SEGURIDAD EN SISTEMAS OPERATIVOS

4º Grado en Informática – Complementos de Ing. del Software

Curso 2019-20

Práctica [1]. Administración de la seguridad en Linux.

Sesión [2]. Herramientas básicas de seguridad.

Autor[1]: Nazaret Román Guerrero

Ejercicio 1.

Resuelve las siguientes cuestiones:

a) **Utiliza esta herramienta para conocer que procesos/servicios de nuestro sistema están accediendo a la red o tienen archivos abiertos. Indicar algunos de los servicios que tenéis activos , es decir, la actividad de la red, indicando qué información da la herramienta.**

b) **Qué órdenes y opciones darías para conocer qué cuenta podría estar generando tráfico saliente malicioso de `ssh` y dónde se encuentra el archivo.**

c) **Muestra los archivos a los que está accediendo un proceso concreto y los que están en uso por un usuario.**

a) Para poder ver los archivos que está usando el sistema y que están relacionados con la red se utiliza la opción `-i` de `lsof`; nos da información sobre qué agente está usando el archivo, el `PID`, el usuario y el tipo, el nombre del archivo y el tipo de protocolo de transporte, entre otros.

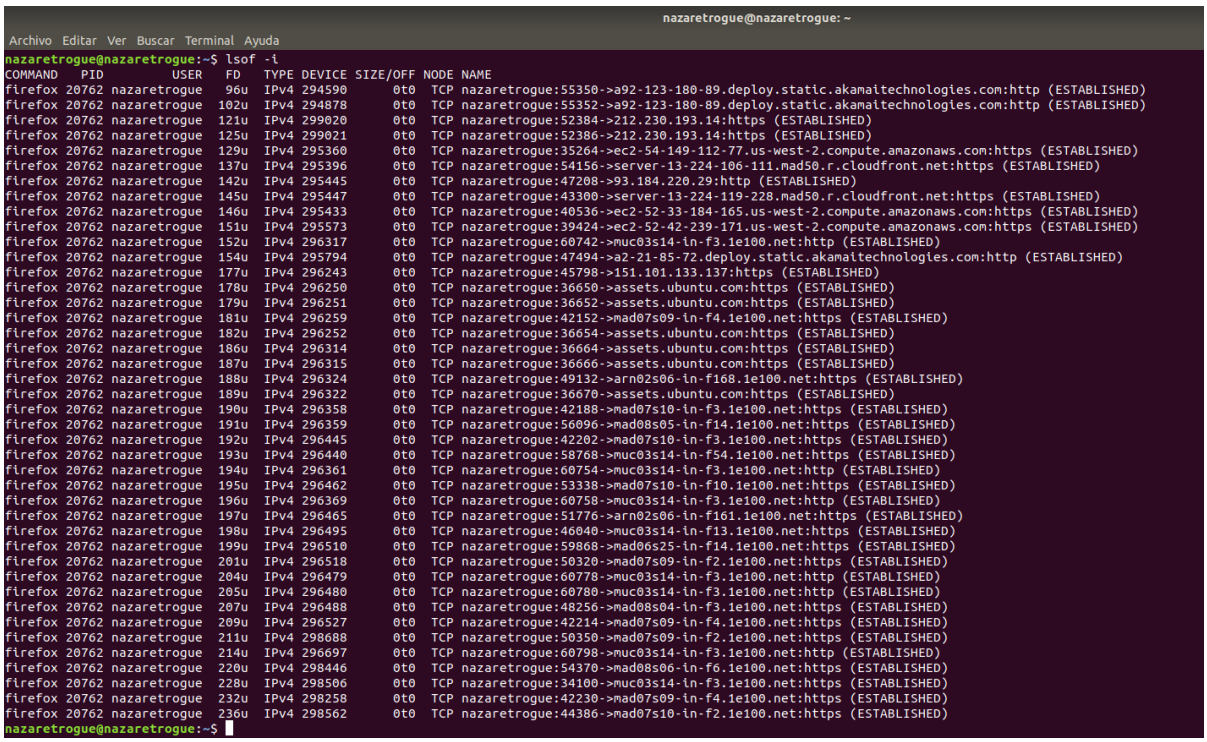
Es importante estar navegando por Internet para que haya algún archivo abierto que esté siendo usado por un agente. Si se ejecuta la orden sin que haya ningún navegador o aplicación que requiera de Internet la orden no da salida alguna, ya que no hay ningún archivo que pueda mostrar relacionado con la red.

