SEGURIDAD EN SISTEMAS OPERATIVOS

4º Grado en Informática

Curso 2019-20

Práctica 1. Administración de la seguridad en Linux

Sesión 2. Herramientas básicas de seguridad.

1.- Introducción

En esta sesión, veremos algunas herramientas útiles para mantener, vigilar y fortalecer la seguridad de nuestro sistema Linux.

2.- Listando los archivos abiertos

La orden **`lsof`** permite listar los ficheros abiertos por nuestro sistema. Para más detalles, podemos mirar el manual de la orden y ver que nos permite hacer.

Por ejemplo, podemos ver las conexiones abiertas en nuestro sistema con `lsof -i`. Esta opción nos permitirá saber si algún servicio o proceso esta funcionado cuando no debería.

Actividad 1.- Resuelve las siguientes cuestiones:

(a) Utiliza esta herramienta para conocer que procesos/servicios[1] de nuestro sistema están accediendo a la red o tiene archivos abiertos. Indicar algunos de los servicios que tenéis activos, es decir, la actividad de la red, indicando que información da la herramienta.

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

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

3.- Conociendo los procesos y servicios de nuestro sistema

Una vieja conocida nuestra es la orden ps. Con ella listamos los procesos que se están ejecutando en el sistema y entre otra información permite saber cuando se han iniciado, por ejemplo si ejecutamos `ps -eau`.

Actividad 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 instantáneas de tu sistema en tres momentos diferentes y comparalas. ¿hay diferencias (utiliza `diff`)? ¿Cual es la causa de las mismas? (indicarlo en términos de procesos en ejecución).

1 Esto nos puede permitir saber si hay trafico sospechoso.

4. Lynis

Lynis es una herramienta de auditoría *open source*[2]. Su uso principal es auditar y fortalecer el sistema de base Linux. Cuando se ejecuta, esta realiza una amplia gama de controles de seguridad (auditoría, test de conformidad -PCI, HIPAA, Sox, …-, escáner de vulnerabilidades, y fortalecimiento del sistema) y va mostrándolos de forma ordenada al usuario. La mayoría de estas pruebas son parte de las guías y estándares de seguridad.

La herramienta puede descargarse de https://cisofy.com/lynis/ o bien estará disponible en la clase de prácticas. Para aquellos que utilicéis vuestro equipos, es muy posible que este disponible en los repositorios de la distribución que usáis (por ejemplo, en Ubuntu, `$ apt-get install lynis`).

Si la instalamos bien por descarga o copia del archivo tar, solo habrá que descomprimirlo para su uso, por ejemplo:

```
$ mkdir -p /usr/local/lynis
$ cd /usr/local
$ tar xfvz lynis-<version>.tar.gz
$ lynis audit --quit
```

Algunas órdenes que admite son:

Orden	Descripción
`audit system`	Realiza una auditoría del sistema
`show commands`	Muestra las órdenes disponibles
`show help`	Suministra una pantalla de ayuda
`show profiles`	Muestra los perfiles descubiertos
`show settings`	Lista los ajustes activos de los perfiles
`show version`	Muestra la versión actual de Lynis

A continuación se muestra la tabla con algunas opciones:

Opción	Abreviatura	Descripción
`--auditor "nombre auditor"`		Asigna el nombre del auditor al informe de auditoría
`--cronjob`		Ejecuta Lynis como `cronjob` (incluye `-c -Q`)
`--debug`		Muestra información de depuración
`--help`	`-h`	Muestra los parámetros válidos
`--man-page`		Ver página de manual
`--no-colors`		No utilizar colores (útil al imprimir el documento)
`--pentest`		Realiza un test de penetración (no privilegiado)
`--quick`	`-Q`	No esperar entradas del usuario, salvo errores
`--quiet`	`-q`	Solo muestra *Warnings* (includes `--quick`, pero no espera)
`--reverse-colors`		Utilizar esquema de color diferente en fondos mas ligeros
`--verbose`		Muestra más pantallas de salida

