Software Libre y Educación:

servicios de red, gestores de contenidos y seguridad

Seguridad

José Ángel Bernal, Fernando Gordillo, Hugo Santander y Paco Villegas

4 de junio de 2005

Índice general

Capítulo 1

Blindaje del sistema

Intruso (hacker): sustantivo.

1)Se dice de quien goza averiguando los detalles de sistemas de cómputo y cómo llevarlos a su límite, en contraposición a la mayoría de usuarios que prefiere aprender lo mínimo necesario.

2)Se dice de quien escribe programas en forma entusiasta o goza programando en lugar de pensar cómo programar.

Guy L. Steele y cols. *The Hacker's Dictionary*

UNIX no se diseñó para ser seguro. Se diseñó para que se pudiera usar la seguridad

Dennis Ritchie

1.1. Seguridad en UNIX

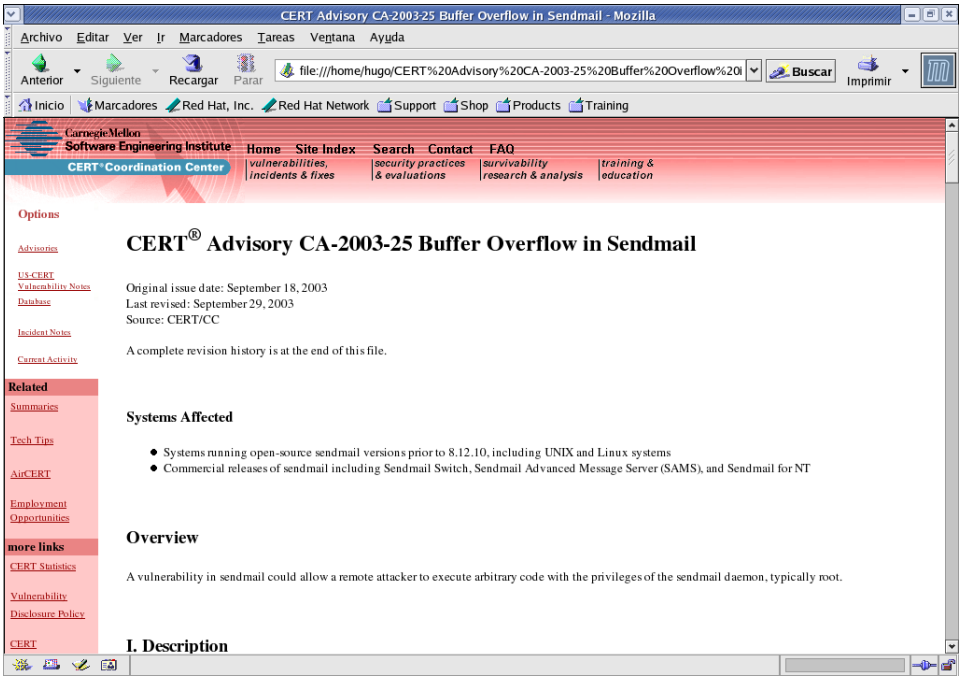
En los años 90 mucha gente pensaba que la seguridad en UNIX era una fantasía. Con la facilidad con que un gurú UNIX podía irrumpir en un sistema y tomar el control del mismo, no concebían la idea de seguridad en estos sistemas.

Desde entonces, las cosas han cambiado, pasando a considerarse los sistemas UNIX como sistemas operativos relativamente seguros. Esto es debido a que en el transcurso de los años se ha avanzado en el estudio de la defensa preventiva de los sistemas UNIX. Cada vez pasa menos tiempo desde que se descubre un agujero de seguridad en algún componente, hasta la aparición del parche o actualización correspondiente.

Sin embargo, por su diseño particular, sigue teniendo fallos. Uno de ellos y quizás el más importante es el *superusuario*, que sigue constituyendo un punto único de ataque. Una vez logrado el acceso a esta cuenta, el atacante tiene control absoluto sobre el sistema.

Fue a raíz del ataque protagonizado por Robert T. Morris en 1988 cuando el tema de la seguridad en sistemas operativos y redes se convirtió en un factor a tener muy en cuenta por cualquier administrador de sistemas. Poco después de este incidente, la agencia DARPA (*Defense Advanced Research Projects Agency*) creó el CERT (*Computer Emergency Response Team*) `http://www.cert.org`, un grupo formado en su mayor parte por voluntarios cualificados de la comunidad informática. El objetivo era facilitar la respuesta rápida ante los problemas de seguridad.

Figura 1.1: Ejemplo de vulnerabilidad aparecida en CERT



Han pasado más de 15 años desde la creación del primer CERT y sigue siendo patente la preocupación por los temas relativos a la seguridad y, sobre todo, se hace patente la necesidad de esa seguridad.

En la actualidad, donde el comercio electrónico y las redes internacionales están a la orden del día, todos los sistemas son potenciales víctimas de un intruso. Prácticamente todos los meses aparece alguna noticia sobre la intrusión en una gran compañia. Estas intrusiones pueden ser totalmente inofensivas y promovidas por la curiosidad o, en el lado opuesto, promovidas por intenciones de lo más siniestras. En este último caso las consecuencias suelen ser desastrosas.

Aunque no se borre o modifique ningún archivo del sistema, es obligación de los administradores una vez detectada la intrusión, chequear el sistema en busca de posibles destrozos o, más importante aún, la colocación de algún programa que realice las funciones de puerta trasera por parte de los intrusos.

Se han desarrollado una gran cantidad de herramientas y técnicas con objeto de ayudar a los administradores de sistemas y servir de medidas preventivas frente a intrusiones.

1.2. Conceptos sobre seguridad

Antes de empezar a hablar de la seguridad de sistemas, es conveniente dejar claro qué se entiende por seguridad. Seguridad es una característica de cualquier sistema informático, que indica que ese sistema está libre de todo peligro de accesos no permitidos que puedan provocar un daño en él. Podría denominarse infalible a un sistema seguro según esta definición.

Particularizando en el terreno que nos ocupa, es muy difícil conseguir esta característica, prácticamente imposible. Se suaviza entonces la definición de seguridad y se habla de fiabilidad, entendiendo como tal la probabilidad de que un sistema se comporte tal y como se espera de él. Más que de sistemas seguros, se habla de sistemas fiables.

Para mantener un sistema seguro o fiable, deben garantizarse tres aspectos:

Confidencialidad. Los objetos de un sistema han de ser accedidos únicamente por los métodos permitidos para ello, y estos métodos no harán disponible esta información a terceros.

Integridad. Los objetos sólo pueden ser modificados por elementos autorizados y de una forma controlada.

Disponibilidad. Los objetos del sistema tienen que permanecer accesibles a elementos autorizados.

Dependiendo del entorno donde se trabaja, puede darse más importancia a alguno de los aspectos anteriores. Por ejemplo, si estamos trabajando en un banco, es prioritario mantener la integridad de los datos, pasando los otros aspectos a un plano inferior.

Una vez claro el concepto de seguridad y lo que conlleva, surge una nueva pregunta ¿qué debe protegerse? Los tres elementos principales a proteger en cualquier sistema informático son:

- Software. Conjunto de programas lógicos que hacen funcionar al hardware instalado, tanto sistemas operativos como aplicaciones.
- Hardware. Conjunto formado por los elementos físicos del sistema informático.
- Datos. Conjunto de información lógica que maneja el hardware y el software.

Una vez claros los conceptos relativos a seguridad de sistemas, es hora de pasar a la acción. Para proteger nuestro sistema la primera tarea a realizar es un análisis de las amenazas potenciales que puede sufrir[1] y, a partir de la información obtenida, diseñar una política de seguridad que defina responsabilidades y reglas a seguir para evitar las amenazas o minimizar su incidencia. A los mecanismos utilizados para la implementación de esta política, se les denomina mecanismos de seguridad. Estos mecanismos serán la parte más visible del sistema de seguridad.

Los mecanismos de seguridad se dividen en tres grupos:

Prevención. Son mecanismos que aumentan la seguridad de un sistema durante el funcionamiento normal de éste, previniendo la ocurrencia de violaciones a la seguridad.

Detección. Son aquellos que se utilizan para detectar violaciones o intentos de violación del sistema.

Recuperación. Son los que se aplican cuando se ha detectado una violación del sistema y quiere devolverse el mismo a un estado estable. Dentro de estos mecanismos está el denominado análisis forense. El objetivo no sólo es retornar a una situación segura y estable, sino también conocer el alcance real de la violación, las actividades llevadas a cabo por el intruso y la forma de entrada.

Debe emplearse el máximo esfuerzo en implementar unos mecanismos de prevención lo suficientemente robustos en nuestro sistema. Se hace patente el dicho "más vale prevenir que curar" ya que es mejor dedicar tiempo a evitar los ataques que a recuperar un sistema que han violado.

1.3. Planificación de Seguridad

La planificación de la seguridad de un sistema puede dividirse en seis etapas diferentes:

1. Planificación de las necesidades de seguridad
2. Análisis de riesgos
3. Análisis de costo-beneficio

[1] Si tenemos un ordenador para uso personal que no tiene acceso a ninguna red, da igual que tenga todas las vulnerabilidades posibles referentes a acceso remoto, no es posible explotarlas. Aunque de qué nos sirve...

4. Creación de políticas de seguridad
5. Implementación
6. Auditoría y respuesta ante incidentes

Un servidor será seguro si se comporta de la manera esperada. Sin embargo, hay que tener en cuenta que pueden considerarse muchos tipos de seguridad:

- Confidencialidad. Proteger la información para que nadie pueda leerla o copiarla sin autorización del dueño.
- Integridad de los datos. Proteger la información (datos y programas) para evitar el borrado o alteración de la misma sin el permiso del dueño.
- Disponibilidad. Proteger los servicios para que no se degraden o dejen de estar disponibles sin autorización.
- Consistencia. Asegurar que el sistema se comporta como esperan los usuarios autorizados.
- Control. Reglamentar el acceso al sistema.
- Auditoría. Registro de las acciones que realizan tanto los usuarios autorizados, como los intentos de acceso de los no autorizados.

1.4. Mecanismos de prevención

Lamentablemente, muchos administradores de equipos UNIX no disponen de los conocimientos o simplemente deciden no dedicar tiempo a la seguridad del sistema del cual son responsables. Esto hace que sean servidores abiertos a cualquier ataque.

Es necesario marcar una serie de pautas o recomendaciones mínimas que ayuden a minimizar este problema al máximo con técnicas de prevención. Estos mecanismos deben ser importantes para cualquier administrador y deben tenerse en cuenta en todas y cada una de las máquinas que administre. Sin embargo, esto no significa que el sistema esté a salvo, son simplemente actuaciones básicas.

1.4.1. Cierre de servicios innecesarios

Cada uno de los servicios ofrecidos en el sistema se convierte en una potencial puerta de entrada a nuestro sistema. Uno de los primeros puntos a comprobar es el superdemonio de red en cualquiera de sus versiones (`xinetd` o `inetd`). Se cerrará cada uno de los servicios que no se utilice o que no se conozca para qué sirve. En el caso de desactivar un servicio que sea necesario, simplemente se activa de nuevo, modificando los ficheros necesarios.

En el caso de `xinetd` se accederá al directorio `/etc/xinetd.d` editando los ficheros que correspondan a servicios que no sean necesarios. Únicamente se establece el atributo `disable=yes` tal como se vio en las primeras entregas del curso.

Para `inetd` únicamente hay que editar el fichero `/etc/inetd.conf`, comentando las entradas correspondientes a servicios que no se vayan a utilizar.

Una vez comprobados los servicios arrancados por `xinetd/inetd` se verán los servicios que se inician al arrancar el servidor. Estos servicios arrancarán procesos independientes que estarán a la escucha de peticiones desde el exterior. La localización de los scripts de arranque es `/etc/rc?.d` o `/etc/rc.d`. Supongamos que queremos desactivar el servicio de correo proporcionado por Sendmail.

```
root@guadalinex:~# find /etc/rc* -name "*sendmail"
/etc/rc0.d/K19sendmail
/etc/rc1.d/K19sendmail
/etc/rc2.d/S21sendmail
/etc/rc3.d/S21sendmail
/etc/rc4.d/S21sendmail
/etc/rc5.d/S21sendmail
/etc/rc6.d/K19sendmail
root@guadalinex:~# find /etc/init.d/ -name "sendmail"
/etc/init.d/sendmail
```

Bastará con renombrar los archivos `S21sendmail` a `noS21sendmail` para que en el próximo arranque del sistema, este servicio no se inicie. Hay que tener en cuenta que el renombrar estos ficheros hace que no se arranque en el próximo inicio de la máquina, pero seguirá activo hasta que se produzca ese evento. Pararemos el servicio con:

```
root@guadalinex:~# /etc/init.d/sendmail stop
```

1.4.2. Instalación de envolventes (*wrappers*)

A pesar de seguir las recomendaciones anteriores y suponiendo que nuestro sistema únicamente tiene activos los servicios necesarios, aún puede hacerse más. El siguiente paso es securizar los servicios que se ha decidido mantener. Es muy conveniente el uso de *wrappers* ya que nos permiten restringir el acceso a los servicios, aceptando únicamente las conexiones que se definan.

Anteriormente se vio uno de los más extendidos, `tcp-wrappers`, por lo que éste es un buen momento para poner en acción los conocimientos adquiridos y empezar a restringir el acceso a los servicios activos. Es conveniente que se configuren `tcp-wrappers` para que únicamente accedan a los servicios las direcciones IP que se indiquen. Dejar que una persona externa a nuestra organización pueda acceder a un servicio es dejarle la puerta abierta para que acceda de forma incontrolada al mismo.

1.4.3. Seguridad de las claves

Claves en Linux

Las claves en Linux no se almacenan en un formato legible. La contraseña se convierte en una cadena de texto mediante un algoritmo criptográfico que proporciona una cadena totalmente distinta de la original. La función que utiliza para realizar esta tarea es `crypt()`. Es una función de un único sentido[2], lo que evita que puede utilizarse para averiguar la clave, cuyo resultado se almacenará en los archivos `/etc/passwd` o `/etc/shadow`.

Cuando un usuario se conecta al sistema con `/bin/login,` lo que se hace realmente es generar de nuevo la clave modificada a partir de la que se introduce por el teclado. La función `crypt()` la transforma y la compara con la que hay almacenada en el sistema. La función `crypt()` se ha demostrado que es suficientemente sólida. Es una mala elección de contraseña lo que provoca los problemas que se verán a continuación.

A pesar de que el código fuente de `crypt()` está disponible, no se ha descubierto ninguna técnica para convertir la contraseña cifrada de nuevo en la contraseña original. Posiblemente, la traducción inversa no sea posible. La única forma conocida de vencer la seguridad de las contraseñas es mediante un ataque de fuerza bruta o mediante un ataque de diccionario.

Elección de las contraseñas

Una contraseña mala es una posible puerta abierta al sistema. Aunque son una de las partes más importantes del sistema, normalmente no se proporciona a los usuarios instrucciones concretas

[2]Puede utilizarse dicha la función, pero no existe la función inversa.

sobre cómo elegir y guardar la contraseña. Todo usuario debe saber que si elige una mala contraseña o se la comunican a alguien que no sea de fiar, comprometen toda la seguridad del sistema.