Adicionalmente se podría usar la opción `-P`, que muestra los puertos, y la opción `-n`, que resuelve las IP y coloca el nombre de dominio en su lugar.

La salida es la que se muestra a continuación, en la imagen que hay en la siguiente página:

---

1 Como autor declaro que los contenidos del presente documento son originales y elaborados por mi. De no cumplir con este compromiso, soy consciente de que, de acuerdo con la "Normativa de evaluación y de calificaciones de los estudiantes de la Universidad de Granada" esto "conllevará la calificación numérica de cero … independientemente del resto de calificaciones que el estudiante hubiera obtenido ..."



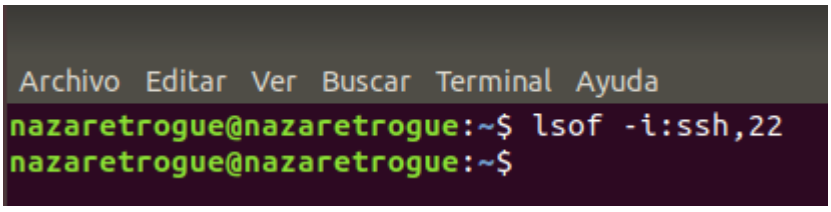
b) En este caso, se pueden utilizar distintas medidas para saber si hay alguien utilizando el servicio de `ssh` en nuestro sistema sin que nosotros lo sepamos.

Una opción es utilizar el comando `lsof` con la opción `-i`, añadiendo parámetros adicionales a esta opción, como el servicio o el puerto. El comando completo quedaría como:

```
lsof -i:ssh,22 [-P -n]
```

donde `ssh` es el servicio y 22 el puerto que éste usa; las opciones `-P` y `-n` no son necesarias como tal, pero son útiles para lo comentado en el apartado anterior.

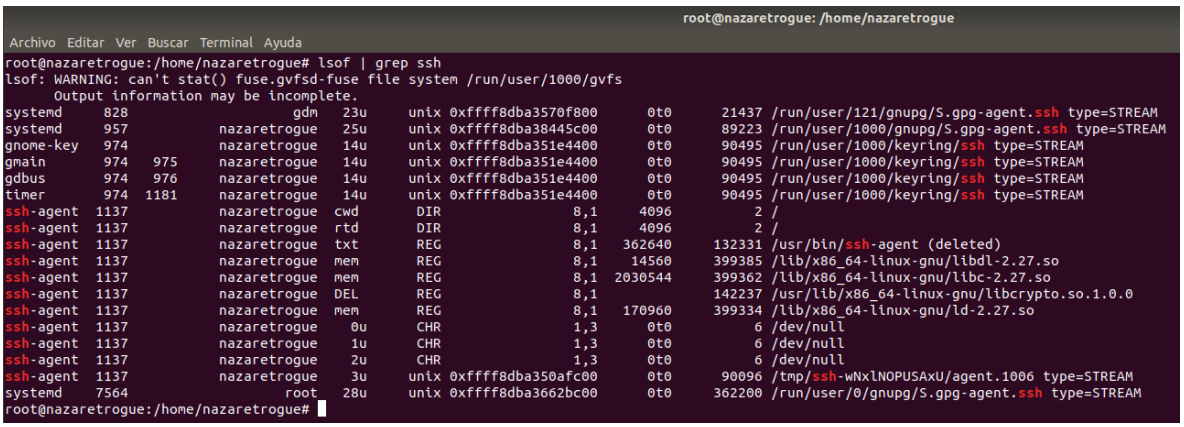
La salida del comando con estas opciones se muestra más abajo. Como se puede comprobar, no hay ningún servicio que esté utilizando el puerto 22, por lo que no hay tráfico malicioso.



