar) y la privada.

La idea es cifrar un paquete con la llave privada, y la otra máquina, será capaz de descifrarlo ya que tiene la llave pública.

Copiaremos la llave pública (de CentOS a Ubuntu) con **ssh-copy-id (usuario@)192.168.56.105 -p 22022**
yes
Ahora nos pedirá el password de nuestro usuario: **ISE**
Ahora, si hacemos quit e intentamos hacer **ssh (usuario@)192.168.56.105 -p 22022** ya no nos pedirá la contraseña.

Luego, deshabilitaremos el acceso con contraseña:
sudo nano /etc/ssh/sshd_config
(escribimos encima de #permitemptypasswords): **PasswordAuthentication no**
systemctl restart sshd (reiniciamos para que se aplique el cambio)

Y comprobamos intentando entrar desde CentOS (que ya tiene acceso) y por ejemplo desde la WSL (no tiene forma de autenticarse).

Para permitir que la WSL subiese su llave pública al servicio ubuntu para poder acceder sin contraseña, deberíamos modificar el archivo que modificamos antes, ponerlo a **yes systemctl restart sshd**, hacer **ssh-copy-id (usuario@)192.168.56.105 -p 22022** y volvemos a desactivar la opción, poniéndolo a no y **systemctl restart sshd**

2) Otra medida de seguridad que haremos hoy, es dejar a unos únicos usuarios que puedan acceder (loggearse)

Creamos usuario en CentOS:
sudo adduser nico

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

sudo passwd nico

Luego, intentaremos acceder al servicio con **ssh nico@192.168.56.105 -p 22022** pero no nos deja, primeramente porque se deshabilitó el acceso por contraseña y luego porque la máquina que tiene el servicio, no tiene el usuario nico.

Creamos el usuario nico en ubuntu con **sudo adduser nico**, **ISE, enter** en el resto de valores.

Volvemos a permitir el acceso con contraseña y hacemos **systemctl restart sshd** y ya podemos entrar como nico a ubuntu.

Lo que queremos es que solo se pueda acceder con nuestro usuario (alvaro), para esto:
sudo nano /etc/ssh/sshd_config

Y escribimos: **AllowUsers alvaro**
systemctl restart sshd

Volvemos a intentar entrar como nico y vemos como no nos deja pero con alvaro como usuario si nos deja.

Ya tenemos configurado los aspectos mínimos de seguridad.

/-----Utilidades Extra-----\

a) Ahora usaremos la utilidad **sshfs**. Montaremos en nuestro equipo, de forma transparente, una ubicación remota. Además, el acceso a esa información será mediante ssh garantizando la seguridad.

(NOTA: él está desde fedora, quizás en la WSL es distinto)
Desde la WSL hacemos **dnf provides sshfs** y vemos que el paquete que podemos utilizar para instalarlo:

sudo dnf install fuse-sshfs-3.7.1-2.fc34.x86_64
($ sudo apt install sshfs creo que es en ubuntu)

En este caso, crearemos una carpeta (en la WSL) **mkdir ./ubuntuserver**
sshfs alvaro@192.168.56.105:/home/alvaro ./ubuntuserver/ -p 22022

Ahora tenemos el contenido de ubuntu server pero en la máquina de WSL (podemos hacer mount para verlo). Si por ejemplo creamos un archivo con **touch Hola**, a los segundos, nos debería salir en el directorio que tenemos en la WSL.

b) Otra funcionalidad que implementaremos será la de **XForwarding**, que la usaremos para:
En nuestro server tendremos una aplicación que haga uso de la interfaz de ventana. Pues XServer, en vez de que las peticiones las haga al servidor local, se reenvían al servidor de nuestro cliente, y esta interfaz le aparece al cliente. Si bien, la ejecución se está haciendo en el servidor.

Desde la WSL me loggeo en el ubuntu server y añadimos la opción -X, haciendo el xforward (**ssh (usuario@)192.168.56.105 -p 22022 -X)**. Entonces con gedit o un

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

navegador, tiene sentido la opción -X (que además es segura). Así, si instalamos el gedit (que no está, con **sudo apt install gedit**) y hacemos gedit, vemos como nos sale en el cliente la interfaz.