Una contraseña mala es aquella que se puede adivinar fácilmente. En el mundo real, los intrusos en lugar de comprobar las contraseñas a mano, utilizan sus propios ordenadores para comprobarlas de forma automática. En lugar de probar todas las combinaciones de letras lo que hacen es probar las contraseñas más comunes.

Surge la pregunta ¿cuáles son las malas contraseñas?, veremos algunos ejemplos:

- Nombre de usuario o el de un conocido o familiar
- Nombres escritos al revés, incluso si mezclan mayúsculas y minúsculas
- Contraseñas cortas de cualquier tipo
- Números telefónicos
- Personajes de películas
- Basadas en modificaciones simples de una palabra (sustituir I por 1, E por 3, ...)
- Palabras en otros idiomas

Es recomendable utilizar mayúsculas y minúsculas, además de dígitos y símbolos de puntuación junto con letras. Sin embargo, no debe caerse en la elección de contraseñas difíciles de recordar y que obliguen a escribirlas en algún sitio, no siendo recomendable que superen los 7 u 8 caracteres.

Para elegir una buena contraseña, se pueden tomar dos palabras cortas y combinarlas intercalando un carácter especial o un número. Otra opción puede ser componerla como un acrónimo de una frase o poema que nos guste, por ejemplo, “Esta contraseña es lo suficientemente segura” generaría la clave E2c0E0l4Ss. Hemos tomado las primeras letras alternando mayúsculas y minúsculas e intercalando el número 2004[3].

En caso que el usuario maneje varias cuentas a la vez en el mismo o distintos sistemas, sería un error utilizar la misma contraseña en todos los sistemas, ya que si averiguan la contraseña de una de las cuentas, la seguridad del resto de sistemas y cuentas se verá comprometida. Un enfoque muy válido es utilizar una contraseña básica y modificarla en función de la cuenta o del sistema en el que estemos accediendo.

Hay una película bastante conocida llamada “Juegos de Guerra” en la que un joven realizaba sus primeros pinitos en la infiltración clandestina de sistemas. Accedía al ordenador central de su instituto para cambiarse las calificaciones, gracias al listado de claves que existía en un papel en la secretaría del centro. Lamentablemente, esto sucede cientos de veces. Una advertencia básica es que los usuarios no anoten la contraseña nunca. Aún así, si un usuario quiere escribir su contraseña en algún sitio debe seguir las siguientes recomendaciones:

- Al escribirla, no identificarla como contraseña
- No incluir el nombre de usuario ni los datos del servidor al que se accede
- Guardar en lugar seguro

1.4.4. Seguridad de los usuarios

Usuarios normales

Se asumirá como práctica de seguridad habitual, en lo referente a usuarios, el uso de las utilidades de clave `shadow`. Los ficheros que se utilizan para la gestión de los usuarios son:

[3]Se aconseja encarecidamente no utilizar este ejemplo concreto en nuestro sistema ya que cualquiera que lea esta documentación conocerá la clave.

```
/etc/passwd
/etc/shadow
/etc/group
/etc/gshadow
```

Para prevenir un borrado o sobreescritura accidental de estos ficheros, es recomendable activar el bit inmutable. Es una forma de protección que también previene de la creación de enlaces simbólicos sobre estos ficheros con objeto de realizar posteriormente un ataque.

```
[root@fedora root]# lsattr /etc/passwd
----------------- /etc/passwd
[root@fedora root]# chattr +i /etc/passwd
[root@fedora root]# chattr +i /etc/shadow
[root@fedora root]# chattr +i /etc/group
[root@fedora root]# chattr +i /etc/gshadow
```

Hay que tener en cuenta que una vez establecido este bit cualquier modificación sobre estos ficheros dará error. De esta forma, para añadir un nuevo usuario es necesario deshabilitar el bit de nuevo. Lo mismo ocurrirá con cualquier paquete de software que durante el proceso de instalación requiere la creación de nuevos usuarios o grupos.

```
[root@fedora root]# lsattr /etc/passwd
---i------------- /etc/passwd
[root@fedora root]# adduser gandalf
adduser: unable to open password file
```

Un punto fundamental en la seguridad de los accesos a un sistema, es el referido a las claves de usuarios. Muchos usuarios piensan que sus ficheros y programas están lo suficientemente protegidos por la elección de una "buena" clave, pero esto no es así. No existen claves irrompibles. Si se da suficiente tiempo y recursos a un intruso, tarde o temprano conseguirá averiguar la clave de un usuario, ya sea por fuerza bruta o por ingeniería social.

En primer lugar, la clave que se elige debe ser segura y no adivinable fácilmente. Para ello las siguientes recomendaciones pueden ser de utilidad:

- La longitud de la clave debe establecerse entre 5 y 8 caracteres, incluyendo algún número o carácter especial en la misma.
- No debe ser trivial (basada en el nombre, familia, lugar de trabajo, ...).
- Debe ser revocada y reseteada después de un número determinado de intentos.

El fichero de sistema `/etc/login.defs` es el fichero de configuración para la utilidad `login`. Es necesario revisarlo una vez que se ha instalado el sistema para revisar los valores que tiene configurados por defecto y establecerlos de acuerdo a las consideraciones de seguridad que se hayan adoptado.

Es recomendable cambiar los parámetros por defecto que se utilizan en la creación de nuevos usuarios. Estos parámetros afectan a la caducidad de las claves y a los grupos a los que pertenecen los nuevos usuarios cuando son creados entre otros[4].

4

Debian:

- Los campos del fichero `/etc/login.defs` cambian los valores por defecto.
- Si bien se puede crear el fichero `/etc/default/useradd`, es mejor usar el comando `adduser`. Su fichero de configuración es `/etc/adduser.conf` y está muy bien documentado.

Fichero configuración	Campo	Valor por defecto (Fedora)	Descripción
/etc/login.defs	PASS_MAX_DAYS	99999	Máximo número de días que una clave es válida
/etc/login.defs	PASS_MIN_DAYS	0	Máximo número de días permitido entre el cambio de clave
/etc/login.defs	PASS_WARN_AGE	7	Número máximo de días antes de forzar un cambio de clave
/etc/login.defs	UID_MIN	500	Mínimo valor para el UID automático
/etc/login.defs	GID_MIN	500	Mínimo valor para el GID automático
/etc/login.defs	PASS_MIN_LEN	5	Longitud máxima de clave aceptable (El módulo `pam_cracklib` sobreescribe este valor con el parámetro `minlen`)
/etc/default/useradd	GROUP	100	Grupo por defecto
/etc/default/useradd	HOME	/home	Directorio local de usuario
/etc/default/useradd	INACTIVE	-1	Máximo número de días después de la expiración de la clave en que el usuario puede cambiar la clave
/etc/default/useradd	EXPIRE	n/a	Fecha de expiración de una cuenta en el formato AAAA-MM-DD
/etc/default/useradd	SHELL	/bin/bash	Shell por defecto
/etc/default/useradd	SKEL	/etc/skel	Directorio de perfil por defecto

Otra consideración a tener en cuenta es la variable de entorno `$PATH`. Cuando un usuario ejecuta un comando, el *shell* buscará en cada uno de los directorios existentes en el *path,* hasta encontrar un comando con el mismo nombre que el tecleado, en cuyo caso lo ejecuta sin más. En caso de no encontrarlo dará un mensaje de error.

¿Qué problemas de seguridad presenta esta variable? Es muy recomendable comprobar que ninguno de los directorios que aparecen en `$PATH` del superusuario tienen permiso de escritura para los usuarios normales. Esto incluye a directorios como `/tmp/` o "`.`".

Imaginemos la siguiente situación. El usuario `root` de nuestro sistema tiene incluído en su variable `$PATH` el directorio actual como uno más donde buscar ejecutables. Si este usuario desea comprobar el contenido del directorio `/tmp/` o el de `$HOME` de alguno de sus usuarios, seguramente cambiará su directorio actual al del usuario en cuestión. ¿Qué sucede si "`.`" está en primer lugar en la variable `$PATH`? El *shell* buscará primero en el directorio actual (recordemos que es el del usuario) y podríamos encontrarnos con:

```
[root@fedora root]# cd /home/hugo/
[root@fedora hugo]# cat ls
#!/bin/bash
cd /
rm -rf *
```

Si tecleamos en la línea de comandos `ls,` se ejecutará el *script* anterior, borrando todo el sistema. Un simple `ls` habría conseguido que se borrase parte del sistema o todo, simplemente porque el administrador no ha tenido la precaución de configurar convenientemente la variable `$PATH`.

Surge entonces la pregunta ¿si ponemos el directorio "." al final de `$PATH` se soluciona el problema? La respuesta es contundente, NO, el problema sigue existiendo. Supongamos ahora que el script se llama *`moer`*. No es un comando que exista, pero ¿cuántas veces nos hemos equivocado al escribir el comando `more` y hemos tecleado `moer`? Si tecleamos `moer`, el shell buscará sin encontrarlo hasta llegar al ultimo directorio de `$PATH` y ejecutará el script. Tenemos el mismo resultado, borrado total o parcial del sistema. Parece claro, tras estos ejemplos que no es una buena práctica poner el directorio "." en la variable de entorno `$PATH` del superusuario.

Usuario root

La cuenta de usuario `root` es la que tiene más privilegios en un sistema UNIX. Este usuario no tiene restricciones de seguridad por lo que hay que operar con esta cuenta con mucha precaución.

Es importante no dejarse nunca esta cuenta abierta en la consola. En caso de que esto ocurra, puede establecerse un tiempo de desconexión del sistema si no se registra ninguna actividad desde esta cuenta. Así, puede establecerse que a los 3 minutos desde la última operación hecha desde una sesión del usuario `root` se desconecte dicha sesión[5].

Como medida de seguridad adicional también puede restringirse el uso del comando `su` para acceder a la cuenta `root`. Puede configurarse el sistema para que sólo los usuarios de un determinado grupo accedan a la cuenta de `root` mediante `su`. El fichero de configuración implicado es `/etc/pam.d/su`

```
auth sufficient /lib/security/pam_rootok.so
auth required /lib/security/pam_stack.so service=system-auth
account required /lib/security/pam_stack.so service=system-auth
password required /lib/security/pam_stack.so service=system-auth
session required /lib/security/pam_stack.so service=system-auth
session optional /lib/security/pam_xauth.so
```

Las líneas que es necesario añadir para reflejar esta configuración dejan el fichero `/etc/pam.d/su` como puede verse a continuación:

```
#Uncomment the following line to implicitly trust users in the 'wheel' group
    .
#auth sufficient /lib/security/pam_wheel.so trust use_uid
#Uncomment the following line to require a user to be in the 'wheel' group.
#auth required /lib/security/pam_wheel.so use_uid
auth sufficient /lib/security/pam_rootok.so
auth sufficient /lib/security/pam_wheel.so trust use_uid
auth required /lib/security/pam_wheel.so use_uid
auth required /lib/security/pam_stack.so service=system-auth
account required /lib/security/pam_stack.so service=system-auth
password required /lib/security/pam_stack.so service=system-auth
session required /lib/security/pam_stack.so service=system-auth
session optional /lib/security/pam_xauth.so
```

Usuarios especiales

Es importante deshabilitar todas las cuentas creadas por defecto en la instalación por los desarrolladores de algunos productos, si no se van a usar en el sistema. Algunas de estas cuentas existen por defecto, incluso si no se han instalado los programas o utilidades asociados. Cuantas más cuentas tenga un sistema más posibilidades hay de encontrar un acceso al mismo a través de una de estas cuentas.

[5] Por ejemplo, añadiendo dentro del fichero `/etc/profile` una variable `TMOUT` a la que se le da un valor en segundos que indica el tiempo de desconexión por inactividad. Podemos incluirla tambien en `.bashrc` de los usuarios para controlar el tiempo de inactividad.

El mismo procedimiento se seguirá con los grupos genéricos instalados en el sistema por defecto. En definitiva, todo aquello que no se use debe ser desinstalado. Esto es una norma general de seguridad del sistema que debe llevarse a cabo siempre.

1.5. SELinux

1.5.1. ¿Qué es SELinux?

A continuación, se explica brevemente en qué consiste SELinux. Sólo queremos que tengáis una pequeña noción de este nuevo Módulo de Seguridad que con el tiempo irá teniendo un mayor peso en las nuevas distribuciones. No entraremos en instalación ni configuración y solo mostraremos algún pequeño ejemplo que ayude a comprender los conceptos básicos, quien esté más interesado en los enlaces de la NSA y Fedora puede encontrar casi toda la información disponible.

Security Enhanced Linux, SELinux, es un sistema que modifica el núcleo de Linux, fortaleciendo los mecanismos de control de acceso y forzando la ejecución de los procesos dentro de un entorno con los mínimos privilegios necesarios.

La primera versión de SELinux se remonta a finales del año 2000 de manos de la NSA (Agencia Nacional de Seguridad de los Estados Unidos). El objetivo del mismo es, por un lado, demostrar la posibilidad de implementar el modelo de seguridad de control de acceso obligatorio y el control de acceso basado en roles en entorno Linux. Como segundo objetivo, hacer frente a la eventualidad de que los sistemas operativos "*trusted*" (confiables y seguros) comerciales dejaran de estar disponibles.

SELinux puede considerarse como una implementación práctica del modelo de seguridad de control de acceso obligatorio basado en el núcleo del sistema operativo Linux. Un administrador de un sistema SELinux tiene la posibilidad de configurar una política donde se definen los archivos a los que tiene acceso cada programa.

Para poder realizar esto, SELinux implementa un mecanismo para establecer en cada archivo y proceso el contexto en el que está siendo utilizado.

Mediante la utilización de un módulo del sistema operativo, es posible establecer reglas para permitir o denegar el acceso a cualquier archivo del sistema (utilizando el concepto de archivo de Unix, lo que incluye a los dispositivos, ficheros...).

El sistema operativo dispone de un proceso servidor de seguridad, que se ejecuta como parte del núcleo, que decide en base a la política de seguridad definida por el administrador, si algo (un proceso o un usuario) dispone de permiso para acceder a un objeto (archivo, dispositivo...). Este mecanismo de control se denomina *Type Enforcement* (TE).

Así, ante una incidencia de seguridad como puede ser un desbordamiento de búfer en un proceso ejecutado por `root`, el atacante sólo podrá acceder a los archivos para los cuales el proceso vulnerable esté autorizado por la política del sistema. No tendrá ningún efecto sobre el resto de archivos u objetos del sistema.

SELinux también permite implementar un modelo adicional de seguridad (*Multi-Level Security*, MLS) en el que además de lo indicado hasta ahora, es posible, para cada objeto, una capa de seguridad (como "altamente secreta", "secreta", "confidencial" y "sin restricción"). En este modelo, a los mecanismos descritos anteriormente se añade la restricción de que únicamente aquellos procesos y usuarios situados en la misma capa (o una capa superior) pueden acceder a los objetos de la misma capa o inferiores, pero nunca al revés. Así un usuario o un proceso de la capa "confidencial" puede acceder a la información "confidencial" y "sin restricción", pero nunca a la información marcada como "secreta" o "altamente secreta".

Puede encontrarse más información en:

`http://www.nsa.gov/selinux/index.cfm`

`http://fedora.redhat.com/projects/selinux/`

`http://fedora.redhat.com/docs/selinux-faq-fc3/`

1.5.2. Terminología SELinux

Pasamos ahora a introducir algunos de los conceptos básicos de SELinux. Se busca facilitar al alumno la comprensión posterior de documentos más detallados.