También podemos utilizar el comando como

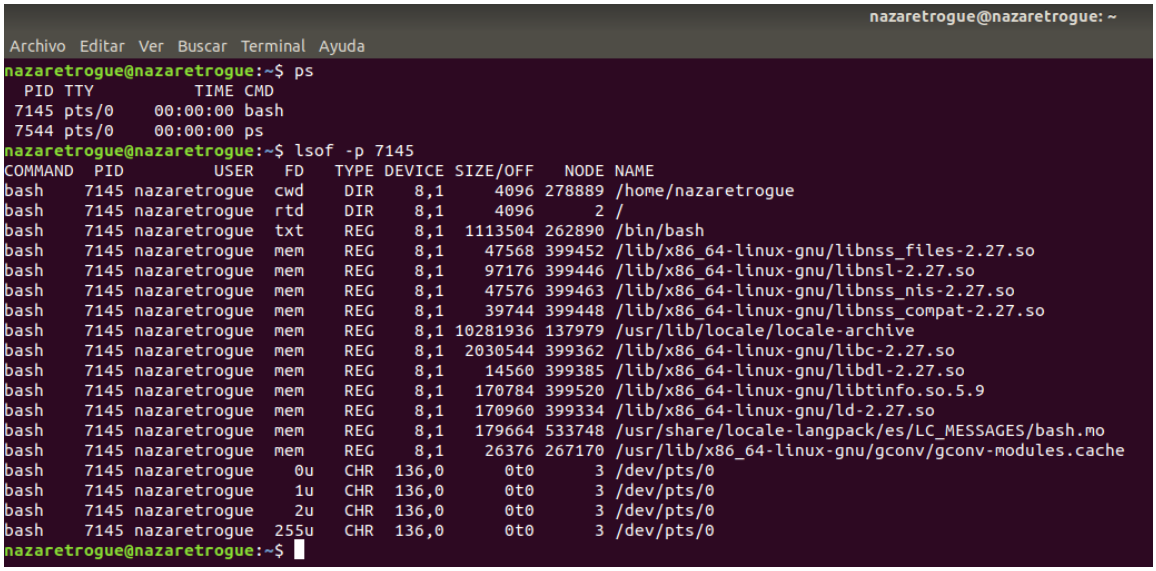
```
lsof | grep ssh
```

que muestra además los procesos que no tienen ningún archivo en ejecución para el servicio de `ssh` (es decir, que realmente no lo están usando) pero cuentan con algún tipo de archivo abierto que contiene en el nombre la palabra `ssh`:

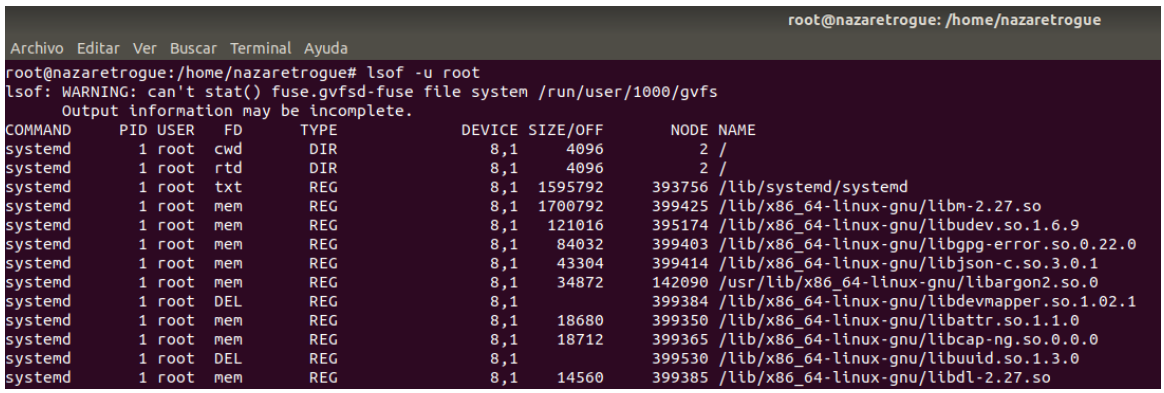


Como se puede ver, se muestra el usuario, la ruta del archivo, el tipo o el `PID` del proceso que lo está usando.

c) Para mostrar todos los archivos abiertos por un proceso se utiliza la opción `-p <PID>` (en este caso en minúscula, ya que la mayúscula es usada para los puertos de los servicios). Al utilizar la opción `-p` hay que añadir el `PID` del proceso del que queremos conocer los archivos abiertos. En el ejemplo de la imagen, se ha utilizado el `PID` del `bash` que se estaba usando:



Para mostrar los archivos abiertos por un usuario en concreto, se usa la opción `-u <user>`, donde `user` es el usuario que buscamos. En este caso, se muestra un fragmento de los archivos abiertos por `root`:

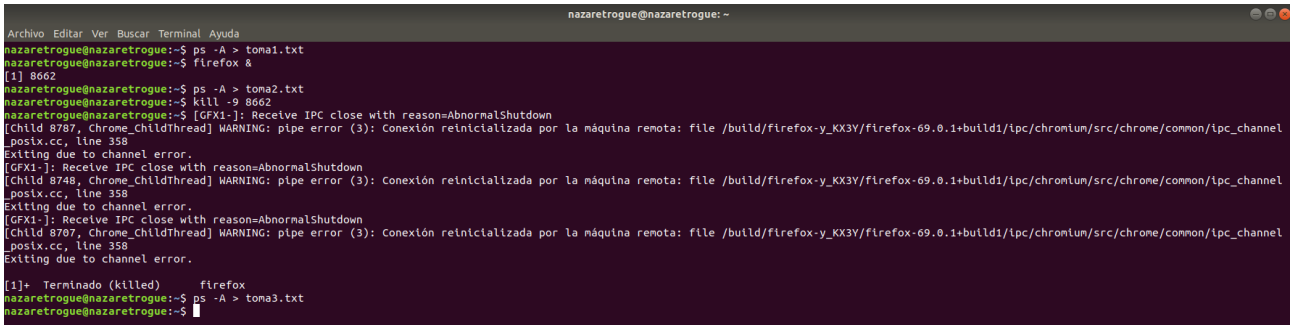


Ejercicio 2.

Ejecuta la orden `ps` para conocer los procesos de sistema habituales que vamos a encontrar en nuestro sistema y que por tanto no deberán considerarse sospechosos en nuestras labores de seguridad. Toma tres intantáneas de tu sistema en tres momentos diferentes y compáralas. ¿Hay diferencias (utiliza `diff`)? ¿Cuál es la causa de las mismas? (Indicarlo en términos de procesos en ejecución).

Para mostrar todos los procesos activos en el sistema, se utiliza la opción `-A` o la opción `-e` (ambas muestran exactamente la misma información). Para poder ver si hay diferencia entre las distintas ejecuciones del comando y saber si hay nuevos procesos funcionando o si se ha cerrado alguno, volcamos la salida en archivos para poder hacer después `diff`.

El primer volcado se lleva a cabo con la máquina recién encendida. El segundo después de abrir el navegador de la máquina, y la última, después de cerrar dicho navegador (matando el proceso).



Una vez sacadas las tomas, buscamos las diferencias entre los archivos.