Destacamos que la interfaz la ejecuta el cliente, pero el procesamiento interno lo hace el servidor. Al guardar, vemos que guarda en el servidor como bien podríamos pensar.

Por último, dos utilidades:
c) screen y tmax -> nos permiten lanzar un trabajo en una terminal y dejar esa terminal sin ninguna sesión vinculada, y volverla a retomar cuando queramos. Si hacemos una ejecución que requiere mucho tiempo, nos desconectamos y conectamos cuando queramos y retomamos el control. Otro escenario es ejecutar un procedimiento en el que se nos va la luz y perdemos la conexión y esto nos obligaría a reiniciar el procedimiento.

Vamos a la WSL y nos conectamos a ubuntu server: **ssh (usuario@)192.168.56.105 -p 22022)**.
Creamos un archivo **nano miarchivo** y si vamos a la máquina con ubuntu, hacemos **ps**, podemos ver que sshd tiene su bash y su aplicación nano editando el archivo. Si cerramos la pestaña de la WSL y volvemos a hacer **ps**, vemos como los procesos anteriores se han terminado.

Para evitar que se terminen, usaremos screen. Nos logueamos en ubuntu y ponemos la orden **screen** y ya hacemos el **nano**. Hacemos **ps ax** en el servidor y vemos que bash ha invocado el comando screen, que ha invocado otro proceso SCREEN, del cual cuelga otro bash y el nano. Si cerramos la pestaña de la WSL y hacemos **ps ax**, vemos que sigue estando SCREEN, donde cuelga el bash y el nano.

Para recuperar el archivo (lo podemos hacer como clientes o en el servidor), nos logueamos de nuevo en ubuntu desde la WSL, hacemos **screen -r -d** recuperamos la última ventana con los cambios que hicimos.

Es importante decir que podemos tener varios screen (CTRL+A + CTRL+D para cerrar). Podemos hacer un listado para verlos con **screen -list** y nos conectamos con **screen -r (números.pts-0.ise_ubuntu) -d**

Nota: solo puede tener una ventana de un archivo a la vez. Si no, hace detached y nos echa de la otra.

tmux es casi igual..
Hacemos **tmux** en la terminal
CTRL+B Crea dos sesiones
CTRL+B + CTRL+D = detached

Vídeo 2: Fail2ban

Servicio con el que interactuaremos con systemd y con un comando llamado fail2ban.client

Lo que hará fail2ban, será hacer un sondeo de los archivos que están en /var/log y analiza el contenido de esos archivos para ver las autenticaciones erróneas que se han producido.

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

En caso de que haya una autenticación erronea, baneará al usuario que ha fallado durante un tiempo determinado (configurado en el archivo de configuración que se encuentra en /etc/)

Este servicio es importante sobretodo para impedir ataques de fuerza bruta.

Procederemos a la instalación de este servicio en CentOS:

dnf search fail2ban -> fail2ban no tiene paquete propio, se encuentra en el conjunto de paquetes extendido

dnf search epel
sudo dnf install epel-release
ISE
s

Ahora si podemos buscar fail2ban
dnf search fail2ban
sudo dnf install fail2ban
s
s

Ya tendríamos instalado fail2ban. Ahora veremos el estado del servicio:
systemctl status fail2ban
sudo systemctl enable fail2ban -> se active en el próximo reinicio
sudo systemctl start fail2ban -> que se active
systemctl status fail2ban -> ya estará el servicio en ejecución

Procederemos a configurarlo. Tenemos el comando fail2ban-client. Se analizarán los distintos logs, se buscarán las direcciones IP, las cuales pasarán a una serie de jails que definiremos para cada servicio.

Las definimos (las cárceles) en el archivo de configuración:
cd /etc/fail2ban
sudo cp -a jail.conf jail.local (no editamos el archivo porque si actualizasemos, se borraría)
sudo nano /etc/fail2ban/jail.local

Buscamos [sshd] (el que no esta comentado) y escribimos en otra línea: enabled = true
sudo fail2ban-client status sshd -> vemos como aun no está activada la cárcel
sudo systemctl restart fail2ban.service
sudo fail2ban-client status sshd -> ya está operativa

Podemos comprobar que funciona, entrando desde otra terminal como la WSL:
ssh 192.168.56.110 -p 22022 (en WSL) (introduzco una contraseña incorrecta 3 veces).
sudo fail2ban-client status sshd (CentOS) -> ya aparece 3 intentos fallidos de acceso. Nos banea, por defecto, a los 5 intentos fallidos. Podemos editarlo en el archivo de configuración.
Volvemos a hacer **ssh 192.168.56.110 -p 22022 (WSL)** (y fallamos 3 veces más)
sudo fail2ban-client status sshd (CentOS) y nos aparece que ha baneado a la IP.