2 Otras herramientas con similitudes a Lynis son Bastille, Nessus y OpenVas.

Como consejo, si Lynis no esta instalado como paquete (con página de manual incluida), utilizar use `--man` o `nroff -man ./lynis.8`. **Utiliza la orden `show options` para ver todos los parámetros de Lynis.**

Los pasos que sigue la herramienta son:

1. Determina el SO.
2. Busca herramientas y utilidades disponibles.
3. Comprueba si esta actualizada.
4. Ejecuta los tests de los plugins.
5. Ejecuta tests por categorías.
6. Informa del estado del escaneo de seguridad (tanto en pantalla como usando el archivo de log).

Una de las ventajas de la herramienta es que permite al usuario incorporar nuevas comprobaciones de seguridad y no requiere de otros servicios o herramientas instalados previamente.

Una vez completado el análisis, la herramienta almacena la información en los archivos:

- */var/log/lynis.log* Información de prueba y depuración
- */var/log/lynis-report.data* Datos del informe

Podéis encontrar una documentación más detalla de la herramienta en :

http://cisofy.com/documentation/lynis/

http://linoxide.com/how-tos/lynis-security-tool-audit-hardening-linux/

http://www.welivesecurity.com/la-es/2014/09/01/tutorial-de-lynis-aprende-auditar-seguridad-linux/
http://www.youtube.com/watch?v=eYNwzUUZKto

Actividad 3: Instalar y ejecutar la citada herramienta en vuestro sistema de cara a:

a) Mostrar que 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* (CVE-2014-6271), indicar si la herramienta citada comprueba dicha vulnerabilidad y explicar cómo lo hace (esto nos servirá para conocer como podríamos 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).

---

5. Anti-rootkits

Los *rootkits* son un conjunto de utilidades utilizadas por un atacante para mantener altos privilegios de acceso (root) con el objetivo no tanto de ser administrador como de permanecer en el sistema tanto tiempo cuanto sea posible. De esta forma, el atacante utiliza los recursos del sistema sin ser detectado.

Para alcanzar la "invisibilidad", los rootkits toman diversas medidas para esconderse ellos mismos. Comienzan por enmascarar cualquier archivo y directorio relacionado con ellos. Para conseguirlo, se alteran binarios como `ls` o `find`. También se alteran herramientas como `ps` o `lsmod` para evitar que se muestren los procesos o módulos relacionados. Además, muchos de ellos instalan *backdoors* (puertas traseras) que permiten al atacante entrar al sistema en cualquier instante esquivando los mecanismo de autenticación para, entre otras cosas, no dejar rastro en los logs.

La detección del malware en Linux, podemos hacerla:

1. Localmente: debemos ejecutar software especial (chkrootkit[3], ClamAV[4], Linux Malware Detect -LMD[5]-, rkhunter[6]4, uhide[7]) destinado a escanear el sistema de forma regular. Chkrootkit y rkhunter se centran en rootkits y puertas traseras de Linux, CalmAV y LMD se centran más en puertas traseras y malware genérico. Una combinación de ambas nos dará una mayor oportunidad de detectar cualquier traza de malware.
2. Por cambios: utilizar herramientas de integridad para comprobar posibles cambios en los archivos más críticos (AIDE[8], samhaim[9], OSSEC[10], tripwire[11])
3. En la red: supone una detección temprana que puede ahorrar mucho tiempo. Por ejemplo, pfSense[12], snort11[13].

¿Qué hacemos tras una infección? Centrándonos en un rootkit, que ha necesitado permisos de root para instalarse, el principal y mejor consejo es reinstalar el sistema limpio. Ya que si quisiésemos limpiar el sistema, deberíamos tener muy claro que es posible hacerlo, para lo cual deberíamos conocer como llegó el malware al sistema y cuales son todos los componentes que han sido afectados. Sino es así, es mejor reinstalar el sistema, restaurar los datos (asegurándonos de que no hay trazas del él en los *backups*) e implantar las contramedidas adecuadas.