Como se puede comprobar, existen diferencias en los tres archivos. Eso se debe a los procesos que se crean y mueren en el sistema (tanto los voluntarios, es decir, los que lleva a cabo el usuario de forma consciente, como los involuntarios, en los que el usuario no tiene conocimiento de que se han iniciado o matado procesos, ya que se encarga el sistema de manera autónoma).

```
nazaretrogue@nazaretrogue:~$ diff toma1.txt toma2.txt
96c96
<  1182 tty2     00:02:43 gnome-shell
---
>  1182 tty2     00:02:44 gnome-shell
184c184,188
<  8659 pts/0    00:00:00 ps
---
>  8662 pts/0    00:00:01 firefox
>  8707 pts/0    00:00:00 Web Content
>  8748 pts/0    00:00:00 WebExtensions
>  8787 pts/0    00:00:00 Web Content
>  8815 pts/0    00:00:00 ps
```

```
nazaretrogue@nazaretrogue:~$ diff toma1.txt toma3.txt
96c96
<  1182 tty2     00:02:43 gnome-shell
---
>  1182 tty2     00:02:45 gnome-shell
163c163
<  1579 tty2     00:00:01 ibus-engine-sim
---
>  1579 tty2     00:00:02 ibus-engine-sim
184c184
<  8659 pts/0    00:00:00 ps
---
>  8817 pts/0    00:00:00 ps
```

```
nazaretrogue@nazaretrogue:~$ diff toma2.txt toma3.txt
96c96
<  1182 tty2     00:02:44 gnome-shell
---
>  1182 tty2     00:02:45 gnome-shell
163c163
<  1579 tty2     00:00:01 ibus-engine-sim
---
>  1579 tty2     00:00:02 ibus-engine-sim
184,188c184
<  8662 pts/0    00:00:01 firefox
<  8707 pts/0    00:00:00 Web Content
<  8748 pts/0    00:00:00 WebExtensions
<  8787 pts/0    00:00:00 Web Content
<  8815 pts/0    00:00:00 ps
---
>  8817 pts/0    00:00:00 ps
```

Ejercicio 3.

Instalar y ejecutar la citada herramienta en vuestro sistema de cara a:

a) Mostrar qué vulnerabilidades hay en vuestro sistema, asignarle un grado de severidad (en una escala: alta, media o baja) e indicar qué pasos debemos dar para eliminarlas.

b) En clase de teoría vimos la vulnerabilidad `Shellshock`, indicar si la herramienta citada comprueba dicha vulnerabilidad y explicar cómo lo hace (esto nos servirá para conocer como prodríamos desarrollar nuestro propio test). Consejo, revisar el contenido del archivo de la herramienta `include/tests_shells`.

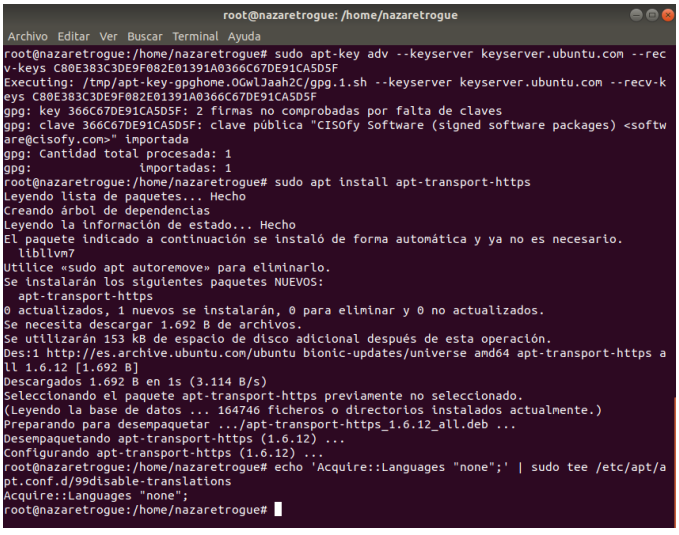
c) Suponiendo que nuestro sistema tiene un antivirus, Avx, no contemplado por la herramienta. Indicar qué debemos hacer para que la herramienta lo detecte y no muestre en el informe final que no tenemos solución (antivirus).

