UNIVERSIDADE DA CORUÑA

Armando Martínez Noya (a.mnoya)
3º Ingeniería Informática

PRÁCTICA 1 LSI:

a) Configure su máquina virtual de laboratorio con los datos proporcionados por el profesor. Analice los ficheros básicos de configuración (interfaces, hosts, resolv.conf, nsswitch.conf, sources.list, etc).

Para empezar, desactivamos los servicios relacionados con el avahi-daemon, estos servicios se encargan de detectar automáticamente los recursos de una red local y conectarse a ella.

```
lsi@debian:~$ systemctl stop avahi-daemon.socket
lsi@debian:~$ systemctl stop avahi-daemon.service
lsi@debian:~$ systemctl stop NetworkManager.service
```

```
lsi@debian:~$ systemctl disable avahi-daemon.socket
lsi@debian:~$ systemctl disable avahi-daemon.service
lsi@debian:~$ systemctl disable NetworkManager.service
```

Además de esto configuramos las interfaces de red de nuestra maquina editando el fichero /etc/network/interfaces escribiendo lo siguiente:

```
auto lo ens33 ens34
iface lo inet loopback

iface ens33 inet static
address 10.11.49.48
netmask 255.255.254.0
broadcast 10.11.49.255
network 10.11.48.0
gateway 10.11.48.1

iface ens34 inet static
address 10.11.51.48
netmask 255.255.254.0
broadcast 10.11.51.255
network 10.11.50.0
```

Para hacer efectivos los cambios ejecutamos el comando:

```
lsi@debian:~$ systemctl restart networking.service
```

Ficheros básicos de configuración:

-***resolv.conf***: fichero que contiene las direcciones de los servidores DNS de nuestra máquina.

-***nsswitch.conf***: también llamado Name **Service Switch configuration file**, tiene como objetivo especificar de donde obtienen la información ciertos valores especiales, también llamados categorías, relacionados con **GNU C**

Library, y en caso de obtener la información de varios orígenes, en qué orden consultar éstos con el fin de saber qué fuente tiene más prioridad.

-***sources.list***: es un archive que contiene las direcciones web de los diferentes repositorios del sistema.

b)¿Qué distro y versión tiene la máquina inicialmente entregada? Actualice su máquina a la última versión estable disponible.

Para ver la versión que tenemos actualmente instalada podemos usar el comando:

```
lsi@debian:~$ lsb_release -a
```

Consiguiendo de esta manera la siguiente salida por pantalla:

```
lsi@debian:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux 10 (buster)
Release:        10
Codename:       buster
```

Para actualizarla a la última versión (Debian 11) actualizamos primero Debian 10, luego de tenerlo actualizado nos dirigimos a /etc/apt/source.list y cambiamos los repositorios de **buster** por los nuevos de **bullseye.**

Finalmente hacemos un apt full-upgrade y reiniciamos la máquina (reboot).

```
lsi@debian:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux 11 (bullseye)
Release:        11
Codename:       bullseye
```

c)Identifique la secuencia completa de arranque de una máquina basada en la distribución de referencia (desde la pulsación del botón de arranque hasta la pantalla de login). ¿Qué target por defecto tiene su máquina? ¿Cómo podría cambiar el target de arranque? ¿Qué targets tiene su sistema y en qué estado se encuentran? ¿Y los services? Obtenga la relación de servicios de su sistema y su estado. ¿Qué otro tipo de unidades existen?

Para ver los registros del sistema (logs)del boot actual de la maquina ejecutamos el comando:

```
lsi@debian:~$ journalctl -b
```

Secuencia completa de arranque:

1.- El hardware carga la BIOS, esta hace comprobaciones de hardware y carga el MBR (que se encuentra en el primer sector del disco).

2.- El MBR se encarga entre otras cosas de cargar y ejecutar el gestor de arranque, en nuestro caso el GRUB.

3.- El GRUB elige el SO que se iniciará y carga el Kernel. Es decir se inicializa el núcleo y monta la partición raíz.

4.- Se ejecuta el proceso init o initramfs, fichero raíz en RAM. Init configura todos los servicios y estructuras que no sean del SO. Initramfs entrega el control a systemd.

5.- Systemd inicia grupos de procesos y servicios en paralelo organizados en targets. Una vez finalice la inicialización, el SO quedará iniciado.