Como ya se ha comentado, mediante SELinux se definen una serie de políticas que controlan el acceso de forma que los usuarios únicamente obtienen los privilegios necesarios para realizar su trabajo. Así, es posible reducir o eliminar los daños producidos por posibles errores en la configuración o *buffer overflows*. Este mecanismo opera de forma independiente al tradicional mecanismo de control de acceso de Linux. No existe el concepto de superusuario o `root` ni los binarios setuid/setgid.

En el caso que la distribución de linux no tenga el soporte SELinux instalado es necesario instalar un kernel modificado que incluya estas funcionalidades.

SELinux proporciona compatibilidad con las aplicaciones linux existentes y con los módulos del kernel. Sin embargo, algunos módulos del kernel puede que requieran su modificación para interactuar de forma adecuada con SELinux. Las categorías de compatibilidad son:

- Compatibilidad de aplicación. SELinux proporciona compatibilidad con las aplicaciones existentes. No se han cambiado las estructuras de datos visibles por las aplicaciones ni el interfaz de las llamadas al sistema existentes, por lo que las aplicaciones correrán sin cambios si la política de seguridad autoriza la operación.
- Compatibilidad de módulos del kernel. Originalmente, SELinux solo proporciona compatibilidad para los módulos del kernel existentes, por lo que era necesario recompilar los módulos con las estructuras de seguridad necesarias. Ahora están integrados en el kernel 2.6, proporcionando compatibilidad binaria con los módulos existentes, con algunas excepciones que precisan modificaciones en los módulos.

SELinux utiliza un sistema de ficheros especial como parte de la instalación. Será necesario modificar `/etc/fstab` para que se monte de forma correcta.

```
none /selinux selinuxfs noauto 0 0
```

El sistema de ficheros `/selinux` es similar a `/proc`, es un pseudo sistema de archivos.

```
ls -l /selinux
total 0
-rw-rw-rw- 1 root root 0 Nov 25 11:27 access
-rw-rw-rw- 1 root root 0 Nov 25 11:27 context
-rw-rw-rw- 1 root root 0 Nov 25 11:27 create
-rw------- 1 root root 0 Nov 25 14:19 enforce
-rw------- 1 root root 0 Nov 25 11:27 load
-r--r--r-- 1 root root 0 Nov 25 11:27 policyvers
-rw-rw-rw- 1 root root 0 Nov 25 11:27 relabel
-rw-rw-rw- 1 root root 0 Nov 25 11:27 user
```

Si se ejecuta `cat /selinux/enforce` se obtendrá 1 ó 0 dependiendo de si estamos en modo forzado o permisivo respectivamente. Posteriormente se indicará que implica cada uno de estos modos.

En Debian y derivados, el directorio de configuración de políticas estará en `/etc/selinux`. En el caso de Fedora es `/etc/security/selinux/src/policy`.

A continuación veremos los conceptos con los que trabaja SELinux.

Identidad

Una identidad bajo SELinux no tiene el mismo significado que un uid. Ambos pueden convivir en el mismo sistema pero son diferentes. Las identidades para SELinux forman parte de un contexto de seguridad que definirá a qué dominios se puede acceder (basicamente, qué se puede hacer). Si se ejecuta el comando `su` no se cambiaría la identidad bajo SELinux.

Dominio

Cada proceso se ejecuta en un dominio. Un dominio determina el acceso que tiene un proceso, es una lista de qué procesos puede hacer o qué acciones puede realizar un proceso. En este caso sí hay similitud con el uid. Supongamos que `root` tiene un programa al que le ejecuta `chmod 4777` haciéndolo setuid `root`. Cualquiera en el sistema puede ejecutar este programa con privilegios de `root`, lo que representa un claro agujero de seguridad. Con SELinux si se tiene un proceso que hace una transición a un dominio privilegiado, si el *role* del proceso no lo autoriza a entrar entonces no se ejecutará.

Ejemplos de dominios son `sysadm_t` que es el dominio de administración del sistema y `user_t` que es el dominio general sin privilegios.

Tipo

Un tipo es asignado a un objeto y determina quién tiene acceso a este objeto. La definición para dominio es similar, excepto que un dominio se aplica a un proceso y el tipo se aplica a objetos como pueden ser directorios, ficheros, sockets, etc.

Role

Un role determina qué dominios pueden ser usados. Los dominios que un role de usuario puede acceder están predefinidos en los ficheros de configuración de políticas. Si un role no está autorizado para entrar en un dominio, se le negará el acceso.

Así, para permitir a un usuario del dominio sin privilegios `user_t` ejecutar el comando `passwd` será necesario especificarlo:

```
role user_r types user_passwd_t
```

De esta forma un usuario en el role `user_r` se le permite entrar en el dominio `user_passwd_t` donde puede ejecutar el comando `passwd`.

Contexto de seguridad

Un contexto de seguridad tiene todos los atributos que están asociados a cosas como ficheros, directorios, procesos, sockets TCP, entre otros. Un contexto de seguridad está formado por la identidad, role y dominio o tipo. Para verificar el contexto de seguridad actual se ejecuta `id` bajo SELinux.

En el caso de crear un fichero, el contexto de seguridad variará dependiendo del dominio que lo cree. Por defecto, el nuevo fichero hereda el mismo tipo que el directorio padre, aunque este comportamiento puede cambiarse con las políticas.

Si el usuario `legolas` crea un fichero con el nombre `prueba` en su directorio `$HOME`:

```
ls --context prueba
-rw-r--r-- legolas legolas legolas:object_r:user_home_t prueba
```

Si a continuación crea un fichero en `/tmp` llamado `tmpprueba`:

```
ls --context /tmp/tmpprueba
-rw-r--r-- legolas legolas legolas:object_r:user_tmp_t /tmp/tmpprueba
```

El tipo ha cambiado dependiendo del directorio donde fue creado el fichero.

Una forma de cambiar el contexto de seguridad es usando el comando `newrole -r role`, donde `role` es el nuevo role que quiere adoptarse. De esta forma, si un usuario quiere adoptar el role `sysadm_r`:

```
newrole -r sysadm_r
```

Será necesario proporcionar la clave para la identidad del usuario, la cual puede chequearse con el comando `id`. En caso de no tener autorización para entrar en el nuevo role:

```
newrole -r sysadm_r
legolas:sysadm_r:sysadm_t is not a valid context
```

Con este mensaje se indica que el usuario `legolas` no puede entrar en el role:dominio `sysadm_r:sysadm_t` debido a que no está autorizado.

Transición

Una transición determina qué contexto de seguridad será asignado a la operación solicitada. Hay dos tipos de transición:

- Transición del dominio de un proceso que es usado, cuando se ejecuta un proceso de un tipo especificado.
- Transición de un tipo de fichero, cuando se crea un fichero bajo un directorio en particular.

Una transición de tipo es lo que se vio en el ejemplo anterior.

Políticas

Las políticas están constituidas por un conjunto de reglas que definen cosas como los roles a los que un usuario tiene acceso. Estas reglas se editan conforme a como se desee que se defina la seguridad del sistema.

1.5.3. Modos de uso de SELinux

El **modo Permisivo** es el utilizado cuando el servidor se dedica a guardar información en ficheros de log referente a SELinux. No se siguen las reglas definidas en las políticas, simplemente se almacenan los eventos que se producen relativos a SELinux. Este modo es el adecuado para depuración ya que se pueden repasar los mensajes generados y verificar que la configuración es correcta.

El otro modo existente es el **modo Reforzado**. En este modo el sistema sigue las políticas que se hayan definido en SELinux. Hay que tener cuidado cuando se active este modo, si hay algún fallo en la configuración de las políticas puede perserse parte del acceso al sistema.

Para poder pasar de un modo a otro es necesario que esté definida la opción `CONFIG_SECURITY_SELINUX_DEVELOP` en la compilación del kernel. Posteriormente, para pasar del modo permisivo al reforzado es necesario ejecutar:

```
echo "1" > /etc/selinux/enforce
```

En caso contrario se sustituye `1` con `0`. Esto proporciona un método para saber en que modo está el sistema. Basta con verificar el valor de `/etc/selinux/enforce`.

Si se compila el kernel con el modo desarrollo activado el servidor arrancará con el modo permisivo. Este comportamiento puede modificarse creando un script que se ejecute en el arranque y que cambie de modo pasando el parámetro `enforcing=1` al kernel durante el arranque.

Capítulo 2

Vulnerabilidades del sistema

> Los favorables al Open Source defienden que la naturaleza del software de fuentes abiertas lo hace más seguro. Los críticos al movimiento Open Source defienden que el software abierto es menos seguro.
>
> *Hacking Exposed Linux*

2.1. Tipos de ataques y vulnerabilidades

2.1.1. Ataques de negación de servicio (*denial of service*)

El ataque denominado *Denial of Service*[1] o de negación de servicio es un tipo de ataque cuya meta fundamental es la de negar el acceso del atacado a un recurso determinado o a sus propios recursos. Los ataques de negación de servicio pueden dejar inoperativo un servidor o una red. De esta forma, toda una organización puede quedar fuera de Internet durante un tiempo determinado. Algunos ejemplos de este tipo de ataque son:

- Intentos de inundar (*flood*) una red, evitando de esta manera el tráfico de datos en la misma.
- Intentos de interrumpir las conexiones entre dos máquinas, para evitar el acceso a un servicio por parte del resto de usuarios.
- Intentos de evitar que un determinado usuario tenga acceso a un servicio.
- Intentos de interrumpir un servicio específico a un sistema o a un usuario.

También hay que tener en cuenta que el uso ilegítimo de recursos puede dar lugar, igualmente, a la negación de un servicio. Por ejemplo, un intruso puede utilizar un área del FTP anónimo como lugar para salvar archivos, consumiendo, de esta manera, espacio en el disco y generando tráfico en la red.

Algunos ataques de negación de servicio se pueden ejecutar con recursos muy limitados contra un sitio grande y sofisticado. Este tipo de ataque se denomina ataque asimétrico. Por ejemplo, un atacante con un ordenador anticuado y un módem puede poner fuera de combate a máquinas rápidas y sofisticadas.

Hay tres tipos de ataques básicos de negación de servicio:

- Consumo de recursos escasos, limitados, o no renovables
- Destrucción o alteración de información de configuración
- Destrucción o alteración física de los componentes de la red

[1] También se hace referencia a él como ataque DoS.

Ataques que podemos sufrir

Los ordenadores y las redes necesitan para funcionar ciertos recursos: ancho de banda de la red, espacio de memoria y disco, tiempo de CPU, estructuras de datos, acceso a otros ordenadores y redes, entre otros.

Los ataques de negación de servicio se ejecutan, con frecuencia, contra la conectividad de la red. La meta del atacante es evitar que los ordenadores se comuniquen en la red. La inundación o *flooding* es uno de las formas más antiguas de ataques DoS a internet. En la actualidad el ancho de banda de la red es tan importante como cualquiera de los elementos físicos del servidor. Los ataques DoS de inundación de red pretenden evitar que los ordenadores que forman nuestra red puedan comunicarse. El ataque comienza con un proceso de establecimiento de conexión con la máquina objetivo, uno de los ejemplos más claros es el ataque `SYN` *flood*. Veamos con un poco de detalle en qué consiste este ataque.

Cuando un sistema (cliente) intenta establecer una conexión TCP con el sistema que proporciona servicios (servidor), tanto el cliente como el servidor van a intercambiar un conjunto de mensajes. Esta técnica de conexión se aplica a todas las conexiones TCP. El cliente comienza enviando un mensaje `SYN` al servidor. A su recepción, el servidor da su reconocimiento al mensaje enviando un mensaje `SYN-ACK`. El cliente finaliza la conexión respondiendo con un mensaje `ACK`. Es en este momento en el que se establece la comunicación entre cliente y servidor[2].

El potencial ataque tiene lugar en este punto donde el servidor ha enviado un reconocimiento `SYN-ACK` de vuelta al cliente, pero aún no ha recibido el `ACK`. El servidor ha construido en memoria una estructura para almacenar todas las conexiones pendientes. Esta estructura tiene un tamaño limitado y puede excederse creando muchas conexiones como la descrita. El atacante envía muchos mensajes `SYN` haciéndose pasar por clientes desconocidos que no van a responder. Logrará de esta manera que el servidor no acepte más conexiones entrantes cuando sature la estructura de datos donde almacena las comunicaciones pendientes de confirmación.

Debemos tener en cuenta que este tipo de ataque no depende del ancho de banda que disponga el atacante. En este caso, el atacante está consumiendo las estructuras de datos del kernel, implicadas en establecer una conexión TCP. Un atacante con una simple conexión telefónica puede realizar este ataque contra un servidor de los más potentes (éste último es un buen ejemplo de un ataque asimétrico).

Un atacante también puede utilizar los recursos que disponemos contra nosotros mismos, de maneras inesperadas. Por ejemplo, el caso de DoS UDP. En este ataque, el atacante utiliza los paquetes falsificados de UDP para conectar el servicio de generación de *echo* en una máquina con el servicio de *chargen* en otra máquina. El resultado es, que los dos servicios consumen todo el ancho de banda de red entre ellos. Así, la conectividad para todas las máquinas en la misma red desde cualquiera de las máquinas atacadas se ve afectada.

Un atacante puede, también, consumir todo el ancho de banda disponible en su red generando una gran cantidad de paquetes dirigidos a la misma. Típicamente, estos paquetes son de generación de *echo* de `ICMP` (`ping`), pero pueden ser cualquier otra cosa. Además, el atacante no necesita operar desde una sola máquina; podría coordinar varias máquinas en diversas redes para alcanzar el mismo efecto.

Además del ancho de banda de la red, los atacantes pueden consumir otros recursos que nuestro sistema necesite para funcionar. Por ejemplo, en muchos sistemas, un número limitado de las estructuras de datos en el kernel está disponible para almacenar información de procesos (identificadores, entradas en tablas de procesos, *slots* , etc.). Si se consumen estas estructuras de datos escribiendo un programa o un script que no haga nada, pero que cree en varias ocasiones copias de sí mismo conseguiríamos un ataque DoS. Muchos sistemas operativos modernos, aunque no la totalidad de ellos, tienen recursos para protegerse contra este problema.

Un atacante puede también consumir su espacio en disco de otras maneras, por ejemplo:

- Generar miles de correos (*Spam*, *Bombing*).

[2]Lo que conocemos como *Three way handshake* o establecimiento de conexión en tres pasos.

- Generar intencionalmente errores que deben ser reflejados en los ficheros de log. En este tipo de ataque, podemos citar también la utilización indebida de `syslog`. Es decir, utilizar el proceso `syslog` de la víctima para que registre eventos de otra máquina, llenando el espacio en disco.
- Colocar archivos en su disco, utilizando `ftp` anónimo.

En general, se puede utilizar cualquier cosa que permita que los datos sean escritos en el disco para ejecutar un ataque DoS si no hay límites en la cantidad de datos que se pueden escribir (*quotas*).

No obstante, muchos sitios tienen esquemas de *lockout* de cuenta después de un cierto número de *logins* fallados. Una configuración típica bloquea el *login* después de 3 a 5 intentos fallidos. Un atacante puede utilizar este esquema para evitar que los usuarios legítimos entren. En algunos casos, incluso las cuentas privilegiadas, tales como `root`, pueden ser víctimas de este tipo de ataque. Es fundamental disponer siempre de un método para acceder ante la emergencia de este tipo de ataques.

