

COMPROBANDO LA INTEGRIDAD DEL SISTEMA LINUX

Uno de los posibles ataques a un sistema Linux que un potencial atacante puede llegar a realizar es la modificación de archivos clave o incluso la instalación de rootkits o puertas traseras que le permitan elevar sus privilegios y poder de esa manera manejar a su antojo el sistema.

La comprobación de archivos puede ser vital para el correcto funcionamiento y es algo que no podemos obviar, sobre todo si el sistema en cuestión está expuesto a ataques tanto desde el exterior, como desde el interior por parte de los propios usuarios.

Primeramente vamos a ver nociones básicas sobre modificaciones ilícitas de archivos para luego centrarnos en aplicaciones que automatizarán y facilitarán este proceso.

Para saber si un archivo ha sido modificado podemos realizar una suma de comprobación. Se llama suma de comprobación a una cadena de texto generada mediante un algoritmo matemático, la cual comparamos con una que teníamos antes o que se nos proporciona y que tomamos como verdadera. Si el resultado de esa comprobación es distinto es que el archivo en cuestión ha sido modificado y por tanto es digno de toda sospecha, ya sea porque un atacante ha realizado cambios o porque no se ha descargado correctamente de internet.

Existen varios métodos de suma de comprobación, pero en este manual hablaremos del algoritmo MD5, muy fuerte y muy utilizado en la actualidad.

Con un sencillo ejemplo veremos enseguida el funcionamiento de este algoritmo. Imaginemos un archivo de texto llamado "archivo.txt", cuyo contenido es el siguiente:

Este archivo es el original sin ninguna modificación.

Ahora vamos a realizar su suma de comprobación:

```
vlad@localhost ~ $ md5sum archivo.txt > sumas_de_comprobacion.txt
vlad@localhost ~ $ cat sumas_de_comprobacion.txt
aa9f3672bb482a947e3518aad26cdaab archivo.txt
```

Hemos realizado la suma con el comando **md5sum** y volcado la salida hacia el archivo "**archivo.txt**". Ahora vamos a modificar el archivo simulando un cambio ilícito:

Este archivo es el original sin ninguna modificación. O tal vez si.

Y volvemos a repetir el proceso:

```
vlad@localhost ~ $ md5sum archivo.txt >>
sumas_de_comprobacion.txt
vlad@localhost ~ $ cat sumas_de_comprobacion.txt
aa9f3672bb482a947e3518aad26cdaab archivo.txt
5ef558b092a1280c79eb45e7d791b4b5 archivo.txt
```

Como podemos ver los resultados son diferentes, signo inequívoco de que algo ha pasado.

En una situación real lo ideal sería tener sumas de comprobación o incluso originales de los archivos o ejecutables mas delicados del sistema en un medio extraíble, como pueda ser un disquete, un cdrom o un dispositivo usb. De esta manera esa información tan importante para nosotros no estará al alcance de ningún atacante.

Vamos a ver ahora dos herramientas que van automatizar este proceso y que nos van ayudar a la hora de comprobar la integridad de los principales archivos del sistema, así como de buscar conocidos rootkits que pongan en peligro la seguridad de nuestro Linux. Estas herramientas son Chkrootkit y Rkhunter. Vayamos primero con Chkrootkit.

Chkrootkit es un shell script que buscará en nuestro sistema binarios modificados por los rootkits mas conocidos. Veamos mediante su comando de ayuda sus principales opciones:

```
localhost vlad # chkrootkit --help
```

```
Usage: /usr/sbin/chkrootkit [options] [test ...]
```

```
Options:
```

```
-h          show this help and exit  
-V          show version information and exit  
-l          show available tests and exit  
-d          debug  
-q          quiet mode  
-x          expert mode  
-r dir      use dir as the root directory  
-p dir1:dir2:dirN path for the external commands used by  
chkrootkit  
-n          skip NFS mounted dirs
```

Mediante la opción -l vamos a poder ver exactamente los test que Chkrootkit realizará a nuestro sistema:

```
localhost vlad # chkrootkit -l
```

```
/usr/sbin/chkrootkit: tests: aliens asp bindshell lkm rexedcs sniffer  
w55808 wted scalper slapper z2 chkutmp amd basename biff chfn  
chsh cron date du dirname echo egrep env find fingerd gpm grep  
hdparm su ifconfig inetd inetdconf identd init killall ldsopreload login  
ls lsof mail mingetty netstat named passwd pidof pop2 pop3 ps pstree  
rpcinfo rlogind rshd slogin sendmail sshd syslogd tar tcpd tcpdump  
top telnetd timed traceroute vdir w write
```

También destaca la opción -x ya que con ella vamos a poder montar una partición de otro linux que tengamos instalado en el mismo pc e indicarle a Chkrootkit cual es el directorio / por el cual tiene que empezar a buscar.

Para poner en funcionamiento este programa bastará con escribir **chkrootkit** en una línea de comandos como usuario root:

```
localhost vlad # chkrootkit
```

```
ROOTDIR is `/'
```

```
Checking `amd'... not found
```

```
Checking `basename'... not infected
```

```
Checking `biff'... not found
```

```
Checking `chfn'... not infected
Checking `chsh'... not infected
Checking `cron'... not infected
Checking `date'... not infected
Checking `du'... not infected
Checking `dirname'... not infected
Checking `echo'... not infected
Checking `egrep'... not infected
Checking `env'... not infected
Checking `find'... not infected
Checking `fingerd'... not found
Checking `gpm'... not infected
Checking `grep'... not infected
Checking `hdparm'... not infected
Checking `su'... not infected
Checking `ifconfig'... not infected
Checking `inetd'... not tested
```

.....