a) Para llevar a cabo un test de vulnerabilidades en `lynis` utilizamos la opción `--checkall`, cuya resultado es el siguiente:

```
-[ Lynis 2.6.2 Results ]-

Warnings (4):
----------------------------
! Version of Lynis is very old and should be updated [LYNIS]
    https://cisofy.com/controls/LYNIS/

! No password set for single mode [AUTH-9308]
    https://cisofy.com/controls/AUTH-9308/

! Couldn't find 2 responsive nameservers [NETW-2705]
    https://cisofy.com/controls/NETW-2705/

! iptables module(s) loaded, but no rules active [FIRE-4512]
    https://cisofy.com/controls/FIRE-4512/
```

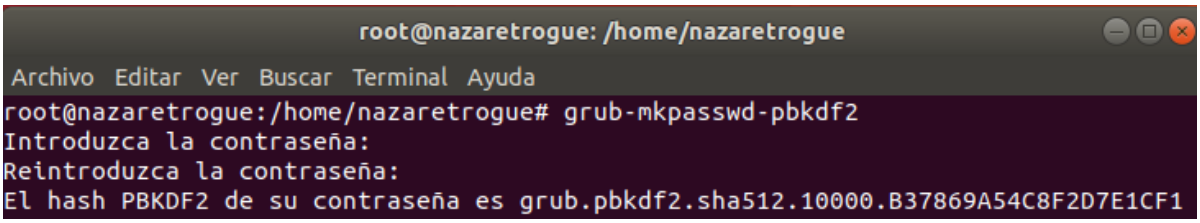
Arreglar estas vulnerabilidades es sencillo.

- La versión de `lynis` es antigua y debe ser actualizada. Es un grado de severidad alto, debido a que un sistema desactualizado supone una presa fácil. Para arreglarlo solo tenemos que actualizar `lynis`.



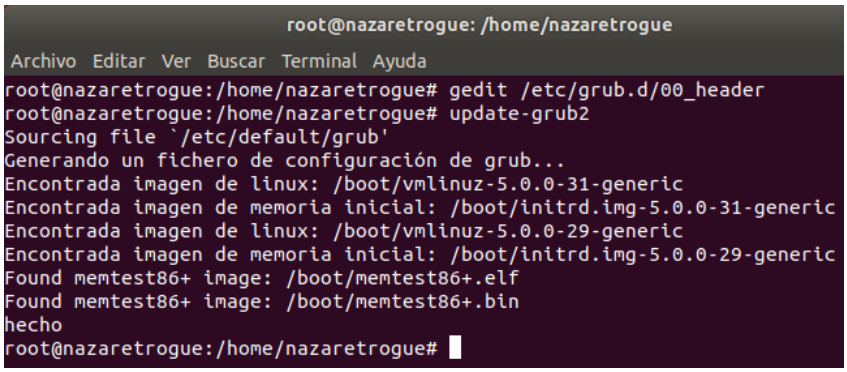
- El segundo `warning` nos indica que el `grub` no tiene contraseña (de hecho, ni siquiera está configurado), lo que puede suponer una amenza. No obstante, en este caso la amenaza es de severidad media-baja, puesto que al no estar configurado no se puede acceder desde el `grub` a la `BIOS`, directamente entra al sistema operativo de la máquina. No obstante, el arreglo es sencillo, solo hay que configurar el `grub`.

  Para ello, primero generamos una clave pública.