Algunas veces, errores en la programación del kernel de Linux pueden llevarnos a un DoS:

- Ping de la muerte (*ping of death*). Algún software permite enviar paquetes ICMP que son mayores de 65.536 bytes, valor máximo que la especificación TCP/IP permite. En la actualidad, sólo algunas pilas TCP/IP son vulnerables a este ataque y la mayoría de los routers de internet filtrarán los paquetes de este tamaño.
- *Teardrop*. Es similar al ping de la muerte, pero en este caso intenta romper la pila de la red destino proporcionándole múltiples fragmentos que no se reensamblan de forma adecuada. El resultado es un *kernel panic* y el consiguiente reinicio de la máquina.
- *Deep simlink bug*. En algunas versiones del kernel, incluyendo hasta la 2.4.9 hay un error que permite realizar un DoS local. Un atacante puede crear un directorio que contenga múltiples directorios y enlaces simbólicos que referencien a ellos mismos de forma repetida. Cuando uno de los ficheros es leído, el kernel consume un periodo largo de tiempo intentando resolver dónde se encuentra el fichero original. Durante dicho periodo ningún otro proceso puede ejecutarse y la máquina se bloquea.

Hay otros componentes que pueden ser vulnerables a la negación de servicio y que deben vigilarse. Estos incluyen:

- Impresoras
- Unidades de cinta
- Conexiones de red
- Otros recursos limitados importantes para la operación del sistema.

Un ordenador incorrectamente configurado puede no funcionar bien, o directamente no arrancar. Un atacante puede alterar o destruir la información de configuración del sistema operativo, evitando de esta forma que pueda usarse el ordenador atacado. Veamos algunos ejemplos:

- Si un atacante puede cambiar la información de enrutado de sus *routers*, la red puede ser deshabilitada.
- Si un atacante puede modificar cualquier fichero de configuración del sistema, de los existentes en `/etc` el sistema puede no arrancar. Supongamos que se borran las entradas de `/etc/fstab`.

También es muy importante la seguridad física de la red. Se debe resguardar contra el acceso no autorizado a los ordenadores, los *routers*, los *racks* de cableado de red, los segmentos del *backbone* de la red, y cualquier otro componente crítico de la red.

Prevención y respuesta a los ataques

Tal como se ha expresado anteriormente, los ataques DoS pueden dar lugar a pérdidas significativas de tiempo (y dinero) para muchas organizaciones, por lo que se recomiendan una serie de medidas:

- Colocar listas de control de acceso en los *routers*. Esto reducirá su exposición a ciertos ataques DoS
- Instalar parches a su sistema operativo contra *flooding* de `TCP SYN`. Esta acción permitirá reducir sustancialmente la exposición a estos ataques, aunque no pueda eliminar el riesgo de forma definitiva.
- Invalidar cualquier servicio de red innecesario o no utilizado. Esto puede limitar la capacidad de un atacante de aprovecharse de esos servicios para ejecutar un ataque DoS.
- Implementar sistema de cuotas.
- Observar el funcionamiento del sistema y establecer valores base para la actividad ordinaria. Utilizar estos valores para calibrar niveles inusuales de la actividad del disco, del uso de la CPU, o del tráfico de red.
- Incluir, como parte de su rutina, el examen de su seguridad física. Considerar, entre otras cosas, los servidores, *routers*, terminales desatendidos, puertos de acceso de red y los *racks* de cableado.
- Utilizar Tripwire o una herramienta similar para detectar cambios en la información de configuración u otros archivos.
- Utilizar configuraciones de red redundantes y tolerantes a fallos.

2.1.2. Cracking de passwords

En este apartado, se presentarán una serie de consideraciones referidas al *cracking* de *passwords* basadas en UNIX. El objetivo inicial consiste en entrar al servidor. Debido a que se permite el acceso a múltiples usuarios, los sistemas UNIX nos solicitarán un nombre de identificación acompañado de una clave. Dicho nombre deberá darse de alta en el sistema para que se pueda acceder.

Cuando un usuario desea entrar en una máquina, el sistema solicitará un *login* de acceso o nombre de usuario. Si el *login* es incorrecto, el sistema no lo notificará para impedirle conocer qué accesos se encuentran dados de alta. Si la *password* coincide con la que tiene asignada el *login* que se emplea, el sistema permitirá el acceso.

Una vez encriptada una password, no se puede desencriptar. Sin embargo, esto no garantiza la seguridad de la *password*, puesto que no significa que la *password* no se pueda averiguar.

El mecanismo que se utiliza para descubrir (no desencriptar) las *passwords* consiste en efectuar encriptaciones de palabras (posibles *passwords*) y comparar estas encriptaciones con el original.

¿De que depende el éxito? El éxito depende de la calidad del diccionario (archivo que contiene un conjunto de posibles *passwords*), del programa que se utilice, de la CPU y, por supuesto, de nuestra paciencia. Los programas buscadores de contraseñas son fácilmente diseñables. Si mediante un *bug* se obtiene el archivo `/etc/passwd`, se puede iniciar un ataque de diccionario contra el mismo obteniéndose, de este modo, las *passwords*.

Otro tipo de ataque es el de fuerza bruta, que consiste simplemente en realizar todas la combinaciones posibles de caracteres hasta hallar la password.

2.1.3. E-mail bombing y spamming

En este apartado, se presentarán algunas de las dificultades que pueden surgir como consecuencia de la utilización de los servicios de *mail*. Se brindarán, por otro lado, algunas respuestas a dichos obstáculos.

El *e-mail bombing* consiste en enviar muchas veces un mensaje idéntico a una misma dirección, saturando el *mailbox* del destinatario. El *spamming*, que es una variante del *e-mail bombing*, se refiere a enviar el *email* a centenares o millares de usuarios e, inclusive, a listas de interés. El *spamming* puede resultar aún más perjudicial si los destinatarios contestan el *mail*, haciendo que todos reciban la respuesta.

Puede, además, ocurrir inocentemente como resultado de enviar un mensaje a la lista y no darse cuenta de que la lista lo distribuye a millares de usuarios, o como resultado de mala configuración de un autorespondedor, por ejemplo el vacation[3].

El *e-mail bombing/spamming* se puede combinar con el *e-mail spoofing*[4], logrando que sea más difícil determinar quién está enviando realmente el *mail*.

Cuando se proveen los servicios de *e-mail* los usuarios son, lógicamente, vulnerables al *e-mail bombing* y *spamming*. En efecto, el *e-mail spamming* es casi imposible de prevenir. Un usuario con una dirección válida de *mail* puede realizar *spam* a cualquier otra dirección de *mail* o *newsgroup*.

Cuando gran cantidad de *mails* son dirigidos a un solo sitio, éste puede sufrir DoS por pérdida de conectividad, caerse el sistema o producirse fallos en el servicio debido a:

- sobrecarga de conexiones de red
- utilización de todos los recursos de sistema disponibles
- llenado del disco como resultado de *postings* múltiples y de entradas en el *syslog*.

Si el sistema aparece repentinamente lento (el *e-mail* está lento o no parece que se envía o recibe), la razón puede ser que el servidor de correo está intentando procesar una excesiva cantidad de mensajes. Esto puede comprobarse a través del log de sistema.

Es importante:

- Identificar la fuente del *e-mail bomb/spam* y configurar el *router* para evitar el acceso de los paquetes entrantes de esa dirección. Puede colocarse una lista de control de acceso[5] en el puerto 25 (`SMTP`) para esa dirección.
- Observar las cabeceras del *e-mail* para determinar su origen verdadero.
- Ponerse en contacto con el sitio que se identifique como origen con el propósito de alertarlo de la actividad del *spammer*.
- Asegurarse de tener la versión más actualizada del demonio de *mail* (por ejemplo Sendmail) y aumentar el grado de *debug* o *log* que posea el proceso, para detectar o alertar sobre estas actividades. Vigilar el tamaño del archivo de log, ya que puede crecer considerablemente, si se está bajo un *e-mail bombing*.

Desafortunadamente, hasta el momento, no hay manera de prevenir el bombardeo de *e-mail* o *spamming* y es imposible predecir el origen del ataque siguiente. Es trivial obtener acceso a listas de interés o acceder a información que contenga grandes volúmenes de direcciones de *e-mail*, las que proporcionan al atacante direcciones de destino para el *spam*.

Pueden desarrollarse herramientas internas, que pueden ayudar a reconocer y a responder al *e-mail bombing/spamming* reduciendo, de esta manera, el impacto de tal actividad. Tales herramientas deben aumentar las capacidades de log y alertar de mensajes que vienen de un mismo

[3] Responde con un mensaje automáticamente diciendo que estamos de vacaciones o no podemos atender el correo por algún tiempo determinado.

[4] Que altera la identidad de la cuenta que envía el mail

[5] Ver el apartado sobre `tcp-wrappers` de las entregas anteriores.

lugar en un corto período de tiempo. Asimismo, deberían ser capaces de rechazar esos mensajes, o descartarlos.

Si un sitio utiliza un número pequeño de servidores de *e-mail*, podría configurarse un *firewall* para asegurarse de que las conexiones de *smtp* fuera del *firewall* puedan hacerse solamente a sus *pasarelas* de *mail* y a ninguno de los otros equipos. Aunque esta operación no prevendrá un ataque, reduce al mínimo el número de las máquinas disponibles para un ataque basado en SMTP. De este modo, se puede controlar el tráfico entrante SMTP y filtrarlo de manera acorde.

Es importante no contestar y/o hacer un *forward (reenvío)* de los *spams*. De este modo evitaremos que el problema se propague.

2.1.4. Seguridad en WWW

A continuación vamos a tratar algunos aspectos relativos a la seguridad de un servidor web. Nos centraremos en Apache, tratado en una entrega anterior. No hay que confundir la seguridad del servidor web con la de los clientes web (Netscape, Internet Explorer, ...). Los problemas que puedan originarse en unos y otros no tienen nada que ver.

Cuando accedemos a un link a través del navegador web, realizamos una conexión TCP/IP al servidor donde residen las páginas. Esta conexión se suele realizar a través del puerto 80 (HTTP). El navegador envía un mensaje denominado (HTTP request) al servidor y éste responde con los datos solicitados. Podemos también efectuar una conexión mediante telnet en lugar de con un navegador web

```
telnet www.midomio.org 80
```

Supongamos que utilizamos la utilidad `curl`[6]:

```
# curl --head http://www.midominio.org
HTTP/1.1 302 Found
Date: Sat, 04 Jun 2005 16:51:07 GMT
Server: Apache/2.0.54 (Debian GNU/Linux) PHP/4.3.10-15 mod_ssl/2.0.54
    OpenSSL/0.9.7e mod_perl/1.999.21 Perl/v5.8.4
Location: http://www.midominio.org/
Content-Type: text/html; charset=iso-8859-1
```

Un intruso podría obtener de esta manera información sobre la versión de Apache que se encuentra corriendo en el servidor. Sólo tendrá que encontrar un posible *exploit*[7] de esa versión o esperar a que aparezca alguno. Una manera de evitar esto, es modificar la información que aparece en la cabecera que muestra el servidor web Apache[8].

La directiva de Apache `ServerTokens` tiene esta misión:

- `ServerTokens Full`: Muestra todos los nombres de módulos y versiones del servidor
- `ServerTokens Min`: Muestra únicamente el nombre de los módulos y del servidor
- `ServerTokens ProductOnly`: Muestra únicamente el nombre del servidor, en nuestro caso Apache

Este tipo de solución se denomina "seguridad por oscuridad". Se trata de evitar proporcionar información a cualquier usuario y en concreto a los que tienen intenciones no muy recomendables, sobre el software y versiones utilizados.

[6] Para poder disponer de ella:

```
#apt-get install curl
```

Para mayor información sobre esta utilidad consultar el manual `man curl`

[7] Utilizado en el argot para nombrar a un programa que aprovecha un fallo y que permite sacar partido de él.

[8] Estudiado en la 3ª entrega del curso.

Otra recomendación de seguridad es estar informado de las posibles vulnerabilidades que aparecen de los servidores web Apache. Si tenemos una versión con una vulnerabilidad que se soluciona en una versión posterior, debemos actualizar la versión lo antes posible.

Es importante también controlar los datos que están disponibles. Por defecto, todos los datos tienen acceso por parte de cualquier persona que acceda al servidor mediante un navegador web. Cualquier tipo de datos que requieran un acceso restringido deberán configurarse para que se cumpla esa condición. Utilizaremos las directivas de Apache que restringen el acceso a determinadas direcciones IP o a determinados usuarios.

Hasta aquí hemos tratado recomendaciones referentes a la configuración de Apache. Sin embargo, no debemos olvidar la seguridad desde el punto de vista del sistema operativo. Si dentro de un directorio del `DocumentRoot` de Apache tenemos un enlace simbólico a `/etc`, cualquier usuario de internet podrá tener acceso al fichero `/etc/password`. Únicamente tendrá que escribir la URL de forma adecuada y obtendrá en su navegador el fichero de password. Es recomendable el uso de las opciones de Apache `FollowSymLinks` y `SymLinkIfOwnerMatch`.

Permitir el índice de los directorios tampoco es una buena idea, ya que el intruso tendrá información de la estructura de directorios del servidor web. Podría encontrar cualquier fichero que hayamos dejado por olvido y que tenga información sensible. Podemos controlar este comportamiento con `Option Indexes`.

En el caso de usar los mecanismos de autenticación proporcionados por Apache, para limitar el acceso a determinadas áreas de los contenidos del servidor web, tendremos que ser igualmente cuidadosos. Normalmente usaremos el fichero `.htaccess`, restringiendo al máximo el acceso al mismo. No podemos permitir que un intruso, por el método de "prueba y error" construya URLs en las que busque obtener el fichero `.htaccess`. Para ello es conveniente utilizar:

```
<Files .htaccess>
     Order allow, deny
     Deny from all
</Files>
```

Otra opción es configurar el acceso restringido en el propio `httpd.conf`, con lo que evitaríamos la circunstancia anterior.

Existe otro punto que debemos controlar y son los *scripts* y *cgi-bin*. Atacar el sistema operativo vía internet implica buscar algún error en un *script cgi* o lograr que el servidor web haga algo para lo que no fue pensado, como por ejemplo dar al intruso acceso al *shell* del servidor, que ese intruso ejecute comandos arbitrarios en él , o consiga información útil para lograr esos objetivos.

Es obvio que los datos proporcionados a cualquier *cgi script* a través de un formulario, deben ser probados para su validez por una razón u otra, y una de esas razones indudablemente es la seguridad. Dependiendo de lo que el *script* vaya a hacer, la entrada aparentemente inocua de información puede tener graves consecuencias.

Por ejemplo, consideremos el siguiente script en `perl` en el cual se realiza un `finger` al usuario que se indicó en el campo de entrada del formulario y devuelve los resultados al navegador web:

```
#!/usr/local/bin/perl
$|=1;
require 'cgi-lib.pl';
&ReadParse;
Script poco seguro
print &PrintHeader;
open(IN, "/usr/bin/finger $in{'user_id'} |");
@stuff=;
foreach(@stuff) { print; }
exit;
```

Si porporcionamos la siguiente entrada:

```
legolas; /bin/cat / etc/passwd
```

En caso de no poseer soporte de *shadow passwords*, tendremos graves problemas. El *script* anterior constituye un ejemplo muy básico de la forma que un *password grab*[9] podría tomar. El origen del problema radica en que la cadena de entrada podría contener cualquier comando arbitrario. El ejemplo anterior no controló si la entrada en el formulario era un usuario o una "bomba atómica".