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

ssh 192.168.56.110 -p 22022 (WSL) -> Pero si ponemos bien la contraseña, nos podemos logear. El problema ha sido que no hemos modificado el puerto. **sudo nano /etc/fail2ban/jail.local** y volvemos a **buscar [sshd] y en port, escribimos 22022**
sudo systemctl restart fail2ban.service

ssh 192.168.56.110 -p 22022 (WSL) -> y ahora no nos debe ni dejar poner la contraseña.
sudo fail2ban-client status sshd (CentOS) -> sigue baneada la WSL

Ahora, debemos ver como sacar una IP de baneadas:
sudo fail2ban-client set sshd unbanip 192.168.56.100
sudo fail2ban-client status sshd -> no está la IP en la lista.
ssh 192.168.56.110 -p 22022 (WSL) -> nos volvería a dejar

Ya luego, podemos modificar en el archivo de configuración .local ciertos parámetros como:

Tiempo de baneo de una IP (bantime)
Número de intentos antes de banear una IP (maxretry)

Lo bueno de fail2ban, no solo puede usarse con sshd, si no, con más servicios que tengan archivos por el log.

Vídeo 3: Instalación de LAMP

Lo haremos en **CentOS**.

Pila porque es: datos (BD) , sobre esto la lógica que interactúa con los datos (el LP) y una interfaz que es el servidor web (Apache suele ser).

-- Nota histórica en el vídeo hasta 9:35

Instalaremos en CentOS la pila LAMP: (en ubuntu es de otra forma)
(sudo ifup enp0s3)
dnf search apache
sudo dnf install httpd

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

ISE
s
s

Comprobamos con **systemctl status httpd** su estado
sudo systemctl enable httpd (que este activo en el próximo reinicio)
sudo systemctl start httpd (iniciar ahora)
systemctl status

Probamos que funciona con el comando curl (para ver una url)
curl localhost, vemos que nos devuelve una página web

dnf search php
sudo dnf install php
s
(Esto es un intérprete, no un servicio, debemos provar de otra forma)
php -a

Ahora instalamos la BD
dnf search mariadb
sudo dnf install mariadb -> instala cliente
sudo dnf install mariadb-server.x86_64 -> el servicio

systemctl status mariadb -> está desactivado tras instalarlo. Muestra estado de servicio
sudo systemctl enable mariadb
sudo systemctl start mariadb

mysql (= MariaDB)
Problema: no podemos conectarnos al servidor con nuestro usuario
mysql -u root (si nos deja como root y sin contraseña)

Tras la instalación, el manual recomienda ejecutar **mysql_secure_installation** (que es un script) para ganar seguridad.

enter
Y
ISE (contraseña del root)
ISE
Y (borrar usuario anónimo)
Y (quitar root de mysql de forma remota)
Y (borrar tabla de prueba que viene)
Y (recargar tabla de privilegios)

Con esto tenemos configurado la base de datos de MariaDB. Comprobamos:

mysql
mysql -u root
mysql -u root -p [contraseña] (si no ponemos [] nos la pide ahora)

Probamos a conectarnos con apache. Abrimos la WSL y hacemos **curl 192.168.56.110** (no nos deja) pero **ping 192.168.56.110** si funciona.

El problema es causado por el firewall:

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

sudo firewall-cmd --add-port=80/tcp (ahora)
sudo firewall-cmd --add-port=80/tcp --permanent(para las siguientes ocasiones)
sudo firewall-cmd --reload

curl 192.168.56.110 (desde WSL) y ya si nos devuelve la conexión.

