ESCUELA SUSPERIOR DE INFORMÁTICA

SEGURIDAD EN REDES

PRÁCTICA 4

API RESTful DISTRIBUIDA

Juan Bautista Castaño García-Cervigón

Alberto Mayorga Naranjo

Arturo Gómez Jiménez

Rodrigo Gómez Sánchez

Cristian Ballesteros Serrano

Índice

I. Explicación de la práctica.

Los objetivos de la práctica son:

- Entender e implementar un sistema distribuido no-trivial con diferentes usuarios y niveles de acceso.
- Entender e implementar mecanismos de protección de intrusos.
- Entender e implementar mecanismos de detección de intrusos.
- Entender e implementar mecanismos de logging y auditoría.

Vamos a implementar un sistema distribuido basándonos en la API RESTful de la práctica 3. Este sistema distribuido consta de dos partes: servicio de base de datos y acceso SSH del personal.

a. API distribuida.

Ahora debemos reestructurar nuestro anterior servicio de base de datos de forma que pueda escalar para ser un servicio en la nube. La aplicación se divide en 3 partes:

- auth: servicio de autentificación que implementa todo lo relacionado con los usuarios y sus tokens.
- file: servicio de almacenamiento que implementa todo lo relacionado con la gestión de archivos.
- broker: servicio que recibe las distintas peticiones de los usuarios y redirige las llamadas a auth o a file dependiendo de cuál sea la petición.
  - */version* la atiende el propio broker.
  - */login* y */signup* las redirigirá a auth.
  - El resto las redirige a file.

b. Acceso SSH al personal.

Tenemos que gestionar el acceso SSH del personal en todos los nodos. Existen dos tipos de usuario:

- dev: usuario de desarrollo que sólo tiene acceso a un nodo de trabajo que llamaremos *work* a la cual accede a través del nodo *jump*. No tiene acceso a *sudo*. Este usuario no tiene acceso a sudo.
- op: usuario operador que tiene acceso a todos los nodos del sistema a las que debe acceder a través del nodo *work*, que previamente se debe acceder a través de *jump*. Este usuario puede ejecutar sudo.

En cuanto a la seguridad, a parte de los mecanismos ya implementados en la práctica 3 respecto a las longitudes y caracteres de las contraseñas y posibles ataques de fuerza bruta, ahora hay que implementar una política de seguridad en la red que tenemos.
Tenemos tres redes diferentes:

- dmz: red que contiene los servicios que tienen que estar conectados y ser accesibles desde el exterior (*jump* y *broker*).
- srv: red que contiene los servicios de *auth* y *file*.
- dev: red donde se pondrán los servicios para el personal (*work*).

En cada nodo debe estar implementado un cortafuegos utilizando **iptables**. Esta implementación se encuentra en los archivos con nombre *entrypoint*. La política de las *chain* que vamos a usar para la tabla *filter* es:

- INPUT -> DROP
- FORWARD -> DROP
- OUTPUT -> ACCEPT

II. Contenidos práctica.

La práctica está formada por archivos *Dockerfile*, *Entrypoint* y los archivos .py con la API de la anterior práctica.

Los *Dockerfile* contienen las distintas configuraciones que nuestro contenedor debe tener instaladas, además de los *COPY* que son los archivos que queremos que tenga el contenedor.

Los Entrypoint contienen todas las reglas *iptables* relacionadas con la política de seguridad en la red y SSH de los usuarios *dev* y *op*.

La API está dividida en varios .py, los cuales están repartidos en sus correspondientes nodos, junto con los códigos de error HTTP. La funcionalidad es exactamente igual que la de la práctica 3, lo único que nos hemos visto obligados a cambiar ha sido en el archivo '*apiAuth.py'* que he tenido que crear un método '*verify'* para comprobar las cabeceras y hacerle llamada el nodo files.

Como comentario, a pesar de ser conscientes de la importancia de incorporar los certificados SSL, hemos tomado la decisión de no incluirlos en el desarrollo del trabajo por los problemas que se presentaban en la tarea cuando se incorporaban (la no conexión entre el broker y el auth al hacer algunas request). De todas formas, vamos a incorporar la forma en la cual estábamos creando dichos certificados.

III. Ejecución.

Antes de todo debemos tener instalado *Docker*.

Para la ejecución, debemos situarnos en el directorio raíz de la práctica y hacer '*make* all'. Esta instrucción ejecuta *DNS* para la traducción de nombre, *certificates* para crear los certificados SSL, *build* para crear los contenedores, *network* para crear las redes y *containers* para ejecutar los contenedores.

Para construir las key de los usuarios hay que ejecutar el comando *'make key'*.

Por último, ejecutamos ./test.py para realizar todas las pruebas. A parte es posible realizar peticiones por nuestra cuenta sin necesidad del test siempre y cuando sepamos la sintaxis correcta de las peticiones.