Otro aspecto a tener en cuenta es que durante los últimos años, en los cuales se ha extendido el uso de documentos dinámicos, otras vulnerabilidades han entrado en escena. El uso de los *Server Side Includes* (SSIs), en algunos casos significó una extensión nueva de archivo, como `shtml`, en otros significó permitir SSIs para cada documento en el servidor o en un árbol dado del documento. En cualquier caso, permitir SSIs permite un `exec`. Un uso legítimo típico de una etiqueta `exec` es:

```
Esta ápgina ha sido visitada <!-- #exec cgi="/cgi-bin/counter " --> veces
```

Pero imaginemos un sistema de mensajería de alguna clase basado en HTML, por ejemplo un libro de visitas que toma la entrada y construye un documento HTML. Alguien entra y deja:

```
Hey! Que ápginas áms bonitas, évolver pronto!
<!-- #exec cmd="/bin/cat /etc/passwd " -->
```

Si no se están analizando los campos que introducimos, nuevamente tenemos un *password grab*. O podría introducirse cualquier cosa que el servidor pudiera ejecutar, lo cual sería fatal para nuestro sistema si el servicio web se ejecuta como `root`.

Las últimas versiones de Apache proporcionan como opción invalidar los SSIs de tal manera que se pueden habilitar sin el `exec`. Muchos de estos problemas se pueden reducir permitiendo el *chrooting*[10] del servidor web, aunque a pesar de los aumentos que se hacen de seguridad, éstos no son de ninguna forma un ejercicio trivial.

2.2. Analizador de vulnerabilidades Nessus

La herramienta Nessus `http://www.nessus.org` es un proyecto que proporciona a la comunidad de internet una herramienta de análisis de vulnerabilidades fácil de usar, gratuita y muy potente. Nessus es un programa de dominio público desarrollado bajo licencia GPL que permite automatizar la comprobación de forma remota de los posibles agujeros de seguridad por parte de los administradores de sistemas, determinando así por qué sitios pueden acceder los intrusos. Se diseñó para ayudar a identificar y resolver los problemas conocidos de los diferentes servicios que corren en un servidor, permitiendo adelantarnos a los intrusos.

A diferencia de otros analizadores de vulnerabilidades, Nessus no supone nada, es decir, no considerará que un servicio está a la escucha en un puerto específico. Si nuestro servidor web está a la escucha en el puerto 2004, Nessus detectará que hay corriendo un servidor web en ese puerto y buscará las posibles vulnerabilidades del servicio. Del mismo modo, tampoco supondrá que existe una vulnerabilidad a partir de la versión del software que proporciona el servicio, sino que intentará explotarla.

Nessus tiene una arquitectura modular cliente/servidor, lo que permite tener una máquina que realiza los chequeos de seguridad (servidores) y la interfaz gráfica en varias estaciones de trabajo (clientes). Los servidores que se van a chequear no es necesario que ejecuten ningún software adicional y pueden chequearse al mismo tiempo tantos como queramos.

Otro de los puntos fuertes de Nessus es su actualización constante de la base de datos de vulnerabilidades. Todos los nuevos chequeos de seguridad en forma de scripts, se pueden encontrar en `http://www.nessus.org/scripts.php`.

[9] Recolector de passwords

[10] Consiste en limitar el `DocumentRoot` a una especie de partición dentro del sistema de archivos en el que se encuentra. Algo así como una jaula de la que no puede salir.

Los informes que proporciona Nessus a la finalización de un escaneo en busca de vulnerabilidades pueden grabarse en disco en distintos formatos (XML, HTML, Texto, ...) y ofrecen detalles sobre las vulnerabilidades encontradas, así como las referencias de información al respecto.

2.2.1. Instalación de Nessus

La instalación de Nessus es bastante simple. Es recomendable, aunque no estrictamente necesario, la instalación[11] de varios programas externos que aumentan la potencia de Nessus. Estos programas son:

- NMAP (escaneador de puertos)
- Hydra[12] (chequeador de password sencillo)
- Nikto (chequeador de cgi/script)

Nos referimos a ellos porque son los mejores en su categoría. Si se instalan en su ruta por defecto, durante el proceso de instalación de Nessus, estarán disponibles de forma automática.

Los requerimientos referentes a paquetes adicionales necesarios para instalar Nessus son:

- Los paquetes `shareutils`, `bison`, `flex`
- El compilador gcc instalado, ya que se realizará la compilación de Nessus
- La librería `gtk` para el cliente gráfico

La instalación de Nessus se puede hacer de tres formas, siempre con el usuario `root`:

A partir de los paquetes: es la más sencilla para Debian, ya que sólo hay que ejecutar:

```
#apt-get install nessus
```

y en su caso, los paquetes `nessus-plugins` y `nessusd`

Compilación manual: Se bajarán los fuentes de las distintas partes que componen Nessus y se compilarán en el orden que se indica en la página web de Nessus.

Compilación automática: Se baja un único paquete que descomprime los paquetes con los fuentes y realiza el proceso de compilación e instalación.

Optaremos por documentar la tercera opción, ya que la primera no presenta mayor problema.

Bajaremos el archivo `nessus-installer.sh` de `http://www.nessus.org/nessus_2_0.html`

```
                    NESSUS INSTALLATION SCRIPT

Welcome to the Nessus Installation Script !
This script will install Nessus 2.0.10a (STABLE) on your system.
Please note that you will need root privileges at some point so that
the installation can complete.
Nessus is released under the version 2 of the GNU General Public License
(see http://www.gnu.org/licences/gpl.html for details).
To get the latest version of Nessus, visit http://www.nessus.org
Press ENTER to continue
x - creating lock directory
```

[11] `#apt-get install nmap nikto`

[12] `http://thc.org/thc-hydra/`

```
x - extracting nessus.tar.gz (binary)
x - now extracting this archive
x - done
-------------------------------------------------------------------------------
Nessus installation : installation location
-------------------------------------------------------------------------------
Where do you want the whole Nessus package to be installed ?
[/usr/local]
-------------------------------------------------------------------------------
Nessus installation : Ready to install
-------------------------------------------------------------------------------
Nessus is now ready to be installed on this host.
The installation process will first compile it then install it
Press ENTER to continue
-------------------------------------------------------------------------------
Nessus installation : Finished
-------------------------------------------------------------------------------
Congratulations ! Nessus is now installed on this host
. Create a nessusd certificate using /usr/local/sbin/nessus-mkcert
. Add a nessusd user use /usr/local/sbin/nessus-adduser
. Start the Nessus daemon (nessusd) use /usr/local/sbin/nessusd -D
. Start the Nessus client (nessus) use /usr/local/bin/nessus
. To uninstall Nessus, use /usr/local/sbin/uninstall-nessus
. Remember to invoke 'nessus-update-plugins' periodically to update your
  list of plugins
. A step by step demo of Nessus is available at :
        http://www.nessus.org/demo/
Press ENTER to quit
```

Una vez instalado, pasaremos a realizar los pasos que se nos indican. Cuando hemos instalado el servicio de Nessus, es necesario llevar a cabo una serie de sencillos pasos. La primera tarea es generar un certificado para encriptar el tráfico entre el cliente y el servidor. El comando `nessus-mkcert` se encarga de realizar esta función.

```
[root@fedora root]# nessus-mkcert
/usr/local/var/nessus/CA created
/usr/local/com/nessus/CA created
-------------------------------------------------------------------------------
                        Creation of the Nessus SSL Certificate
-------------------------------------------------------------------------------
This script will now ask you the relevant information to create the SSL
certificate of Nessus. Note that this information will *NOT* be sent to
anybody (everything stays local), but anyone with the ability to connect to
    your
Nessus daemon will be able to retrieve this information.
CA certificate life time in days [1460]:
Server certificate life time in days [365]:
Your country (two letter code) [FR]: ES
Your state or province name [none]: Sevilla
Your location (e.g. town) [Paris]: Sevilla
Your organization [Nessus Users United]: Mi Organizacion
-------------------------------------------------------------------------------
                        Creation of the Nessus SSL Certificate
-------------------------------------------------------------------------------
Congratulations. Your server certificate was properly created.
/usr/local/etc/nessus/nessusd.conf updated
The following files were created :
. Certification authority :
```

```
    Certificate = /usr/local/com/nessus/CA/cacert.pem
    Private key = /usr/local/var/nessus/CA/cakey.pem
. Nessus Server :
    Certificate = /usr/local/com/nessus/CA/servercert.pem
    Private key = /usr/local/var/nessus/CA/serverkey.pem
Press [ENTER] to exit
```

La siguiente tarea para completar la instalación, es añadir un usuario. Puede añadirse mediante el comando `nessus-adduser`. Este script preguntará qué método de autenticación queremos usar, el recomendado (y más simple) es el de password. La siguiente pregunta que se nos hace es referente a las reglas para restringir la cuenta del usuario. Podemos restringir a un determinado usuario para que sólo pueda realizar escaneos de determinadas IP. El usuario que creemos será un usuario propio de Nessus y no tendrá reflejo en el sistema.

```
[root@fedora root]# nessus-adduser
Using /var/tmp as a temporary file holder
Add a new nessusd user
----------------------

Login : nessus
Authentication (pass/cert) [pass] :
Login password : nessus
User rules
----------

nessusd has a rules system which allows you to restrict the hosts
that nessus has the right to test. For instance, you may want
him to be able to scan his own host only.
Please see the nessus-adduser(8) man page for the rules syntax
Enter the rules for this user, and hit ctrl-D once you are done :
(the user can have an empty rules set)
Login                : nessus
Password             : nessus
DN                   :
Rules                :
Is that ok ? (y/n) [y] y
user added.
```

Una vez creado el usuario de Nessus tenemos la infraestructura preparada para poder arrancar las partes de que se compone Nessus. Arrancaremos el demonio `nessusd` y a continuación el cliente gráfico. Supondremos que tanto el cliente gráfico como el demonio han sido instalados en la misma máquina.

2.2.2. Actualización de plugins

Antes de realizar un chequeo de un servidor, es recomendable actualizar los plugins que tenemos instalados. Los plugins de Nessus son como las firmas de virus para los antivirus. Cada uno está hecho para una vulnerabilidad específica, explotando la vulnerabilidad en cuestión, o simplemente comprobando versiones de software que son vulnerables. Normalmente están escritps en NAS (*Nessus Attack Scripting Language*) que es un lenguaje propio de Nessus, aunque pueden ser escritos en casi cualquier lenguaje de programación. La actualización de estos plugins debe ser hecha frecuentemente al descubrirse nuevas vulnerabilidades prácticamente todos los días.

El script `nessus-update-plugins` buscará los nuevos scripts que detectan nuevas vulnerabilidades. Este script hará uso de las utilidades `lynx`, `tar` y `gzip`.

```
nessus-update-plugins [-v] [-r <pluginname>] [-h] [-i <pluginname>]
```

Para más detalles, puede verse la entrada en el manual de sistema (`man nessus-update-plugins`).

En caso de que la máquina donde está instalado `nessusd` esté situada detrás de un proxy, será necesario crear el archivo `.nessus-update-pluginsrc` en el directorio `$HOME` del usuario que esté ejectando el script. Incluiremos las siguientes líneas en este fichero:

```
proxy_user= username
proxy_passwd= password
proxy= address_of_your_proxy
```

Un ejemplo de un fichero sería:

```
proxy_user=hugo
proxy_passwd=topsecr3t
proxy=proxy.miordenador.es:3128
```

2.2.3. Arrancando Nessus

Una vez que tenemos Nessus correctamente instalado y configurado y con los últimos plugins instalados, podemos arrancar el demonio. La forma más simple es arrancar con el usuario root el demonio `nessusd`

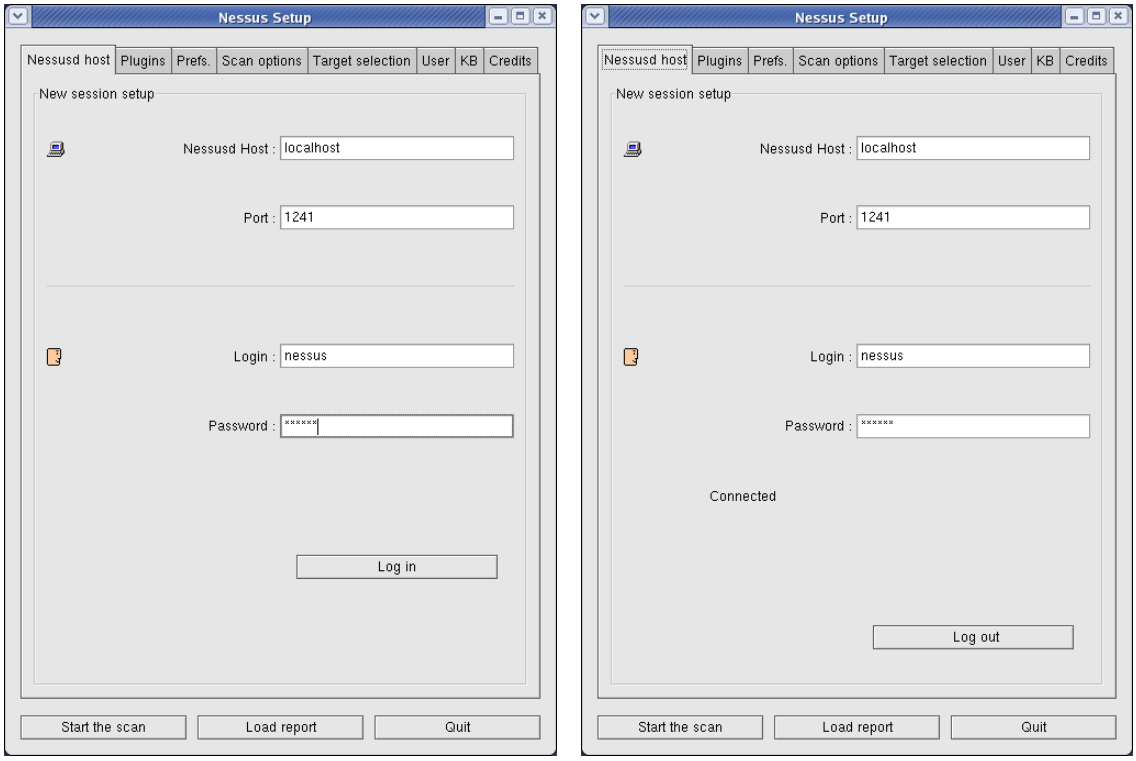
```
nessusd -D
```

A continuación, será necesario utilizar un cliente, instalado en el mismo servidor o en otro ordenador, para conectarnos y comenzar los escaneos. Existe también la posibilidad de prescindir de la interfaz gráfica y utilizar Nessus desde la línea de comandos. El cliente gráfico se arranca:

```
nessus
```

Una vez arrancado, es necesario conectarnos al servidor donde se encuentra arrancado `nessusd`. Es necesario proporcionar la dirección IP del servidor Nessus, así como el usuario y clave con el que nos conectaremos.