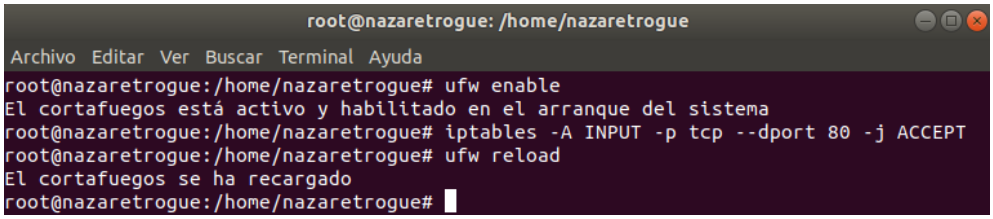
  Tras esto, la incluimos en el fichero `/etc/grub.d/00_header` y como se ve en la imagen y actualizamos el `grub`.

```
cat << EOF
set superusers="admin"
password_pbkdf2 admin
grub.pbkdf2.sha512.1000
EOF
```

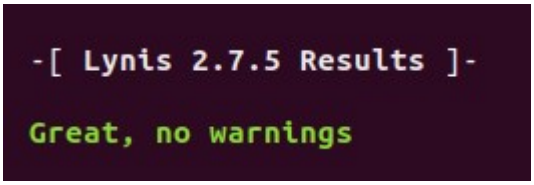


- El tercer `warning` se arregla automáticamente al actualizar la versión de `lynis`. Al tener una versión antigua, no leía correctamente el fichero donde se encuentran las IPs de los servidores DNS; una vez actualizada, ese fallo desaparece. Esta vulnerabilidad es de severidad baja puesto que con que un solo DNS funcione, todo irá bien.
- El último `warning` nos advierte que no tenemos ninguna regla en el cortafuegos. Su severidad alta puesto que nuestro sistema está expuesto. De hecho el cortafuegos ni siquiera está activado, por lo que el primer paso es habilitarlo. Tras esto, se añade una regla que acepta todo el tráfico cuyo protocolo sea `TCP` y proveniente del puerto 80.

  Una vez añadida, se recarga el cortafuegos para que se active la nueva configuración.

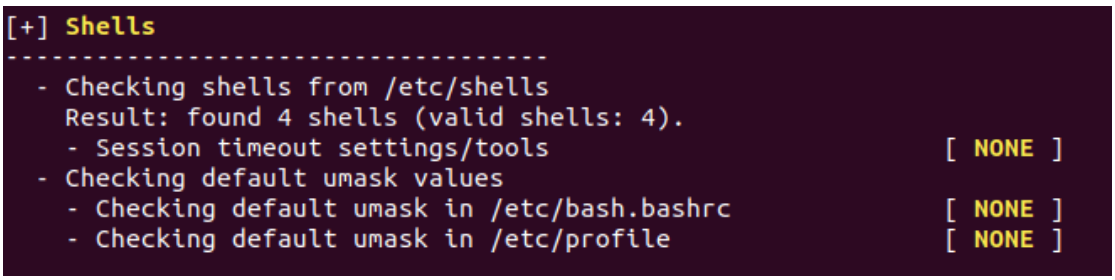


Una vez tratadas todos los `warnings` (y ejecutando el test después de eliminar cada una para comprobar que lo que se está haciendo va por buen camino), volvemos a ejecutar el test, obteniendo la siguiente salida:



Por lo que nuestro sistema ahora no tiene vulnerabilidades.

b) En el archivo de `lynis tests_shells`, se llevan a cabo tests para vulnerabilidades (concretamente para la 6211, la 6220 y la 6230) pero no para `shellshock`, cuyo identificador es el 6271. También se puede observar al ejecutar un test (como el del apartado anterior), cuya salida es:



Por lo que podemos ver que no está el test para `shellshock` y que, por tanto, no lo comprueba.

c) Para evitar que la herramienta detecte que no hay antivirus cuando éste es desconocido, hay que modificar el archivo `include/tests_malware`.