En esta sesión, vamos a utilizar **`rkhunter`** que como hemos indicado comprueba rootkit y puertas traseras mediante el escaneo de los archivos del sistema. También escanea las conexiones de red y puertos, los componentes de arranque,y anomalías en permisos y cambios de claves. Como cualquier antivirus, tiene una base de datos de definiciones de rootkit que utiliza para compararla con los archivos, incluidos los archivos del sistema.

La instalación dependerá del sistema:

```
# sudo yum install rkhunter       # CentOS, REdHat
# sudo apt-get install rkhunter   # Ubuntu
```

Para comprobar que tenemos la última versión instalada utilizaremos el indicador –versioncheck.

```
# sudo rkhunter –versioncheck
```

Dado que el funcionamiento se basa en la comparación de los archivos con la base de datos, debemos asegurarnos que esta última esta actualizada:

```
# sudo rkhunter –update
```

Como la herramienta también comprueba la posible modificación de archivos de sistema para ver si han sido reemplazados, es necesario primero construir una base de datos de tales archivos:

```
# sudo rkhunter –propupd
```

Realizados estos pasos, ya podemos chequear el sistema con la opción --check. También se puede usar --skip-keypress para evitar que nos pida pulsar “enter” y --report-warning-only para que solo muestre los avisos:

```
# sudo rkhunter –check  [--skip-keypress][--report-warning-only]
```

El resultado se muestra en el archivo de log */var/log/rkhunter/rkhunter.log*.

El mejor enfoque es ejecutarlo via `cron`. Aunque hay varias formas de hacerlo, la mejor es ejecutarlo directamente desde la tarea cron con el indicador `--cronjob` (que enviará al root un correo con los warning). También es aconsejable utilizar las opciones `--versioncheck` para comprobar las actualizaciones y `--update` para actualizar la base de datos. Cuando se ejecuta via cron es bueno indicar que muestra. Para ello, usamos el indicador `--rwo` (*report warning only*). Una entrada típica de cron sería:

```
30 5 * * * /usr/local/bin/rkhunter --vesioncheck --cronjob --update --rwo
```

3 http://www.chkrootkit.org/
4 http://www.clamav.net/
5 https://www.rfxn.com/projects/linux-malware-detect/
6 http://rkhunter.sourceforge.net/
7 http://www.unhide-forensics.info/
8 http://aide.sourceforge.net/
9 http://www.la-samhna.de/samhain/s_download.html
10 http://ossec.github.io/
11 http://www.tripwire.org/
12 https://pfsense.org/
13 https://www.snort.org/

Cuando lo ejecutamos de forma automática, en el archivo */etc/rkhunter.conf* podemos definir la dirección de correo para enviar los warning en la variable `MAIL-ON-WARNING`. Por ejemplo, `MAIL-ON-WARNIG="usuario@correo.com"`.

Si queremos comprobar que no hay nada erróneo en la configuración de la herramienta podemos utilizar la opción `-C`, que realiza una comprobación sintáctica sobre los archivos de configuración y notifica cualquier potencial problema12.

Indicar que *rkhunter* esta diseñado solamente para generar avisos e informar de problemas potenciales. Cuando ejecutamos la herramienta por primera vez en un sistema nuevo podemos ver muchos avisos, por esto es importante ejecutarla recién instalado el sistema. Si recibimos un correo con un aviso durante la operación normal del sistema, deberemos acceder a él y ver que esta pasando. Algunos avisos son benignos y se deben a actualizaciones normales de archivos, pero no debemos ignorarlos.