Figura 2.1: Acceso de usuario en Nessus

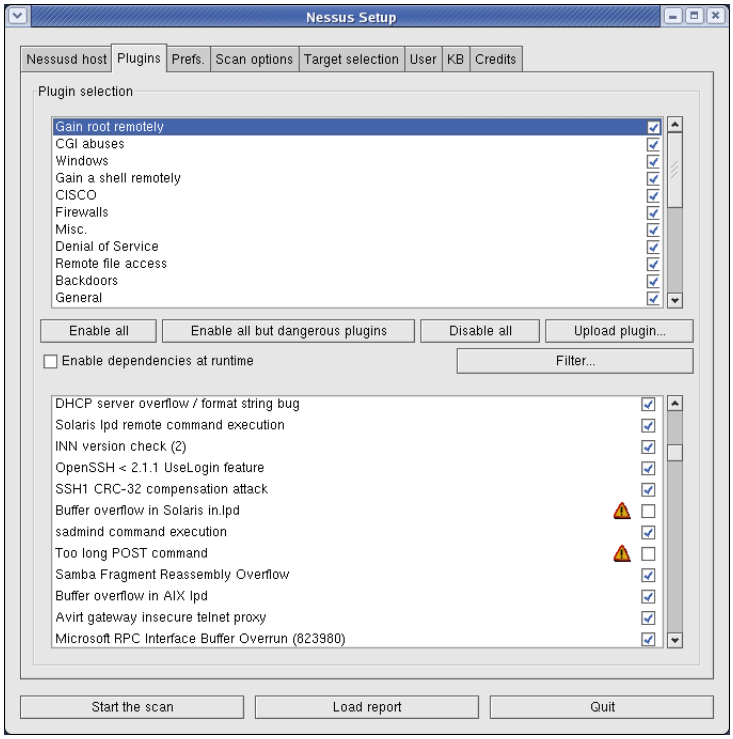


El cliente se conecta al servidor a través de una conexión SSL, cargando la lista de los plugins que tiene instalados. La primera vez que lo ejecutemos, el certificado pedirá una confirmación del mismo antes de ser descargado. Esto asegura que en el futuro las comunicaciones se realizarán con este servidor.

2.2.4. Usando Nessus

Como ya hemos indicado, uno de los aspectos que destacan a Nessus es la cantidad de plugins. Dependiendo de los plugins que seleccionemos, obtendremos un informe más o menos útil para nuestros propósitos. Hay que tener en cuenta que algunos plugins pueden darnos información de vulnerabilidades existentes cuando esto no es así, provocando lo que se denomina falsos positivos. No debemos sorprendernos si realizamos el escaneo de nuestro sistema Linux y detectamos algún tipo de vulnerabilidad relacionada con Windows. Este tipo de problemas no hay que interpretarlos como inestabilidad o falta de confianza sobre Nessus. Por el contrario, el origen es la mala utilización de algunos de los plugins (uso de plugins que buscan vulnerabilidades de Windows en sistemas Unix). Estos comportamientos erróneos también se producen en el software comercial destinado a este propósito.

Figura 2.2: Plugins de Nessus



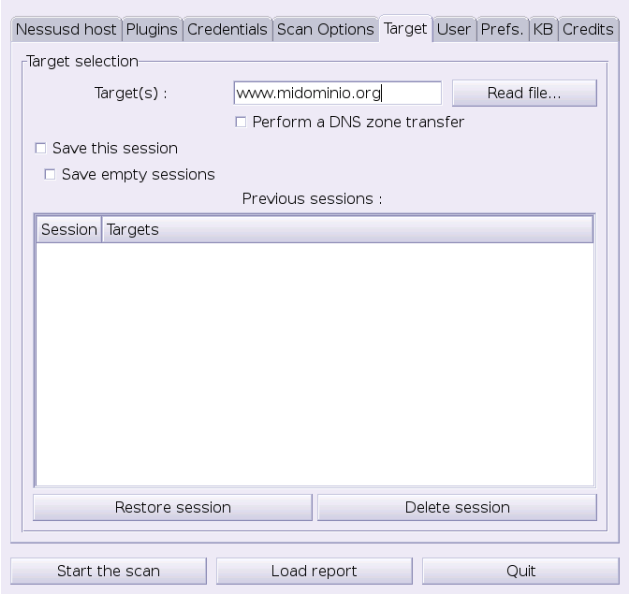
Los plugins están categorizados de varias formas, lo que en ocasiones puede llevarnos a confusión. Uno de los métodos de agrupación de los plugins, es por categoría. Más importante aún es la distinción de los plugins basándonos en su peligrosidad o habilidad para provocar ataques DoS. Estos plugins pueden llegar a bloquear un sistema que sea vulnerable al ataque, por lo que es recomendable utilizarlos con cuidado. No causarán pérdidas irreparables, pero será necesario reiniciar el sistema[13]. Existe un botón donde podemos seleccionar todos los plugins o únicamente aquellos que no sean peligrosos. Otro método de categorización de los plugins es mediante familias:Windows, FTP SNMP, SMB, Cisco, etc.

Otra de las partes críticas del proceso de escaneo, es el escaneo de puertos. Es un proceso en el que se identifican los puertos activos para una dirección IP. Cada puerto está unido a una aplicación específica. Nessus tiene la inteligencia suficiente para realizar un chequeo únicamente si el programa para ese chequeo está disponible. Por ejemplo, sólo se ejecutarán los plugins para servidores web si se encuentra uno. Debido a que muchos servicios no corren en los puertos por defecto, Nessus tiene un plugin que se encarga de reconocer qué aplicaciones están asociadas a cada puerto que se encuentra a la escucha.

Existen más pestañas que nos permiten configurar otras opciones de Nessus, pero no entraremos en ellas. Lo que pretendemos es tener una primera toma de contacto con esta utilidad, lo suficiente para permitirnos lanzar escaneos contra servidores para identificar sus vulnerabilidades. Así, pasaremos directamente a la pestaña **Target Selection**, donde debemos identificar nuestros objetivos. Pueden especificarse mediante una dirección IP, como una subred o como un rango de direcciones IP. Una vez configurado este último punto, podemos comenzar la ejecución del escaneo.

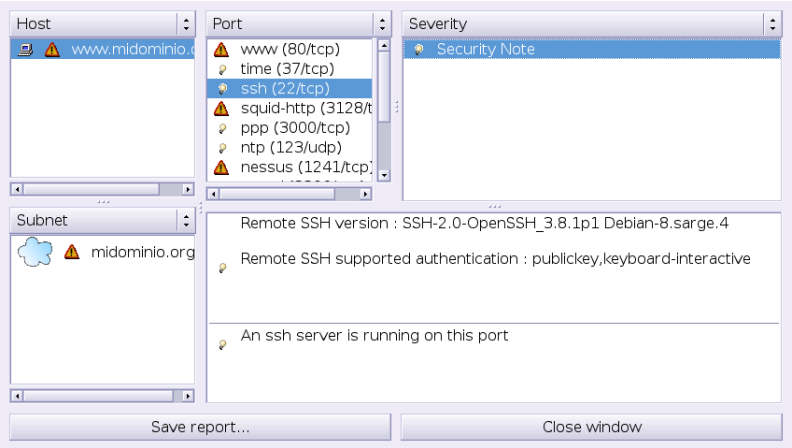
[13]Están identificados en la pestaña de plugins por una señal de peligro.

Figura 2.3: Selección de objetivos



Como ejemplo, utilizaremos el mismo servidor donde hemos instalado el servidor y cliente de Nessus.

Figura 2.4: Informe de vulnerabilidades presentado por Nessus



El informe obtenido puede guardarse en varios formatos (entre los que se encuentra HTML), para su posterior estudio.

2.3. Crackeadores de password: John the Ripper

Uno de los crackeadores de claves más famosos es John the Riper. Está disponible para Unix (hasta 11 tipos independientes de la arquitectura), DOS, Win32, BeOS y OpenVMS. A pesar de que muchas veces su uso tiene oscuras intenciones, se creó como una herramienta capaz de detectar las claves débiles de un sistema Unix.

En los comienzos de Unix, incluso en la actualidad, los administradores de sistemas no le daban importancia a que cualquier usuario del sistema pueda leer el fichero `/etc/passwd`. Confiaban ciegamente en la función `crypt()` que se encargaba de almacenar en este fichero la clave encriptada. Incluso el sistema `shadow` no libra a un sistema de obtener un fichero `/etc/passwd` con las claves encriptadas, basta con conseguir `/etc/passwd` y `/etc/shadow` y migrarlos a otro sistema donde se tenga acceso como `root`. A continuación se verá un ejemplo que representa esta problemática.

2.3.1. Instalacion

Se va a construir la herramienta *John the Ripper* a partir de los fuentes[14], obteniendo de esta manera un binario para el sistema. Lo primero es descargar la última versión de `http://www.openwall.com/john/` y descomprimirlo en el directorio que se desea.

Una vez desempaquetado se procede a compilar el código fuente:

```
cd scr
make
```

Con esto se obtiene una lista de los sistemas soportados. Puede elegirse uno de los que aparece en la lista o simplemente `Generic`:

```
make SISTEMA
```

ó

```
make generic
```

En el caso que la compilación se lleve a cabo con éxito, el binario de la herramienta es `run/john`. Es recomendable que el directorio donde se encuentre instalada esta herramienta tenga un acceso restringido para evitar que cualquier usuario puede hacer un uso fraudulento de sus funcionalidades.

2.3.2. Crackeando el fichero /etc/passwd

La instalación de esta herramienta es extremadamente fácil tal como acaba de verse, sin embargo, su potencia es muy grande, siendo una herramienta de uso obligado para todos los administradores de sistemas. Con un uso regular de John the Riper se obtiene la certeza de una buena elección de las claves de nuestros usuarios.

Una de las principales funcionalidades es el descubrimiento de las claves mediante ataques de diccionario. Este tipo de ataque consiste en utilizar una base de datos de posibles claves cuyo origen es un diccionario. La Real Academia de la Lengua Española proporciona una gran base de datos con esta información.

Para el ejemplo que nos ocupa, en caso de tener soporte `shadow` para las claves será necesario ejecutar la utilidad `pwunconv`, que repone las claves encriptadas al fichero `/etc/passwd`. Se realizará una copia de este fichero para trabajar con él y a continuación se ejecuta `pwconv` para volver a la situación inicial y devolver al sistema el soporte `shadow`. Para realizar esto también puede utilizarse la utilidad `unshadow` que se incluye con John the Riper.

Como primera opción se utilizará el método single, que busca posibles claves en el propio fichero (muchas personas utilizan como clave su nombre o apellidos):

```
root@guadalinex:/usr/local/src/john-1.6# ./run/john -single passwd.john
Loaded 1 password (Standard DES [24/32 4K])
guesses: 0  time: 0:00:00:00 100%  c/s: 738  trying: hugo1934 - hugo1969
```

[14] Se puede optar por:

```
#apt-get install john wenglish wspanish
```

En este caso no ha detectado ninguna clave al ser el fichero elegido el de un sistema con pocos usuarios.

El siguiente paso es realizar la búsqueda de claves comparando contra un listado (lo mejor es tener un diccionario de la lengua de origen de los usuarios del sistema) de posibles claves. Sería el modo *wordlist*. Para indicar el fichero que se utilizará para la comparación se utiliza la opción `-wordfile`.

Para utilizar modificaciones del listado proporcionado se utiliza la opción `-rules`. Las reglas que se utilizan para generar estas variaciones se encuentra en el fichero de configuración `john.ini`, recomendando su visualización para comprender las variaciones que se van a realizar.

A continuación se ve la ejecución de la utilidad indicando que la lista de posibles claves está en `pasword.lst` y utilizando reglas para realizar variaciones de las mismas, estando el fichero de claves en `password.john`.

```
root@guadalinex:/usr/local/src/john-1.6# ./run/john -wordfile:run/password.
    lst -rules passwd.john
Loaded 1 password (Standard DES [24/32 4K])
guesses: 0  time: 0:00:00:01 100%  c/s: 101207  trying: Raptorin - Zenithin
root@guadalinex:/usr/local/src/john-1.6# ./run/john -show passwd.john root:
    administ:0:0:root:/root:/bin/bash
1 password cracked, 2 left
```

En este caso hemos encontrado la clave de `root` (no haría falta ninguna más para poner en apuros al administrador del sistema).

Es recomendable editar el fichero `password.lst` para que contenga posibles claves en español, ya que los valores que contiene se refieren a claves en inglés. Como ya se indicó lo mejor sería conseguir un listado de las palabras de un diccionario, seguro que algún usuario ha utilizado como contraseña algo como "ornitorrinco" pensando que a nadie se le va a ocurrir probar este clave (a John si se le ocurrirá).

Otra forma de realizar de generar claves para chequear es utilizando combinaciones de letras y número. Este sería el modo incremental de verificación de John the Riper. Las opciones disponibles son:

`alpha` Genera palabras con letras solamente, es decir 26 letras

`digits` Genera palabras con numeros solamente, desde el 0 hasta el 9

`all` Genera palabras con letras, numeros y caracteres especiales, en total son 90 caracteres

Cuantas más combinaciones más tiempo se tardará en chequear todo el fichero de claves. Sin embargo, hay que tener en cuenta que un posible intruso suele ser un individuo al cual le sobran recursos y utiliza su tiempo libre para buscar vulnerabilidades.

Son muchas las opciones que presenta John the Riper y se recomienda la lectura de
`http://www.decowar.com/manual_john_the_ripper.htm`
donde se explica de forma detallada cómo crear nuevas reglas. Para un uso ético de esta herramienta bastaría con las opciones que se han detallado anteriormente, la referencia anterior sería para usos más oscuros[15].

2.4. Detección de intrusiones

2.4.1. Razones para la detección de intrusiones

Hasta ahora hemos cubierto algunos aspectos sobre la seguridad del sistema en el ámbito preventivo. Cuando un atacante decide probar su suerte contra nuestro sistema, lo primero que hace

[15]Como se está viendo todas las herramientas de detección de vulnerabilidades tienen su "reverso tenebroso", dependiendo de si la usa Obi-Wan Kenobi o Darth Vader el objetivo que se busca es uno u otro.

es recopilar cuanta información le sea posible acerca del mismo. Toda información que consiga averiguar puede serle útil: sistema operativo, versión, servicios que ofrecemos, versión de los programas que tenemos... Cualquiera de estos datos puede ser suficiente para que su ataque tenga éxito. Basta con que el atacante vea, por ejemplo, que tenemos una versión vieja de un programa, aunque no tenga éste ninguna vulnerabilidad importante, para que se dé cuenta de que no somos administradores muy cuidadosos y probablemente tengamos otros servicios descuidados.

Esto nos exige no sólo que cuidemos las versiones y posibles parches de seguridad a aplicar en nuestro sistema, sino también el vigilar los intentos de acceso.

La técnica más utilizada por los potenciales intrusos de nuestro sistema para obtener información acerca de nosotros, es el barrido de puertos. Esta técnica se basa en intentar conectarse a cada uno de los puertos que tiene abiertos nuestro servidor, anotando qué es lo que tiene activo y analizando dicha información. Una de las herramientas más comunes para realizar barridos de puertos es `nmap`.