Y así hasta completar todo el proceso.

Vamos a ponerlo a prueba. Pondremos Ethereal en funcionamiento y haremos que se ponga a la escucha en la interfaz de red eth0 para ver como se comporta Chkrootkit. Entre todas las salidas de comprobación que nos ofrece Chkrootkit encontraremos esta:

```
Checking `sniffer'... eth0: PF_PACKET(/usr/bin/ethereal)
```

Es muy importante analizar muy bien todas las salidas. Si en este caso hubieramos utilizado la opción `-x` (modo experto), lo habriamos visto mas claro:

```
### Output of: /usr/sbin/ifpromisc
###
eth0: PF_PACKET(/usr/bin/ethereal)
not infected
```

Bien, ya que hemos visto de manera general `chkrootkit` y a buen seguro podemos ya utilizarlo para escanear nuestro sistema, vamos ahora a ver la otra útil herramienta de seguridad: `rkhunter`.

Con Rkhunter podremos hacer una buena búsqueda en nuestro Linux a la caza y captura de los mas conocidos `rootkits` y `exploits` locales mediante comparaciones MD5, permisos erroneos, ficheros ocultos, versiones de servidores vulnerables y otras comprobaciones.

Una vez instalado en nuestro sistema, lo primero que hemos de hacer es actualizarlo de la siguiente forma:

```
localhost ~ # rkhunter --update
Running updater...
```

**Mirrorfile /usr/lib/rkhunter/db/mirrors.dat rotated
Using mirror http://mirror01.mirror.rkhunter.org
[DB] Mirror file : Up to date
[DB] MD5 hashes system binaries : Up to date
[DB] Operating System information : Update available
Action: Database updated (current version: 2005101300, new
version 2005102800)
[DB] MD5 blacklisted tools/binaries : Up to date
[DB] Known good program versions : Update available
Action: Database updated (current version: 2005100400, new
version 2005111500)
[DB] Known bad program versions : Update available
Action: Database updated (current version: 2005100400, new
version 2005111500)**

**Ready.
localhost ~ #**

Ahora echemos un vistazo a su ayuda:

localhost ~ # rkhunter --help

Rootkit Hunter 1.2.7, Copyright 2003-2005, Michael Boelen

**Rootkit Hunter comes with ABSOLUTELY NO WARRANTY. This is free
software,
and you are welcome to redistribute it under the terms of the GNU
General
Public License. See LICENSE for details.**

Valid parameters:

**--checkall (-c) : Check system
--createlogfile <file>* : Create logfile (file is optional, defaults to
: /var/log/rkhunter.log)
--cronjob : Run as cronjob (removes colored layout)
--display-logfile : Show logfile at end of the output
--help (-h) : Show this help
--nocolors* : Don't use colors for output
--report-mode* : Don't show uninteresting information for
reports
--report-warnings-only* : Show only warnings (lesser output than --
report-mode,
more than --quiet)
--skip-application-check* : Don't run application version checks
--skip-keypress* : Don't wait after every test (non-interactive)
--quick* : Perform quick scan (instead of full scan)
--quiet* : Be quiet (only show warnings)
--update : Run update tool and check for database updates
--version : Show version and quit**

--versioncheck : Check for latest version

--bindir <bindir>* : Use <bindir> instead of using default binaries

--configfile <file>* : Use different configuration file

--dbdir <dir>* : Use <dbdir> as database directory

--rootdir <rootdir>* : Use <rootdir> instead of / (slash at end)

--tmpdir <tempdir>* : Use <tempdir> as temporary directory

Explicit scan options:

--allow-ssh-root-user* : Allow usage of SSH root user login

--disable-md5-check* : Disable MD5 checks

--disable-passwd-check* : Disable passwd/group checks

--scan-knownbad-files* : Perform besides 'known good' check a 'known bad' check

Multiple parameters are allowed

***) Parameter can only be used with other parameters**

Vamos a hacer un escaneo global y ademas le vamos a pedir que nos cree un archivo log que por defecto irá a /var/log/rkhunter.log, pero que nosotros dejaremos en el directorio de trabajo porque asi nos convenga:

localhost ~ # rkhunter -c --createlogfile rkhunter.log

Ante nuestros ojos veremos aparecer todas las comprobaciones de Rkhunter hace y a cada comprobación que hace nos pedirá que pulsemos ENTER para continuar con la siguiente.

Hay que saber también interpretar los resultados ya que por ejemplo si hemos añadido un usuario nuevo al sistema desde la última vez que ejecutamos este escáner, ahora nos dará un aviso ya que encontrará diferencias en los ficheros *passwd* y *group* si también hubieramos creado un grupo propio para ese usuario.

Pues tanto con Rkhunter como con Chkrootkit vamos a poder mantener nuestro sistema en guardia ante programas maliciosos o modificaciones incorrectas, sin duda son el tipo de programa que un usuario novel de Linux quiere para iniciarse en el mundo de la seguridad del propio equipo, sencillos y fáciles de utilizar.

Por VI@d.

Liberada bajo licencia



<http://creativecommons.org/licenses/by-nc-sa/2.5/>