Target: son los diferentes estados de ejecución que definen el conjunto de servicios y características activas por defecto en el sistema, también se les conoce por "runlevel".

Para conocer el target por defecto de nuestra maquina llega con ejecutar el comando:

```
lsi@debian:~$ systemctl get-default
```

Siendo nuestro target por defecto el *graphical.target.*

Una buena idea será cambiar nuestro target por defecto al **multi-user.target** obviando así la ejecución de muchos servicios que no nos interesan, esto podemos hacerlo con el comando:

```
lsi@debian:~$ systemctl set-default multi-user.target
```

Para conocer todos los targets de nuestro sistema y los estados de estos ejecutamos el comando:

```
lsi@debian:~$ systemctl list-units --type=target
```

Consiguiendo así la siguiente lista:

```
lsi@debian:~$ systemctl list-units --type=target
  UNIT                   LOAD   ACTIVE SUB    DESCRIPTION
  basic.target           loaded active active Basic System
  cryptsetup.target      loaded active active Local Encrypted Volumes
  getty.target           loaded active active Login Prompts
  graphical.target       loaded active active Graphical Interface
  local-fs-pre.target    loaded active active Local File Systems (Pre)
  local-fs.target        loaded active active Local File Systems
  multi-user.target      loaded active active Multi-User System
  network.target         loaded active active Network
  nss-user-lookup.target loaded active active User and Group Name Lookups
  paths.target           loaded active active Paths
  remote-fs.target       loaded active active Remote File Systems
  slices.target          loaded active active Slices
  sockets.target         loaded active active Sockets
  swap.target            loaded active active Swap
  sysinit.target         loaded active active System Initialization
  time-set.target        loaded active active System Time Set
  time-sync.target       loaded active active System Time Synchronized
  timers.target          loaded active active Timers

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.
18 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

Para conocer todos los targets de nuestro sistema y los estados de estos ejecutamos el comando:

```
lsi@debian:~$ systemctl list-units --type=service
```

Dicha lista no se muestra en la práctica por ser demasiado extensa

Para conocer todos los servicios y estados del sistema basta con ejecutar el comando:

```
lsi@debian:~$ systemctl list-units - -type=service -all
```

Dicha lista no se muestra en la práctica por ser demasiado extensa

Además de los targets y los services existen otro tipo de unidades como pueden ser, por ejemplo:

-Sockets: similar a un socket de Internet que se utiliza en los sistemas operativos POSIX para comunicación entre procesos. Estas conexiones

aparecen como flujos de bytes, al igual que las conexiones de red, pero todos los datos se mantienen dentro del ordenador local.

-Timers: son servicios que ejecutan funciones con cierta periodicidad.

-Mount, Slice, device, path, scope, etc.

d)Determine los tiempos aproximados de botado de su kernel y del userspace. Obtenga la relación de los tiempos de ejecución de los services de su sistema.

Para obtener los tiempos de bootado tanto del kernel como del userspace llega con ejecutar el comando:

```
lsi@debian:~$ systemd-analyze

Startup finished in 3.090s (kernel) + 28.127s (userspace) = 31.218s
graphical.target reached after 28.073s in userspace
```

Para obtener los tiempos de ejecución de cada uno de los servicios ejecutamos:

```
lsi@debian:~$ systemd-analyze blame
```

e)Investigue si alguno de los servicios del sistema falla. Pruebe algunas de las opciones del sistema de registro journald. Obtenga toda la información journald referente al proceso de botado de la máquina. ¿Qué hace el systemd-timesyncd?

Para listar los servicios que fallan ejecutamos el comando:

```
lsi@debian:~$ systemctl --state=failed
```

Opciones del registro journald:

- journalctl: muestra todos los logs del sistema.
- journalctl -f: muestra los mensajes que se van generando en tiempo real.
- journalctl -r: revertir la salida para que las entradas más recientes se muestren primero.
- journalctl -b: muestra los logs del botado de la máquina.

El comando **systemd-timesyncd** es un daemon que sirve para sincronizar el reloj del sistema a través de la red, utilizando el protocolo SNTP.

f)Identifique y cambie los principales parámetros de su segunda interface de red (ens34). Configure una segunda interface lógico. Al terminar, déjelo como estaba.

Primero, para ver la configuración actual de nuestras interfaces de red podemos ejecutar el comando:

```
lsi@debian:~$ ip a s
```

Los parámetros de la interfaz **ens34** los configuramos editando el fichero /etc/network/interfaces como se muestra en el apartado "a". Para crear una interfaz lógica ejecutamos:

```
lsi@debian:~$ ifconfig ens34:0 10.11.49.253 netmask 255.255.254.0
```

De esta manera creamos la interfaz lógica 0 en la interfaz física ens34 de nuestra máquina.

Si queremos borrar alguna configuración sobre algún interfaz primero visualizamos las subredes asociadas a esta con el comando:

```
lsi@debian:~$ ip a ls ens34 | grep inet
```

Luego de tener claro que interfaz queremos eliminar de nuestra configuración ejecutamos el siguiente comando sobre nuestra interfaz.

```
lsi@debian:~$ ip a del 10.11.49.253/23 dev ens34:0
```

g) ¿Qué rutas (routing) están definidas en su sistema? Incluya una nueva ruta estática a una determinada red.

Para saber que rutas están definidas en nuestro sistema podemos hacerlo revisando la tabla de enrutamiento de este con el comando:

```
lsi@debian:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.11.48.1      0.0.0.0         UG    0      0        0 ens33
10.11.48.0      0.0.0.0         255.255.254.0   U     0      0        0 ens33
10.11.50.0      0.0.0.0         255.255.254.0   U     0      0        0 ens34
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 ens33
```

Para añadir nuevas rutas estáticas a nuestro dispositivo nos encontramos con diversas formas y comandos, la que nosotros preferimos usar es:

```
lsi@debian:~$ route add -net <IP> netmask <netmask> gw <gateaway>
```

Por ejemplo, para añadir una ruta a la red 10.11.130.0 tendríamos que hacer lo siguiente:

```
lsi@debian:~$ route add -net 10.11.130.0 netmask 255.255.254.0 gw 10.11.48.1
```

Consiguiendo así añadir dicha ruta a la tabla de enrutamiento anterior:

```
lsi@debian:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.11.48.1      0.0.0.0         UG    0      0        0 ens33
10.11.48.0      0.0.0.0         255.255.254.0   U     0      0        0 ens33
10.11.50.0      0.0.0.0         255.255.254.0   U     0      0        0 ens34
10.11.130.0     10.11.48.1      255.255.254.0   UG    0      0        0 ens33
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 ens33
```

Para borrar los cambios realizados y eliminar dicha ruta de nuestra tabla de enrutamiento basta con ejecutar el comando:

```
lsi@debian:~$ route del -net 10.11.130.0 netmask 255.255.254.0 gw 10.11.48.1
```

h)En el apartado d) se ha familiarizado con los services que corren en su sistema. ¿Son necesarios todos ellos? Si identifica servicios no necesarios, proceda adecuadamente. Una limpieza no le vendrá mal a su equipo, tanto desde el punto de vista de la seguridad, como del rendimiento.

Al principio de esta práctica ya dimos de baja un par de servicios innecesarios para nuestros equipos, como fue el avahi-daemon (estos servicios se encargan de detectar automáticamente los recursos de una red local y conectarse a ella), y también el NetworkManager (es una aplicación grafica de escritorio que sirve para administrar y gestionar las conexiones de nuestro equipo), para hacer un estudio más exhaustivo estudiaremos los tiempos de ejecución de cada servicio y los estados en los que se encuentran.

Como vimos anteriormente para ver los tiempos de ejecución tenemos el comando:

```
lsi@debian:~$ systemd-analyze blame
```

Luego para ver los servicios en "enabled":