No entraremos en el uso de `nmap`, ya que se trató en una entrega anterior. Recordemos que tras una ejecución de esta herramienta, obtendremos un listado de los puertos abiertos, pudiendo el atacante revisar si encuentra alguna versión vieja o vulnerable de software.

```
[root@fedora root]# nmap -sT fedora
Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2004-05-18 13:38
   CEST
Interesting ports on localhost (172.26.0.40):
(The 1648 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
80/tcp    open  http
110/tcp   open  pop-3
111/tcp   open  rpcbind
443/tcp   open  https
514/tcp   open  shell
10000/tcp open  snet-sensor-mgmt
10082/tcp open  amandaidx
10083/tcp open  amidxtape
32770/tcp open  sometimes-rpc3
Nmap run completed -- 1 IP address (1 host up) scanned in 3.660 seconds
```

Ahora podemos intentar averiguar más información sobre algunos de los servicios:

```
[root@fedora root]# telnet 172.26.0.40
Trying 172.26.0.40...
Connected to 172.26.0.40.
Escape character is '^]'.
Fedora Core release 1 (Yarrow)
Kernel 2.4.22-1.2115.nptl on an i686
login:
[root@fedora root]# telnet 172.26.0.40 22
Trying 172.26.0.40...
Connected to 172.26.0.40.
Escape character is '^]'.
SSH-1.99-OpenSSH_3.6.1p2
[root@fedora root]# telnet 172.26.0.40 25
Trying 172.26.0.40...
Connected to 172.26.0.40.
Escape character is '^]'.
220 fedora.elpiso.es ESMTP Sendmail 8.12.10/8.12.10; Tue, 18 May 2004
    13:50:11 +0200
```

A partir de la información obtenida con `nmap`, hemos logrado averiguar los siguientes datos acerca de las versiones del software instalado:

- SSH-1.99-OpenSSH_3.6.1p2,
- 8.12.10 de sendmail y
- kernel 2.4.22-1.2115.nptl.

Bastaría con buscar información referente a vulnerabilidades sobre este software para explotar agujeros que nos permitirían acceso como root al servidor.

Cabe mencionar que nmap no es una herramienta utilizada exclusivamente por atacantes. Puede utilizarse para barrer nuestras máquinas buscando servicios que hayamos dejado abiertos por error.

2.4.2. Intentos de intrusión en el sistema (Portsentry)

Vemos claramente la gran cantidad de información que en menos de un minuto puede ser expuesta por nuestro mal administrado servidor. Incluso aunque el servidor se revise periodicamente, ¿por qué permitir que un desconocido sepa qué versiones de software tenemos instaladas? Podría guardar estos datos a la espera de la aparición de nuevas vulnerabilidades, adelantándose a nuestras actuaciones.

Como acabamos de ver, es fundamental la detección de estos barridos. Detectar un barrido de puertos es muy fácil; se producirán muchas conexiones casi simultáneas a una gran cantidad de puertos originadas desde la misma máquina. Aunque los programas que se dedican a realizar estos barridos se han vuelto muy sofisticados y cada vez es más difícil detectarlos por las diferentes estrategias que emplean[16], el principio básico es el mismo. Hay un excelente programa dedicado precisamente a encontrar este patrón y tomar la acción que le indique el administrador del sistema: Portsentry, de Psionic.

Portsentry es un programa muy sencillo. Su misión es escuchar a los puertos que le indiquemos que deben permanecer siempre inactivos. En caso de llegar una conexión a uno de ellos, puede marcarlo en la bitácora del sistema, bloquear toda la comunicación con la dirección identificada como agresora, o correr un comando externo. Podemos bajar la última versión de `http://sourceforge.net/projects/sentrytools/`.

La compilación[17] de Portsentry es muy sencilla. Lo primero que haremos es extraer los ficheros que componen el paquete:

```
[root@fedora src]# tar -zxvf portsentry-1.2.tar.gz
portsentry_beta/
portsentry_beta/portsentry.c
portsentry_beta/portsentry.h
portsentry_beta/portsentry_io.c
portsentry_beta/portsentry_io.h
portsentry_beta/portsentry_util.c
portsentry_beta/portsentry_util.h
portsentry_beta/portsentry_config.h
portsentry_beta/portsentry_tcpip.h
portsentry_beta/portsentry.ignore
portsentry_beta/portsentry.conf
portsentry_beta/Makefile
portsentry_beta/README.COMPAT
```

[16] Nmap sabe hacer desde una sencilla conexión TCP hasta un barrido silencioso con SYN, FIN, Xmas, Null, UDP, paquetes fragmentados, barridos paralelos de diferentes tipos.

[17] Si deseamos instalar el paquete usando el comando `apt-get`, escribiremos

```
#apt-get install portsentry
```

```
portsentry_beta/README.install
portsentry_beta/README.methods
portsentry_beta/README.stealth
portsentry_beta/CHANGES
portsentry_beta/CREDITS
portsentry_beta/LICENSE
portsentry_beta/ignore.csh
```

Es conveniete leer los distintos ficheros README que se han obtenido del paquete, para conocer con más detalle la operación del programa y los pasos a seguir. Una vez descomprimido el paquete, es necesario compilar. Esto se hace con un simple `make <sistema>`, sustituyendo `<sistema>` por nuestro tipo de sistema operativo. En nuestro caso es un sistema Linux:

```
[root@fedora portsentry_beta]# make linux
SYSTYPE=linux
Making
cc -O -Wall -DLINUX -DSUPPORT_STEALTH -o ./portsentry ./portsentry.c \
./portsentry_io.c ./portsentry_util.c
```

Por último, pasamos a instalar portsentry:

```
[root@fedora portsentry_beta]# make install
Creating psionic directory /usr/local/psionic
Setting directory permissions
Creating portsentry directory /usr/local/psionic/portsentry
Setting directory permissions
chmod 700 /usr/local/psionic/portsentry
Copying files
cp ./portsentry.conf /usr/local/psionic/portsentry
cp ./portsentry.ignore /usr/local/psionic/portsentry
cp ./portsentry /usr/local/psionic/portsentry
Setting permissions
chmod 600 /usr/local/psionic/portsentry/portsentry.ignore
chmod 600 /usr/local/psionic/portsentry/portsentry.conf
chmod 700 /usr/local/psionic/portsentry/portsentry
Edit /usr/local/psionic/portsentry/portsentry.conf and change
your settings if you haven't already. (route, etc)
WARNING: This version and above now use a new
directory structure for storing the program
and config files (/usr/local/psionic/portsentry).
Please make sure you delete the old files when
the testing of this install is complete.
```

Con esto, Portsentry quedará instalado en el directorio `/usr/local/psionic/portsentry` listo para ser configurado.

Configuración de portsentry

La configuración de Portsentry se hace en el archivo[18] `/usr/local/psionic/portsentry/portsentry.conf`. A primera vista el archivo parece muy complicado, con muchas opciones. Sin embargo, configurarlo es más fácil de lo que parece.

Portsentry tiene varios modos de operación. El más común y sencillo es el modo clásico, y el más potente (aunque no disponible en todos los sistemas) es el modo avanzado.

Modo clásico. En este modo, le especificamos a Portsentry que escuche determinados puertos TCP y UDP, especificados con las opciones UDP_PORTS y TCP_PORTS por los cuales

[18] En `/etc/portsentry/portsentry.conf` si hemos usado el `.deb`

no estamos dando ningún servicio y son poco solicitados. Por ejemplo, puede que nuestro servidor no esté dando servicio de red SMB (Samba, red tipo Microsoft). Sin embargo, es común que las computadoras windows manden mensajes broadcast buscando a un sistema en específico, con lo que podríamos recibir una gran cantidad de alertas en falso. Vienen varios puertos predefinidos en el archivo de configuración, y son buenos como recomendación inicial.

Modo *stealth*. En este modo Portsentry abre sockets crudos (raw), lo que le permite detectar una mayor cantidad de barridos y ataques tales como ataques de conexión normal, SYN/half-open, FIN, NULL y XMAS. Este modo es un tanto experimental, por lo cual no funcionará en todos los sistemas.

Modo avanzado. Este modo es también considerado, hasta cierto punto, perteneciente a la categoría stealth. En este modo, Portsentry no abre ningún puerto, sino que le pide al kernel que le notifique si llega alguna petición a algún puerto menor al especificado en las opciones ADVANCED_PORTS_TCP y ADVANCED_PORTS_UDP. Será necesario excluir algunos puertos que puedan generar falsas alarmas, como el ejemplo que comentábamos sobre SMB. Para excluir estos puertos tenemos las opciones ADVANCED_EXCLUDE_TCP y ADVANCED_EXCLUDE_UDP.

Como ventaja adicional del modo avanzado, está que al ejecutar `netstat -na` no nos reportará los puertos que está escuchando, dado que no están realmente abiertos. Ésto puede simplificar un tanto nuestro trabajo como administradores. El modo avanzado es mucho más sensible que el modo clásico, dado que escucha a muchos más puertos, por lo que puede efectivamente causar una negación de servicio si no es configurado con cuidado.

Tras haber especificado los puertos que deseamos escuchar, hay algunos parámetros adicionales que debemos especificar:

IGNORE_FILE es el nombre del archivo que incluye la lista de direcciones en las que confiamos, y por tanto no queremos bloquear si intentan acceder a un puerto bloqueado.

HISTORY_FILE contiene la lista de direcciones que Portsentry ha detectado intentando acceder a puertos monitorizados.

BLOCKED_FILE es equivalente a HISTORY_FILE, pero relevante únicamente a la sesión actual de Portsentry.

BLOCK_TCP especifica qué hacer cuando un barrido de puertos TCP es detectado. Tiene tres posibles valores:

0	sólo registrar el intento,
1	bloquear la dirección que intentó averiguar acerca de nosotros, y
2	correr un comando externo especificado en KILL_RUN_CMD.

BLOCK_UDP es equivalente a BLOCK_TCP para barridos de puertos UDP.

KILL_ROUTE guarda el comando utilizado para descartar toda la comunicación con una dirección, normalmente mediante reglas de filtrado de paquetes.

KILL_HOSTS_DENY tiene la línea que deberá ser agregada a `/etc/hosts.deny` para que la dirección atacante sea bloqueada por TCPwrappers. Es conveniente activarlo, pues a diferencia de las reglas manejadas por KILL_ROUTE este archivo permanecerá aunque el sistema se rearranque.

KILL_RUN_CMD puede guardar un comando para ser ejecutado en caso de ser detectada una intrusión. No recomendamos utilizar esta opción, ya que puede fácilmente llevar a una negación de servicio. Hay administradores que sugieren utilizar esta opción para lanzar un contraataque contra el atacante. La mejor defensa es tener nuestro sistema seguro, no atacar al enemigo. Al atacar al enemigo, lo más probable es que centre más su atención en nosotros[19] y, con el paso del tiempo, logre penetrar nuestra seguridad. Es mucho mejor aparentar que no ocurrió nada o simplemente tirar la conexión que atacar de vuelta.

SCAN_TRIGGER indica con qué retardo se marcará un intento de conexión fallido como un ataque. Probablemente, si a la primera bloqueamos toda comunicación con el presunto atacante, dejaremos fuera a muchos usuarios legítimos que por casualidad hicieron la conexión equivocada. Sin embargo, si ponemos un número muy alto nos exponemos a dar más información de la que hubiéramos querido. Un valor de 1 ó 2 es recomendado, aunque los muy paranoicos querrán mantenerlo en 0.

Arranque automático de portsentry

La manera más genérica[20] de iniciar portsentry es incluirlo en el último archivo que se ejecuta al iniciar el sistema `/etc/rc.local`. Basta con agregar al final de éste las líneas necesarias para levantar a portsentry con la configuración que deseemos. Las opciones de arranque de portsentry son:

-tcp Iniciar en modo clásico, escuchar TCP.

-udp Iniciar en modo clásico, escuchar UDP.

-stcp Iniciar en modo stealth, escuchar TCP

-sudp Iniciar en modo stealth, escuchar UDP

-atcp Iniciar en modo avanzado, escuchar TCP

-audp Iniciar en modo avanzado, escuchar UDP

Normalmente levantaremos dos copias del programa en el mismo modo general, una escuchando UDP y la otra TCP.

Respuesta de portsentry ante un ataque

El simple hecho de que Portsentry evite ciertos ataques al sistema es de por sí muy bueno y deseable. Sin embargo, para que ésto nos sea realmente útil, tenemos que analizar nuestras bitácoras y llevar registro de quién y cuándo intentó barrer nuestros puertos. Además, sólo leyendo las bitácoras sabremos si estamos limitando de más, bloqueando el acceso de máquinas legítimas.

Afortunadamente, Portsentry utiliza syslog para informar de toda la información que el administrador debe saber, por lo cual todo lo que necesitamos estará típicamente en el archivo `/var/log/messages`, o donde se lo hayamos especificado en el `syslogd.conf`. La detección de un barrido hecho por nmap en un sistema Linux se ve así:

```
[root@fedora root]# nmap -sT fedora
May 18 14:17:44 fedora portsentry[2642]: attackalert: Connect from host:
    localhost/172.26.0.40 to TCP port: 79
May 18 14:17:44 fedora portsentry[2642]: attackalert: Host 172.26.0.40 has
    been blocked via wrappers with string: "ALL: 172.26.0.40"
May 18 14:17:44 fedora xinetd[2648]: libwrap refused connection to shell (
    libwrap=in.rshd) from 172.26.0.40
```

[19] Puede considerarlo un reto al enfrentarse a un administrador de un gran nivel.
[20] Si lo hemos instalado a partir de las fuentes.

```
May 18 14:17:44 fedora portsentry[2642]: attackalert: Connect from host:
    localhost/172.26.0.40 to TCP port: 635
May 18 14:17:44 fedora portsentry[2642]: attackalert: Host: 172.26.0.40 is
    already blocked. Ignoring
May 18 14:17:44 fedora portsentry[2642]: attackalert: Connect from host:
    localhost/172.26.0.40 to TCP port: 1524
May 18 14:17:44 fedora portsentry[2642]: attackalert: Host: 172.26.0.40 is
    already blocked. Ignoring
May 18 14:17:44 fedora xinetd[2650]: libwrap refused connection to amidxtape
     (libwrap=amidxtaped) from 172.26.0.40
May 18 14:17:46 fedora portsentry[2642]: attackalert: Connect from host:
    localhost/172.26.0.40 to TCP port: 119
May 18 14:17:46 fedora xinetd[2652]: libwrap refused connection to amandaidx
     (libwrap=amindexd) from 172.26.0.40
May 18 14:17:46 fedora portsentry[2642]: attackalert: Host: 172.26.0.40 is
    already blocked. Ignoring
May 18 14:17:48 fedora xinetd[2654]: warning: can't get client address:
    Connection reset by peer
May 18 14:17:48 fedora portsentry[2642]: attackalert: Connect from host:
    localhost/172.26.0.40 to TCP port: 1
May 18 14:17:48 fedora xinetd[2655]: warning: can't get client address:
    Connection reset by peer
May 18 14:17:48 fedora portsentry[2642]: attackalert: Host: 172.26.0.40 is
    already blocked. Ignoring
May 18 14:17:48 fedora portsentry[2642]: attackalert: Connect from host:
    localhost/172.26.0.40 to TCP port: 11
May 18 14:17:49 fedora portsentry[2642]: attackalert: Host: 172.26.0.40 is
    already blocked. Ignoring
May 18 14:17:49 fedora portsentry[2642]: attackalert: Connect from host:
    localhost/172.26.0.40 to TCP port: 15
May 18 14:17:49 fedora portsentry[2642]: attackalert: Host: 172.26.0.40 is
    already blocked. Ignoring
May 18 14:17:49 fedora portsentry[2642]: attackalert: Connect from host:
    localhost/172.26.0.40 to TCP port: 111
May 18 14:17:49 fedora portsentry[2642]: attackalert: Host: 172.26.0.40 is
    already blocked. Ignoring
May 18 14:17:49 fedora portsentry[2642]: attackalert: Connect from host:
    localhost/172.26.0.40 to TCP port: 143
May 18 14:17:49 fedora portsentry[2642]: attackalert: Host: 172.26.0.40 is
    already blocked. Ignoring
May 18 14:17:49 fedora portsentry[2642]: attackalert: Connect from host:
    localhost/172.26.0.40 to TCP port: 540
May 18 14:17:49 fedora portsentry[2642]: attackalert: Host: 172.26.0.40 is
    already blocked. Ignoring
May 18 14:17:49 fedora portsentry[2642]: attackalert: Connect from host:
    localhost/172.26.0.40 to TCP port: 1080
(...)
```

Ésta sería la respuesta con el archivo de configuración por defecto. Sin embargo, hemos visto que existen varias acciones a tomar dependiendo de lo que consideremos oportuno como respuesta. Estas acciones quedarán reflejadas también en el fichero de log.

