Laboratorio de administración y gestión de redes y sistemas
Examen de prácticas. 14 de diciembre de 2018.
Grado en Ingeniería Telemática, Universidad Rey Juan Carlos

Instrucciones

- Abre una sesión en el laboratorio con tu usuario y contraseña habituales.
- Ejecuta el script `prepara_lagrs`
  Esto creará en tu cuenta el directorio `~/lagrs.diciembre.18`
  y los ficheros `~/lagrs.diciembre.18/practico.tulogin.txt`,
  `~/lagrs.diciembre.18/claves.tgz` y
  `~/lagrs.diciembre.18/vigila_puertos.py`

Ejercicio 1. (2 puntos)

En el fichero `~/lagrs.diciembre.18/claves.tgz`
tienes todo lo necesario para abrir una sesión ssh en la máquina 193.147.79.2 con el usuario *alumno*. También ficheros que *sobran*. Entra en esa máquina, escribe en `practico.tulogin.txt` los pasos que has seguido y muestra la sesión al profesor.

Ejercicio 2. (4 puntos)

En este ejercicio escribirás un script en python que, usando la order netstat, vigilará ciertos puertos y enviará un mensaje de telegram si algún proceso escucha peticiones en ellos. Edita el fichero `~/lagrs.diciembre.18/vigila_puertos.py` para que se corresponda con la siguiente especificación:

1. El programa leerá de un fichero de texto con nombre `id_usuario.txt` el identificador de usuario de telegram que recibirá las alarmas. No especifiques ningun *path*, de forma que se busque en el directorio actual.
2. El programa leerá el token del bot de telegram en el fichero `token.txt`. De nuevo, no especifiques ningún trayecto, que el script lea el fichero desde el directorio actual.
3. En las primeras líneas del script define una lista parecida a esta:

   ```
   PUERTOS_TCP = [6666, 443]
   ```

4. Cada vez que se ejecute el programa, revisará todos los puertos TCP especificados en PUERTOS_TCP, y si alguno de ellos está ocupado, en cualquier estado (ya sea *ESCUCHAR*, *ESTABLECIDO*, o cualquier otro en cualquier otro idioma: *LISTEN*, *ESTABLISHED*...), enviará una alarma a través de telegram describiendo toda la información relevante proporcionada por netstat.
5. La alarma solo se disparará si el puerto está ocupado en alguna dirección local, sea cual sea, pero no en ninguna dirección remota.
6. Usa romanserver.py para probar el programa. Cuando funcione, enséñaselo al profesor.

Ejercicio 3. (4 puntos)

En este ejercicio prepararás y lanzarás dos contenedores Docker, uno con *romanserver.py* y otro con *romanclient.py*, según la siguiente especificación:

1. Ambos contenedores estarán conectados por un segmento privado de red dentro de Docker.
2. Como todos los contenedores del laboratorio comparten segmento de red virtual, cada alumno necesita su propio puerto. El profesor te indicará un *puerto_alumno*, usa el puerto TCP 12000 + *puerto_alumno*.
3. El contenedor con romanserver.py estará basado en una imagen llamada `exa`. Si tu nombre de usuario fuera *jperez*, el contenedor se llamaría `jperexa01`. Sustituye *jper* por los primeros 4 caracteres de tu nombre de usuario.

   Al ejecutarlo, lanzará el programa `romanserver.py` en el puerto reservado para tí y a continuación abrirá una shell.
4. El contenedor con romanclient.py estará basado en una imagen llamada `exb`.

   Al ejecutarlo, lanzará un terminal, donde el usuario podrá invocar a romanclient.py con los parámetros adecuados. El usuario deberá podar lanzar romanclient.py sin especificar ningún path.
5. El contenedor exa también también tendrá disponible romanclient.py, para poder probar el servicio en local.
6. Ambos contenedores deberán poderse hacer *ping* entre sí. También podrán usar *ifconfig*.
7. Usa el mismo convenio empleado en clase para nombrar los ficheros de configuración de los contenedores, creación de imágenes y ejecución. Aunque guardando todo en el directorio `~/lagrs.diciembre.18`. Esto es, usa los siguientes ficheros:

   ```
   ~/lagrs.diciembre.18/exa/construye.sh
   ~/lagrs.diciembre.18/exa/lanza_jperexa01.sh
   ~/lagrs.diciembre.18/exa/context/Dockerfile
   ~/lagrs.diciembre.18/exa/context/entrypoint.sh
   ~/lagrs.diciembre.18/exa/context/romanserver.py
   ~/lagrs.diciembre.18/exa/context/romanclient.py
   ```

   Y para el cliente:

   ```
   ~/lagrs.diciembre.18/exb/construye.sh
   ~/lagrs.diciembre.18/exb/lanza_jperexb01.sh
   ~/lagrs.diciembre.18/exb/context/Dockerfile
   ~/lagrs.diciembre.18/exb/context/entrypoint.sh
   ~/lagrs.diciembre.18/exb/context/romanclient.py
   ```
8. Dentro de los contenedores, los ficheros romanserver.py y romanclient.py deben estar en el sitio *correcto* según la norma FHS.
9. Instala solo los paquetes necesarios, usa solo los ficheros necesarios (y no otros que hayan hecho falta en las prácticas).
10. Cuando lo tengas funcionando, enséñaselo al profesor.