Nos faltaría probar que podemos integrar un script de php que se conecta a una BD y que nos muestra el resultado a través de html.

https://www.php.net/manual/es/function.mysqli-connect.php
Copiaremos el primer script que aparece en la web anterior para probar si hay conexión.
Nos creamos otra terminal en la WSL y nos conectamos: **ssh alvaro@192.168.56.110 -p 22022**

Tenemos que ubicar nuestro archivo, en un lugar que sea accesible con http:
less /etc/httpd/conf/httpd.conf

Buscamos el parámetro DocumentRoot, y nos pondrá el directorio donde tenemos que poner el script. (/var/www/html)

cd /var/www/html
sudo nano index.php

Insertamos el código de la web. Aunque probablemente tengamos que cambiar los parámetros de la función mysqli_connect. (segundo y tercer parámetro por root, ISE)

mysql -u root -p
ISE
CREATE DATABASE mi_bd; (ya tenemos la BD que aparece en el parámetro)

Vamos a navegador y ponemos la ip de ubuntu:
192.168.56.110 -> va bien
192.168.56.110/index.php -> lo sirve como documento de texto, con el riesgo que conlleva.

Problema: apache no está interpretando el php. Modificaremos la configuración de php:
en CentOS:

sudo nano /etc/httpd/conf/httpd.conf
ISE

Buscamos DirectoryIndex y añadimos después de index.html ***.php** (para que sirva cualquier archivo con ext php)
Hacemos **f5 en el navegador** y vemos que no ha cambiado nada. Reiniciamos el ervicio

sudo systemctl restart httpd
f5 y vemos que hay error http error 500 (error del servidor).

Vamos a CentOS y ejecutamos el archivo:
php index.php (da error ya que no hemos instalado la biblioteca que conecta php y sql)

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

sudo dnf search php | grep mysql
sudo dnf install php-mysqlnd
s

Probamos de nuevo:
php (/var/www/html)index.php (y ya al menos funciona en el servidor)
f5 y vemos que hay error de permisos

Por ver xd: **sudo less /var/log/audit.log y buscamos un httpd por el final.**

SE es que no da el acceso.
getsebool -a | grep httpd
sudo setsebool httpd_can_network_connect_db=on

getsebool -a | grep httpd (vemos que está a on)
Hacemos **f5** y vemos que tenemos la pila LAMP configurada en nuestro servidor.

NOTAS POST VIDEO + APUNTES

Para examen traer ubuntu y centos con comunicación ya probada.

Integrar código php en html:
<?php
...
?>

LAMP es un stack de desarrollo. Hay otras como XAMP o WAMP.
LAMP: Linux, Apache, MySQL o MariaDB, Php (o Python)
= SO, Servidor, Base de Datos y Lenguaje de Programación

Nginx -> óptimo con E/S asíncronas. Se suele usar para acceso a archivos estáticos tipo estáticos como fotos, texto…
Apache -> Óptimo para contenido dinámico como php o django.

Tendremos que instalar LAMP en ubuntu (que es instalar un paquete) y LAMP en CentOS (paquete a paquete: Apache, MySQL y Php).

Ubuntu según **internet**:
sudo apt update; sudo apt upgrade
sudo apt install -y apache2 apache2-utils
sudo ufw allow http (abrir puerto en el firewall)
sudo apt install mariadb-server mariadb-client
systemctl status mariadb
sudo systemctl start mariadb
sudo systemctl enable mariadb
sudo mysql_secure_installation (hacer como en CentOS)

sudo apt install php8.0 libapache2-mod-php8.0
sudo systemctl restart apache2

WUOLAH

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Ubuntu (lo que hicimos en clase, se parece bastante a lo de internet de arriba)

> sudo apt-get install apache2
(Si no fufa hacer un sudo apt-get update)

> sudo systemctl start apache2
> sudo systemctl status apache2 (comprobar si todo ha ido bien)

- El firewall está tapado si al poner la dir IP de Ubuntu en el navegador no sale nada
> sudo ufw allow 80
- Y debería de ir a poner 192.168.56.105 en el nav.

En esa pagina no se necesita ningún lenguaje porque es una página estática en la que sus datos no cambian. Es como si fuera un blog

- Instalar tanto mysql como php y comprobar si fufa
> sudo apt-get install mysql-server mysql-client
> sudo apt-get install php
> sudo systemctl start mysql
> sudo systemctl status mysql
(php no hace falta)

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.