Universidad Castilla La Mancha

Ingeniería informática

Seguridad en Redes

Laboratorio 4

Laura Toledo Gutiérrez

28 de Noviembre, 2023

Índice

1. Introdución

En esta práctica se nos plantea el reto de implementar un sistema distribuido basandonos en la práctica 3 realizada anteriormente que haga uso de los mecanismos de seguridad que hemos visto en teoria. Cuenta con dos componentes principales: Servicio de base de datos y accesso SSH al personal.

- **Servicio de base de datos**: La aplicacion se divide en 3 partes, auth, files y broker.
- **Acceso SSH al personal**: Hay dos usuarios, dev y op.

2. Requisitos

2.1. Docker

Como se ha comentado, la idea es implementar un sistema distribuido haciendo uso de contenedores Docker. Docker se utlizará para lanzar el entorno, para ello tenemos que tenerlo intalado en nuestro sistema:
La instalación es sencilla:
`sudo apt update`
`sudo apt-get install docker.io`
Una vez instalado con exito, si hacemos uso del siguiente comando nos ahorraremos escribir *sudo* cada vez que usemos el comando *docker*, añadiremos nuestro nombre de usuario al grupo docker:
`sudo usermod -aG docker $USER`

2.2. Lenguajes de desarrollo

Los lenguajes de programación utilizados son **GO!** para el desarrollo de la aplicación y **Python** para el entorno de pruebas.
Para ello, como es obvio, hay que realizar la previa instalación de estas herramientas:

2.3. DNS

Debemos configurar el archivo */etc/hosts* añadiendo la direccion con la que vamos a trabajar:
`172.17.0.2 myserver.local`

3. Certificados

Uno de los requisitos es hacer la comunicacion por HTTPS para ello es necesario la creación de los certificados de cada componente. Para ello se ha hecho uso de un archivo .cnf que se usara para la creación de cada certificado donde esta configurado el IP_Host de cada componente: Ubicados en la carpeta de cada componente. Una vez configurado el archivo, podemos crearlos usando el siguiente comando:

```
openssl req -new -newkey rsa:2048 -nodes -keyout server.key -out server.csr -config
san.cnf

openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt -extensions
v3_req -extfile san.cnf
```

Estos comandos generarán tanto la clave privada como un certificado autofirmado que incluye la direccion IP como un SAN. Se crearán de cada uno de los componentes.

Una vez creados los debemos copiar en la siguiente ruta *usr/local/share/ca-certificates* y despues se deberá actualizar la lista de certificados
`sudo update-ca-certificates --fresh`.

4. Funcionamiento

La implementación de los componentes se ha realizado en GO!, auth, broker y files.

- **Broker**: Se encarga de recibir las peticiones del usuario a traves del router, estas peticiones las redireccionará o al nodo auth o files dependiendo de la petición.
- **Auth**: Se encarga de las peticiones de registro de usuarios
- **Files**: Se encarga del espacio del usuario.

El router es el encargado de mandar los paquetes a los servidores correspondientes, utiliza el puerto 5000 para las peticiones.

4.1. IpTables

Se han implementado **iptables** para la configuración de la red y la seguridad de los servidores usados. Se han implementado las reglas de filtrado necesarias para el funcionamiento correcto del sistema. Hace la funcion de un cortafuegos, este esta implementado en cada uno de los componentes de nuestro sistema. La politica por defecto de las chain para la tabla filter es:
- INPUT -¿DROP
- FORWARD -¿DROP
- OUTPUT -¿ACCEPT

4.2. SSH

Para poder realizar SSH tenemos que tener en cuenta que para acceder por ssh a *work* necesitamos pasar primero por *jump*, pero antes de jump hay que pasar por el *router*.

Se ha realizado esta configuración el archivo *entrypoint.sh* del nodo jump:

```
echo -e "Match Address 10.0.1.2 AllowUsers jump»>/etc/ssh/sshd_config
echo -e "Match Address 10.0.3.3 AllowUsers op»>/etc/ssh/sshd_config
echo -e "Match Address 10.0.3.3 AllowUsers dev»>/etc/ssh/sshd_config
```

Con esto conseguimos establecer las reglas que permiten la conexión de los usuarios.

Ahora hay que realizar una serie de pasos para configurar el uso del ssh a través del nodo intermedio:

- **Agregar claves privadas al SSH Agente**
  - Iniciamos el ssh-agent:
    `eval 'ssh-agent -s'`
  - `ssh-add docker/assets/ssh/op_key`
    ssh-add docker/work/dev_key

- **Conexion SSH a work a traves de jump**
  ssh -J jump@172.17.0.2 -A op@10.0.3.3
- **Conexion a otros nodos desde work**
  ssh op@10.0.2.4

4.3. Rsyslog

Para la centralización de los logs generados se ha configurado el nodo logs para usarlo como servidor central. Se ha utilizado la herramienta **rsyslog** y el modulo *imupd*. Se han añadido los archivos de configuraciones para indicar las rutas donde se almacean los logs y se ha añadido las correspondientes reglas iptables.

5. Ejecución

Nos situamos en el directorio raiz del sistema, una vez ahi podemos lanzar el comando
make
para lanzar el sistema, al hacer make all se construiran y lanzaran los contenedores.

5.1. Ejecución tests

Para facilitar la comprobación de la robustez e integridad del sistema, se ha creado un script en python que lanza pruebas automaticas. Para ejecutar el script podemos hacer uso del Makefile con el comando:
make run-tests