- **Falsos positivos**

En concreto, podemos encontrar avisos cuando la herramienta comprueba los archivos/directorios ocultos que se encuentra en /proc pero que son habituales en algunas distribuciones. Por ejemplo, En Debian tenemos /dev/.static, /dev/.udev, y /dev/.initramfs.

Para evitar estos avisos, debemos reconfigurarlo de manera que los ignore, para lo cual editamos el archivo de configuración /etc/rkhunter.conf y quitamos los comentarios en :

```
ALLOWHIDDENDIR=/dev/.udev
ALLOWHIDDENDIR=/dev/.static
ALLOWHIDDENDIR=/dev/.initramfs
```

Otro tipo de falsos positivos en cuando hay actualizaciones de binarios (/bin, /usr/bin, etc.). Por ejemplo, cuando aparece el mensaje:

```
[04:50:29] /bin/netstat                                    [ Warning ]
[04:50:29] Warning: The file properties have changed:
[04:50:30] File: /bin/netstat
[04:50:30] Current hash: 58ad996b4c822f25760fbe4bfb9904d623aeb51b
[04:50:30]Stored     hash     :     715d86e1b178c19807a315fe339e87b3e5a12c75[04:50:31]
Current inode: 16178    Stored inode: 16175
[04:50:31] Current size: 125996    Stored size: 116940
[04:50:31] Current file modification time: 1263850324
[04:50:32] Stored file modification time : 1258110183
```

La solución en este caso es actualizar la base de datos para que reconozca las modificaciones con `rkhunter –propupd`.

En ocasiones algunos archivos binarios son sustituidos por guiones (scripts). Por ejemplo:

```
[04:51:43] /usr/bin/whatis                                 [ Warning ]
[04:51:43] Warning: The command '/usr/bin/whatis' has been replaced by a script
[04:51:43] Warning: The command '/usr/bin/whatis' has been replaced by a script:
 /usr/bin/whatis: POSIX shell script text executable
...
[04:51:48] /usr/bin/lwp-request                            [ Warning ]
[04:51:48] Warning: The command '/usr/bin/lwp-request' has been replaced by a sc
ript: /usr/bin/lwp-request: a /usr/bin/perl -w script text executable
```

En este caso, debemos inspeccionar el programa para asegurarnos de que esta bien (es un falso positivo). Si es así, para que la herramienta no se “queje”, debemos bien desactivar la prueba, bien hacerle saber que el archivo es un guión y no un binario. Para hacer esto ultimo, debemos buscar en el archivo /etc/rkhunter.conf la variable `SCRIPTWHITELIST` y añadir el camino de cada orden que sea un guión:

```
#
# Allow the specified commands to be scripts.
# One command per line (use multiple SCRIPTWHITELIST lines).
#
#SCRIPTWHITELIST=/sbin/ifup
#SCRIPTWHITELIST=/sbin/ifdown
#SCRIPTWHITELIST=/usr/bin/groups
SCRIPTWHITELIST=/usr/bin/ldd
SCRIPTWHITELIST=/usr/bin/whatis
SCRIPTWHITELIST=/usr/bin/lwp-request
```

Otras veces las queja proviene de aplicaciones desactualizadas. En este caso la mejor solución de seguridad es tener las aplicaciones actualizadas como hemos indicado con anterioridad. Pero si esto no es posible, debemos cambiar la opción `APP_WHITELIST`. Por ejemplo, para que no se queje de las versiones deactualizadas de `gpg` y `sshd`:

```
#
# Allow the following applications,or a specific version of an application,
# to be whitelisted.This option is a space-separated list consisting of the
# application names. If a specific version is to be whitelisted, then the
# name must be followed by a colon and then the version number.
#
# For example: APP_WHITELIST="openssl:0.9.7d gpg"
#
APP_WHITELIST="gpg sshd"
```

Tras realizar estos cambios oportunos, podemos chequear la nueva configuración: `rkhunter -check`.

Actividad 4: Instalar y ejecutar la citada herramienta en vuestro sistema de cara a:

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

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