```
lsi@debian:~$ systemctl list-unit-files --state=enabled
```

Servicios deshabilitados en este apartado:

- **ssa.service:**
- **bluetooth.service:** no se necesita en nuestro caso.
- **switcheroo-control.service:** verifica la disponibilidad de GPU dual.
- **ModemManager.service:** controla conexiones 3G y 4G, en nuestro caso no contamos con conexión móvil.
- **wpa_supplicant.service:** controla el roaming y la autenticación IEEE 802.11 del controlador de wland.

- **cups-browsed.service:** ayuda al descubrimiento de impresoras remotas en la red.
- **cups.service/cups.socket:** este servicio permite configurar un equipo como servidor de impresión en Linux, además de poder instalar impresoras y compartirlas.
- **tpm2-abrmd.service:** es un servicio que se comunica con el chip TPM, en nuestro caso las maquinas carecen de él.

- **accounts-daemon.service:** es útil deshabilitarlo, ya que en caso de que sea necesario este servicio, se inicia de forma automática por DBUS (mecanismo de comunicación entre procesos).

- **console-setup.service:** permite realizar personalizaciones en la consola. En nuestro caso, no necesitamos hacer ninguna modificación.

i)Diseñe y configure un pequeño "script" y defina la correspondiente unidad de tipo service para que se ejecute en el proceso de botado de su máquina.

Para este apartado obviaremos el proceso de creación del script y nos centraremos en la configuración del servicio y botado.

Para comenzar tenemos el script guardado y con permisos de ejecución en la ruta */usr/local/bin/stats2.sh*.

Nos dirigimos a la carpeta */lib/systemd/system* donde se encuentran todos los servicios del sistema y creamos un nuevo servicio para nuestro script:

```
lsi@debian:/lib/systemd/system$ nano
```

Así creamos un archivo vacío y escribimos lo siguiente:

```
[Unit]
Description=Servicio de script stats
After=network.target

[Service]
Type=forking
ExecStart=/usr/local/bin/stats2.sh

[Install]
WantedBy=multi-user.target
```

Los parámetros de este archivo son los siguientes:

- ***Description***: Puedes indicar una descripción simple del servicio.
- ***After***: Sirve para controlar el orden de ejecución.

- ***Wants***: Esto viene a ser una versión ligera de *Requires*. Significa que systemd intentará arrancar todas las unidades listadas aquí, en el momento de arrancar la unidad actual.
- ***ExecStart***. Ruta completa del script con los argumentos que requiera.
- ***WantedBy***: indica el target que solicita la ejecución del servicio.

Finalmente guardamos dicho archivo como mi-serivicio.service, finalmente se

```
lsi@debian:~$ sudo systemctl daemon-reload
```

reinician los Daemon y se habilita el servicio para que inicie con cada

```
lsi@debian:~$ sudo systemctl enable mi-servicio.service
```

inicio del sistema con el comando:

j)Identifique las conexiones de red abiertas a y desde su equipo.

Para identificar las conexiones de red abiertas tenemos que utilizar una herramienta llamada **netstat** que entrega estadísticas básicas sobre la totalidad de las actividades de red. También puede entregar información acerca de los puertos y direcciones a través de los cuales se ejecutan las conexiones **TCP** y **UDP**, al igual que los puertos abiertos para solicitudes.

El comando más utilizado para ver todas las conexiones de red activas y puertos abiertos es:

```
lsi@debian:~$ netstat -ano
```

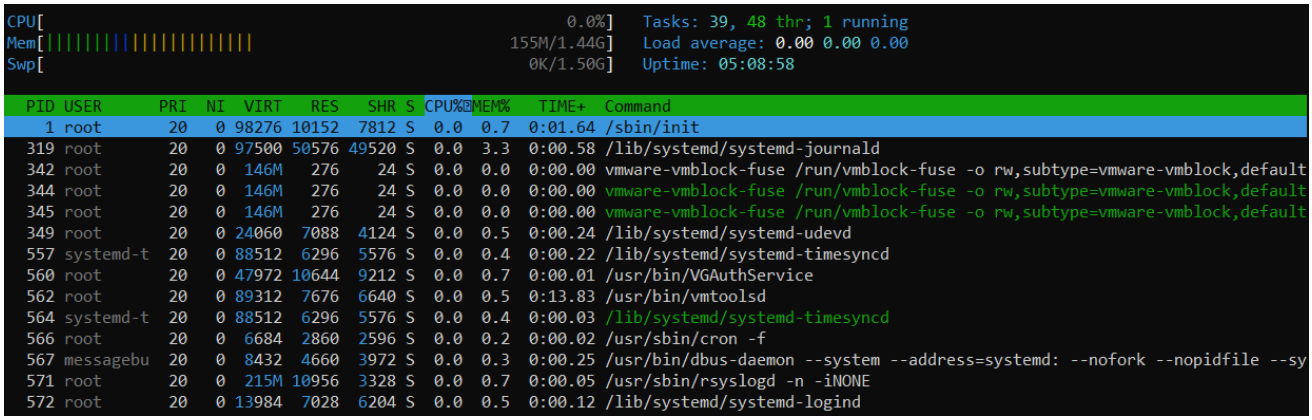
además de este hay diferentes argumentos con el que ejecutar netstat que nos proporcionará diferente información de interés:

- **netstat -p <protocolo>:** lista las conexiones para el protocolo especificado.
- **netstat -t:** lista todos los puertos TCP.
- **netstat -u:** lista todos los puertos UDP.
- **netstat -a:** muestra todas las conexiones activas y los puertos abiertos.
- **netstat -e:** muestra estadísticas por interfaz (paquetes recibidos y enviados).
- **netstat -n:** visualización numérica de las direcciones y números de puerto.
- **netstat -q:** lista todas las conexiones, todos los puertos TCP en escucha y todos los puertos TCP abiertos que no están ene escucha.
- **netstat -l:** lista solo los puertos que están escuchando.

k)Nuestro sistema es el encargado de gestionar la CPU, memoria, red, etc., como soporte a los datos y procesos. Monitorice en "tiempo real" la información relevante de los procesos del sistema y los recursos consumidos. Monitorice en "tiempo real" las conexiones de su sistema.

Para realizar la tarea de monitorización en tiempo real de nuestro sistema existen variedad de opciones, las mas utilizadas son los comandos:

- **top:** este comando te permite ver las tareas del sistema que se ejecutan en tiempo real.
- **htop:** es una herramienta muy parecida a la anteriormente explicada con la diferencia que es mas interactiva y visual y resulta bastante mas cómoda a la hora de visualizar procesos y cerrar alguno que sea de interés.



l)Un primer nivel de filtrado de servicios los constituyen los tcp-wrappers. Configure el tcp-wrapper de su sistema (basado en los ficheros hosts.allow y hosts.deny) para permitir conexiones SSH a un determinado conjunto de IPs y denegar al resto. ¿Qué política general de filtrado ha aplicado? ¿Es lo mismo el tcp-wrapper que un firewall? Procure en este proceso no perder conectividad con su máquina. No se olvide que trabaja contra ella en remoto por SSH.

Editamos el fichero /etc/hosts.allow poniéndole lo siguiente:

```
#MI_IP

sshd: 127.0.0.1 :spawn /bin/echo "($(date)|MILOCAL) from %a service %d
>>/var/log/conexion_ssh.txt

sshd: 10.11.49.48 :spawn /bin/echo "($(date)|MIIP) from %a service %d
>>/var/log/conexion_ssh.txt

sshd: 10.11.51.48 :spawn /bin/echo "($(date)|MIIP34) from %a service %d
>>/var/log/conexion_ssh.txt

#IP_COMPI

sshd: 10.11.49.116 :spawn /bin/echo "($(date)|COMPIIP) from %a service %d
>>/var/log/conexion_ssh.txt

sshd: 10.11.51.116 :spawn /bin/echo "($(date)|COMPIIP34) from %a service %d
>>/var/log/conexion_ssh.txt

#IP_VPN

sshd: 10.30.:spawn /bin/echo "($(date)|VPN) from %a service %d
>>/var/log/conexion_ssh.txt


#IP_EDUROAM
sshd: 10.20.32.0/21:spawn /bin/echo "($(date)|VPN) from %a service %d
>>/var/log/conexion_ssh.txt
```

Una vez configurado el fichero hosts.allow, configuramos el deny, negando todo el resto de las entradas, permitiendo así solo las configuradas anteriormente:

```
ALL:ALL : twist /bin/echo "%h has been banned from this server!"
```

spawn – Lanza un comando de shell como un proceso hijo.

twist – Remplaza el servicio que se solicita por el comando especificado.

La política de bloqueo utilizada es una muy restrictiva ya que bloqueamos cualquier entrada diferente a las que habilitamos expresamente en el fichero ***host.allow.***

Los TCP Wrappers suelen utilizarse para filtrar direcciones ip y hostnames. Y un firewall es un dispositivo de hardware o un software que nos permite gestionar y filtrar la totalidad de tráfico entrante y saliente que hay entre 2 redes u ordenadores de una misma red. Los wrappers también funcionan a nivel de app y los firewalls no.

m)Existen múltiples paquetes para la gestión de logs (syslog, syslog-ng, rsyslog). Utilizando el rsyslog pruebe su sistema de log local.

Rsyslog e encarga de implementar el protocolo de syslog básico, lo extiende con filtrado basado en un contenido dado, con capacidades de filtrado enriquecido, opciones de configuración flexibles y agrega características como uso de TCP para el transporte.

Para probar el funcionamiento del log local de nuestro sistema basta con generar un log a algún recurso dado y verificar si efectivamente esta archivado en el log correspondiente:

```
lsi@debian:~$ logger -p mail.err "Prueba de log local"
```

Para verificar el correcto funcionamiento ingresamos en el usuario root y visualizamos el fichero /var/log/mail.err

```
lsi@debian:~$ cat /var/log/mail.err
Sep 25 18:44:12 debian lsi: Prueba de log local
```

n)Configure IPv6 6to4 y pruebe ping6 y SSH sobre dicho protocolo. ¿Qué hace su tcp-wrapper en las conexiones SSH en IPv6? Modifique su tcp-wapper siguiendo el criterio del apartado.

Añadimos al fichero interfaces la nueva interfaz 6to4:

```
auto lo ens33 ens34 6to4
iface 6to4 inet6 v4tunnel
address 2002:a0b:3130::1
netmask 16
endpoint any
local 10.11.49.48
```

Y luego modificamos el fichero hosts.allow:

```
#MI_IP
sshd: [2002:a0b:3130::1] :spawn /bin/echo "($(date)|MIPV6) from %a service %d
>>/var/log/conexion_ssh.txt
#MI_COMPI
sshd: [2002:a0b:3174::1] :spawn /bin/echo "($(date)|MIPV6) from %a service %d
>>/var/log/conexion_ssh.txt
```

Mi tcp-wrapper para poder funcionar en IPv6 tenemos que añadirle las líneas escritas anteriormente en el host.allow, para así permitir las conexiones. De la otra manera denegará todo tipo de conexiones ipv6.

h).¿Necesita IPv6?. ¿Cómo se deshabilita IPv6 en su equipo

En la actualidad se sigue utilizando el protocolo IPv4 como protocolo de nivel de red principal, es por esto por lo que en nuestro caso y en general no es necesario tener el protocolo IPv6 habilitado, para ello editaremos el archivo ***/etc/sysctl.conf*** y le añadiremos al final del mismo las siguientes líneas:

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

PARTE GRUPAL:

a) En colaboración con otro alumno de prácticas, configure un servidor y un cliente NTP.

En el servidor configuramos:

```
#server ntp.your-provider.example
server 127.127.1.1 minpoll 4 prefer
fudge 127.127.1.1 stratum 10


restrict default ignore
restrict 10.11.49.116 mask 255.255.255.255 noquery nopeer
restrict 127.127.1.1 mask 255.255.255.255 noserve nomodify

# Local users may interrogate the ntp server more closely.
restrict 127.0.0.1 mask 255.255.255.255 noserve nomodify
restrict ::1

# Needed for adding pool entries
restrict source notrap nomodify noquery
```

En el cliente configuramos:

```
server 10.11.49.48 minpoll 4 prefer

# Local users may interrogate the ntp server more closely.
restrict 127.0.0.1
restrict ::1

# Needed for adding pool entries
restrict source notrap nomodify noquery

#Acceso desde la misma máquina, para supervisión
restrict 127.0.0.1 mask 255.255.255.255

#Acceso al servidor
restrict 10.11.49.48
```

b) Cruzando los dos equipos anteriores, configure con rsyslog un servidor y un cliente de logs.

En el servidor configuramos:

```
#Provides TCP syslog reception
module(load=”imtcp”)
input(type=”imtcp” port=”514”)
$AllowedSender TCP, 127.0.0.1, 10.11.49.48


##RULES##
$template remote,"/var/log/%FROMHOST-IP%/%PROGRAMNAME%.log"
:inputname, isequal, "imtcp" ?remote
&stop
```

En el cliente configuramos:

```
*.* action(type="omfwd"
    queue.type="LinkedList"
    queue.filename="cola_logs"
    queue.maxFileSize="1m"
    action.resumeRetryCount="-1"
    queue.saveonshutdown="on"
    Target="10.11.49.116" Port="514" Protocol="tcp")
```

c) Haga todo tipo de propuestas sobre los siguientes aspectos.: ¿Qué problemas de seguridad identifica en los dos apartados anteriores? ¿Cómo podría solucionar los problemas identificados?

Qué problemas de seguridad identifica en los dos apartados anteriores?

Apartado A:

En el apartado A algunos de los problemas que podríamos tener serían, los siguientes, las conexiones NTP, utilizan el protocolo UDP que está siendo muy usado para los ataques de denegación de servicio, es decir estos ataques se basan en la falsificación del tráfico IP (spoofing). Otro posible problema es la posibilidad de que un equipo genere tráfico con una IP que no le corresponde y que éste llegue hasta internet. Recordemos también que con UDP podemos perder paquetes muy fácilmente. En resumen, lo que puede llegar a pasar sería que en las versiones antiguas de NTP algunas redes tienen además un servicio de monitorización que permite a los administradores recopilar una lista de los 600 hosts que se han conectado al servidor.

Los ciberatacantes aprovechan esta característica haciendo un ataque reflejo: envían un paquete con una dirección IP falsa mediante la que obtienen la lista. Después, lo amplifican realizando un ataque de denegación de servicio (DDoS) que puede dejar en fuera de juego temporalmente la conexión de todas las direcciones de hosts de la lista.

Apartado B:

En el apartado B algunos de los problemas que tendríamos serían que, si hackean el servidor de logs, pueden ver todos los logs de la red. También

tenemos la posibilidad de un ataque DDoS, como se explicó en la sección anterior No hay autenticación, permitiendo que cualquiera envíe logs al servidor, no existe confidencialidad de los datos enviados, no hay integridad de los datos (alguien podría modificarlos en el camino – MITM). Aunque probablemente su problema más grave es la pobre seguridad que provee.

¿Cómo podría solucionar los problemas identificados?

Apartado A

Para mitigar un poco estos ataques se pueden aplicar medidas que limiten el tráfico NTP en las conexiones, pero se requiere que haya definido una jerarquía de servidores a nivel interno de la organización de forma el número de equipos que realizan consultas NTP al exterior sea reducido (restringir el acceso). Una buena manera para limitar el acceso es utilizando un firewall rechazando así todo tipo de conexión que no venga de una IP autorizada.

También deberíamos tener en cuenta otros aspectos como son, una correcta configuración del servidor para no correr el riesgo de que nos cojan nuestra lista de hosts. Otra solución sería la actualización de las redes y sus protocolos, estos ataques a NTP se dan en versiones antiguas y no tanto en las recientes. Y finalmente además de todo lo anterior, es imprescindible que las empresas de cualquier tamaño cuenten con una solución de ciberseguridad avanzada, activa en todos los endpoints y que sea capaz de prevenir, detectar y neutralizar los ataques en todo momento.

Aparatado B

Ya utilizamos TCP para no perder paquetes. La IPsec (abreviatura de Internet Protocol security) es un conjunto de protocolos cuya función es asegurar las comunicaciones sobre el Protocolo de Internet (IP) autenticando y/o cifrando cada paquete IP en un flujo de datos. IPsec también incluye protocolos para el establecimiento de claves de cifrado.
E de igual forma que en el apartado A podemos solucionarlo con un firewall rechazando así todo tipo de conexión que no venga de una IP autorizada.

d) En la plataforma de virtualización corren, entre otros equipos, más de 200 máquinas virtuales para LSI. Como los recursos son limitados, y el disco duro también, identifique todas aquellas acciones que pueda hacer para reducir el espacio de disco ocupado.

Ejecutamos:

- apt clean: Para borrar la cache del sistema.
- apt autoclean: Para borrar paquetes de versiones antiguas.
- apt autoremove: Para borrar sin padres que nunca se ejecutaría.
- rm -rf /usr/share/man: Para borrar los manuales del SO.

Luego para una liberación de memoria más exhaustiva podemos eliminar versiones de kernel antiguas.

Primero verificamos nuestra versión actual de kernel con:

- uname -r

Luego revisamos cuales kernels hay instalados en nuestro sistema con:

- dpkg –list | grep linux-image

Para finalmente eliminarlos con el comando:

- sudo apt purge <kernel a eliminar>

Con esto conseguimos bajar el uso del disco de 41% a un 37%