Con este archivo se detectan ciertos antivirus, pero si tenemos uno que no está contemplado, al ejecutar los tests nos avisará de que no hay antivirus en nuestra máquina cuando en realidad sí lo hay. Para evitar esto, una de las variables globales que está iniciada a 0 (a false) se cambiaría a 1 (true) por lo que se habría establecido ya que el antivirus está instalado (aunque no se está comprobando realmente en los tests por lo que no es la mejor opción).

Una segunda opción, más exhaustiva, es crear a mano un test y añadirlo al fichero mencionado, de manera que se ejecute también cuando se lance la herramienta de `lynis`.

Ejercicio 4.

Instalar y ejecutar la citada herramienta en vuestro sistema de cara a:

a) Realizar un análisis del sistema para ver si está o no comprometido.

b) De los avisos, soluciona los que sean falsos positivos, bien eliminando los tests, bien ajustándolos adecuadamente.

a) Para comprobar si el sistema está comprometido se analiza con la opción `-c` (o `--check`) y da como resultado el siguiente resumen:

```
System checks summary
=====================

File properties checks...
    Files checked: 149
    Suspect files: 1

Rootkit checks...
    Rootkits checked : 479
    Possible rootkits: 0

Applications checks...
    All checks skipped

The system checks took: 1 minute and 41 seconds

All results have been written to the log file: /var/log/rkhunter.log

One or more warnings have been found while checking the system.
Please check the log file (/var/log/rkhunter.log)
```

Como podemos observar, no hay `rootkits` en el sistema.

El único `warning` registrado es sobre `/usr/bin/lwp-request`. El archivo de `log` `/var/log/rkhunter.log` explica con más detalle el `warning`: el comando `/usr/bin/lwp-request` ha sido sustituido por un script ejecutable escrito en Perl.

```
/usr/bin/mawk                                             [ OK ]
/usr/bin/lwp-request                                      [ Warning ]
/usr/bin/bsd-mailx                                        [ OK ]
```

```
[15:24:52]   /usr/bin/lwp-request                         [ Warning ]
[15:24:52] Warning: The command '/usr/bin/lwp-request' has been replaced by a script: /usr/bin/lwp-request: Perl script text executable
[15:24:52]   /usr/bin/bsd-mailx                           [ OK ]
```

b) Para solucionar esto, hay que modificar el archivo `/etc/rkhunter.conf` y añadir el script en la parte `SCRIPTWHITELIST`. Como se puede observar, el script ya estaba añadido y solo es necesario descomentar la línea.

```
#
SCRIPTWHITELIST=/bin/egrep
SCRIPTWHITELIST=/bin/fgrep
SCRIPTWHITELIST=/bin/which
SCRIPTWHITELIST=/usr/bin/ldd
#SCRIPTWHITELIST=/usr/bin/lwp-request
SCRIPTWHITELIST=/usr/sbin/adduser
#SCRIPTWHITELIST=/usr/sbin/prelink
#SCRIPTWHITELIST=/usr/sbin/unhide.rb
```

Una vez hecho esto, volvemos a ejecutar `rkhunter -c` para comprobar que la medida ha sido efectiva.

El resumen del nuevo chequeo no contiene ningún archivo sospechoso ni tampoco se indica que haya ningún `warning` en el archivo de `log` (última línea), como se puede ver en la

siguiente imagen:

```
System checks summary
=====================

File properties checks...
    Files checked: 149
    Suspect files: 0

Rootkit checks...
    Rootkits checked : 479
    Possible rootkits: 0

Applications checks...
    All checks skipped

The system checks took: 1 minute and 3 seconds

All results have been written to the log file: /var/log/rkhunter.log

No warnings were found while checking the system.
```

Además, mientras se ejecuta el test podemos comprobar que, en efecto, el comando `/usr/bin/lwp-request` no da `warning` alguno:

Por lo que está arreglado y nuestro sistema ahora es seguro.