Cuando configuremos portsentry es muy común que marquemos direcciones como atacantes por error. Borrarlas de la lista de bloqueo es muy sencillo. Nuestro primer paso será editar el archivo `/etc/hosts.deny` y buscar la dirección que queremos eliminar. En caso de que hayamos establecido algún tipo de filtrado de paquetes (por ejemplo con iptables) será necesario permitir de nuevo la entrada de los paquetes con origen en la dirección que hemos vetado por error.

Las posibilidades de portsentry son tantas como nosotros configuremos, aunque debemos tener

cuidado y afinar bien la configuración para evitar negar el acceso a direcciones por error.

2.4.3. Integridad del sistema (Tripwire)

Existe un **bug** en la **versión actual de Tripwire** que provoca un error en la instalación **sobre Fedora**. Aún así, hemos preferido dar información sobre esta herramienta a la espera de obtener una versión de Tripwire que solucione este error. Puede encontrarse información más detallada en `http://www.redhat.com/archives/fedora-list/2003-November/msg05954.html`.

No existen sistemas perfectos e invulnerables a los ataques de usuarios maliciosos. Además de las posibles medidas preventivas que tomemos (cortafuegos, parches, políticas, ...) siempre cabe la posibilidad de que un intruso logre entrar en nuestro sistema. Normalmente los ataques que se producen, conllevan la modificación parcial del sistema (archivos de configuración, páginas web, ...). Estas modificaciones tienen como objetivo dejar una marca o firma en el servidor como prueba de la violación del sistema o posibles puertas traseras para accesos posteriores.

Tripwire asumirá que los controles preventivos de seguridad han fallado y que nuestro sistema ha sido alterado. El atacante intentará por todos los medios que el administrador no sepa los ficheros que han sido modificados. Es aquí donde interviene Tripwire, alertando al administrador de los cambios que se produzcan en el sistema.

Tripwire realiza una monitorización de la integridad de los archivos que le indiquemos, detectando cualquier cambio en los ficheros indicados. El cambio puede ser tanto de contenido como de permisos.

Instalación de Tripwire

Para bajar la última[21] versión disponible de Tripwire accedemos a
`http://www.tripwire.org/download/index.php`
y seleccionaremos para bajar el fichero `tripwire-2.3-47.bin.tar.gz`.
Una vez bajado en nuestro sistema:

```
# cd /usr/local
# tar -zxvf /home/hugo/Tripwire/tripwire-2.3-47.bin.tar.gz
```

Tripwire utiliza dos claves, que pueden ser palabras u oraciones, para almacenar su información de forma encriptada. La denominada "site key" se emplea para encriptar los archivos de configuración y las políticas. La denominada "local key" se utiliza para encriptar la información referida al estado de los ficheros que se monitorizan. Para finalizar la instalación de Tripwire es necesario ejecutar el script `install.sh`, el cual creará las claves que acabamos de mencionar.

```
[root@fedora tripwire-2.3]# ./install.sh
Installer program for:
Tripwire(R) 2.3 Open Source for LINUX
Copyright (C) 1998-2000 Tripwire (R) Security Systems, Inc.  Tripwire (R)
is a registered trademark of the Purdue Research Foundation and is
licensed exclusively to Tripwire (R) Security Systems, Inc.
LICENSE AGREEMENT for Tripwire(R) 2.3 Open Source for LINUX
Please read the following license agreement.  You must accept the
agreement to continue installing Tripwire.
Press ENTER to view the License Agreement.
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991
```

[21] U optar por

```
#apt-get install tripwire
```

```
 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
                      59 Temple Place, Suite 330, Boston, MA  02111-1307
                         USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.
                            Preamble
  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
(...)
Please type "accept" to indicate your acceptance of this
license agreement. [do not accept] accept
Using configuration file install.cfg
Checking for programs specified in install configuration file....
/usr/lib/sendmail exists.  Continuing installation.
/bin/vi exists.  Continuing installation.
-------------------------------------------------------
Verifying existence of binaries...
./bin/i686-pc-linux_r/siggen found
./bin/i686-pc-linux_r/tripwire found
./bin/i686-pc-linux_r/twprint found
./bin/i686-pc-linux_r/twadmin found
This program will copy Tripwire files to the following directories:
        TWBIN: /usr/sbin
        TWMAN: /usr/man
     TWPOLICY: /etc/tripwire
     TWREPORT: /var/lib/tripwire/report
         TWDB: /var/lib/tripwire
 TWSITEKEYDIR: /etc/tripwire
TWLOCALKEYDIR: /etc/tripwire
CLOBBER is false.
Continue with installation? [y/n]
(...)
-------------------------------------------------------
The Tripwire site and local passphrases are used to
sign a variety of files, such as the configuration,
policy, and database files.
Passphrases should be at least 8 characters in length
and contain both letters and numbers.
See the Tripwire manual for more information.
-------------------------------------------------------
Creating key files...
(When selecting a passphrase, keep in mind that good passphrases typically
have upper and lower case letters, digits and punctuation marks, and are
at least 8 characters in length.)
Enter the site keyfile passphrase: sitekey
Verify the site keyfile passphrase: sitekey
Generating key (this may take several minutes)...
Generating key (this may take several minutes)...Key generation complete.
(When selecting a passphrase, keep in mind that good passphrases typically
have upper and lower case letters, digits and punctuation marks, and are
at least 8 characters in length.)
Enter the local keyfile passphrase: localkey
Verify the local keyfile passphrase: localkey
Generating key (this may take several minutes)...Key generation complete.
-------------------------------------------------------
Generating Tripwire configuration file...
-------------------------------------------------------
Creating signed configuration file...
```

```
Please enter your site passphrase:
Wrote configuration file: /etc/tripwire/tw.cfg
A clear-text version of the Tripwire configuration file
/etc/tripwire/twcfg.txt
has been preserved for your inspection.  It is recommended
that you delete this file manually after you have examined it.
---------------------------------------------------------
Customizing default policy file...
---------------------------------------------------------
Creating signed policy file...
Please enter your site passphrase:
Wrote policy file: /etc/tripwire/tw.pol
A clear-text version of the Tripwire policy file
/etc/tripwire/twpol.txt
has been preserved for your inspection.  This implements
a minimal policy, intended only to test essential
Tripwire functionality.  You should edit the policy file
to describe your system, and then use twadmin to generate
a new signed copy of the Tripwire policy.
---------------------------------------------------------
The installation succeeded.
Please refer to /usr/doc/tripwire/Release_Notes
for release information and to the printed user documentation
for further instructions on using Tripwire 2.3 Open Source for LINUX.
```

Durante la propia instalación ha sido necesario utilizar site key para encriptar el fichero con las políticas que utilizará Tripwire durante su ejecución.

Configuración de Tripwire

La configuración de los archivos que se van a monitorizar se mantiene en el archivo de políticas (*policy file*). Su manipulación es algo tediosa debido a su extensión. A modo de ayuda, Tripwire proporciona un archivo de ejemplo que sirve de plantilla para definir nuestras políticas. Este archivo es `/etc/tripwire/twpol.txt`.

Podemos empezar a trabajar sobre este archivo, aunque es recomendable hacer una copia sin modificar del mismo. Hay que hacer una consideración sobre este archivo. Está creado con vistas a comprobar la integridad de todo el sistema, lo que implica que tardará varios minutos (dependiendo de la potencia de nuestro equipo).

Una vez tengamos claro el archivo de políticas que vamos a utilizar, ya sea el que viene como ejemplo o uno basado en modificaciones sobre el mismo, es necesario instalarlo. Tripwire utilizará una versión compilada y encriptada de este archivo, que se almacenará en `/etc/tripwire/tw.pol`. Para generarlo utilizaremos la utilidad `twadmin`:

```
[root@fedora tripwire]# twadmin -m P /etc/tripwire/twpol.txt
Please enter your site passphrase:
Wrote policy file: /etc/tripwire/tw.pol
```

Una vez configurado e instalado el archivo de políticas, Tripwire necesita recolectar la información actual de los archivos a los cuales comprobará su integridad. Dicha información se almacena en una base de datos, que se genera mediante la utilidad `tripwire`:

```
[root@fedora tripwire]# tripwire -m i
```

En caso de que se generen errores, será necesario corregirlos sobre el fichero `twpol.txt`, siendo necesario instalarlo de nuevo con `twadmin`.

Una vez que tengamos Tripwire correctamente configurado con su base de datos, es el momento de verificar la integridad del sistema. Para ello ejecutaremos de nuevo la utilidad `tripwire`:

```
[root@fedora tripwire]# tripwire -m c
```

Utilizaremos la llamada a la utilidad `tripwire` de esta manera cada vez que deseemos saber el estado en que se encuentra nuestro sistema respecto a la integridad del mismo.

Si por algún motivo, alguno de los archivos que estamos comprobando ha sido modificado, será necesario reconstruir la base de datos de Tripwire. Previamente es necesario haber comprobado que la modificación ha sido originada por un proceso controlado y no es consecuencia de una intrusión al sistema.

Una vez comprobado que el archivo `twpol.txt` se adapta a los requerimientos de control de la integridad de nuestro sistema y que hemos alcanzado un estado estable del mismo, es recomendable programar la ejecución de los chequeos de forma automática. Para ello podemos hacer uso del `cron` con una entrada similar a la siguiente:

```
/usr/sbin/tripwire -m c | mail root@localhost
```

Así, la salida del chequeo será enviada por correo a `root`. Esta funcionalidad de envío por correo de los informes puede conseguirse con la configuración de Tripwire. La directiva `email-to=user@host.domain` proporciona esta funcionalidad. Deberá insertarse en la configuración de cada grupo de archivos que vamos a comprobar. Cuando alguno de estos archivos se modifique, Tripwire notificará al destinatario especificado en esta directiva, utilizando para ello Sendmail o Postfix.

Capítulo 3

Análisis Forense

> La exigencia de la seguridad es la parte más atractiva de la administración de sistemas. Hollywood no describe el drama de trasladar una red operativa de un sitio a otro, la instalación del último hardware, la comedia de las preguntas de los usuarios o la tragedia de operaciones de restauración fallidas. Sin embargo, la intriga de la seguridad en las computadoras ha sido objeto de numerosas películas. (*Administración de Sistemas Linux*, M Carling y otros)

Una de las afirmaciones más conocidas en seguridad es la que dice algo así como que el servidor más seguro es aquél que se encuentra desconectado de la red y guardado en una caja de seguridad. Este servidor es también el más inútil ya que nadie tiene acceso a los posibles servicios que ofrece. De aquí se deduce que la seguridad 100 % no existe y siempre, a pesar de nuestro esfuerzo, es posible que un atacante tenga éxito en sus intentos por acceder de forma clandestina a nuestro sistema.

El Análisis Forense de Sistemas (*Computer Forensics*) es el proceso de extracción, conservación, identificación, documentación, interpretación y presentación de las evidencias de un ataque informático. La víctima de un ataque informático es un servidor y es preciso que se compruebe el alcance de la intrusión. No basta con borrar los contenidos y reinstalar de nuevo. En la reinstalación haremos uso de las copias de seguridad existentes en lo referente a ficheros de configuración y puede que éstos hayan sido modificados por el atacante. Es necesario averiguar cuándo se realizó el ataque, en qué consistió el mismo y cual fue la puerta de acceso al sistema. Sin un análisis forense de sistemas se caerá en los mismos errores que provocaron el acceso al sistema.

La metodología básica de un análisis forense consiste en:

1. Adquirir evidencias sin alterar el sistema original. Un primer paso a realizar es aislar el sistema sospechoso, si es posible deteniendo los servicios proporcionados y desconectándolo de la red.

2. Comprobar las evidencias recogidas. Cualquier evidencia que se encuentre debe ser comprobada mediante técnicas de hashing. De esta forma se obtendrá una huella con la que comparar los datos recogidos.

3. Analizar los datos sin modificarlos. Lo ideal es acceder a los discos del servidor desde otro sistema en modo de solo lectura, evitando la posible modificación accidental de algún dato, y realizar una copia de los mismos.

Los pasos para empezar la investigación de un incidente son diferentes en cada caso. El investigador debe tomar decisiones basándose en su experiencia y el "sexto sentido" para llegar al fondo del asunto. No es necesario seguir pasos determinados, ni su orden es importante a veces.

3.1. Recopilando evidencias

Ya se ha visto que el primer paso a la hora de realizar un análisis del sistema comprometido es aislar los datos. Es importante recopilar la información antes de apagar el sistema comprometido. Una vez apagado el ordenador se borrará cualquier evidencia posible de la intrusión (procesos escuchando en determinados puertos o procesos realizando tareas en segundo plano).

Hay que resaltar que no es factible el análisis de un sistema comprometido utilizando las herramientas del mismo, éstas podrían estar infectadas, haber sido modificadas para borrar evidencias. Esto puede solucionarse si se dispone de un dispositivo de almacenamiento extraible (CDROM, disco USB, ...) con el kit de herramientas necesarias para recopilar los datos[1].

Para obtener datos del sistema de archivos, una opción es realizar una copia de los distintos sistemas de archivos para su estudio en otro servidor seguro, en el caso que no sea posible apagar el sistema comprometido. Será necesario disponer de almacenamiento adicional para el volcado de estas imágenes.

Mediante el comando `mount` se obtiene un listado de los sistemas de archivos montados en el servidor y mediante `fdisk` el listado de las particiones existentes en cada unidad de disco, independientemente que estén montadas o no.

Con los datos proporcionados por `mount` y `fdisk` se puede utilizar el comando `dd` para crear una imagen del disco. Este comando realiza imágenes copiando bit a bit los sistemas de archivo. Es recomendable crear una imagen por cada partición del sistema.

```
dd if=/dev/hda1 of=/tmp/hda1.img
```

Una vez generadas las imágenes es conveniente garantizar la autenticidad de las mismas para su posterior verificación. El comando `md5sum` proporcionará una huella de las imágenes para su posterior comprobación.

Para enviar estos datos a otro servidor, en el cual se realizará la investigación, puede utilizarse el comando `netcat`[2]:

```
(host remoto) # nc -l -p 8888 > fichero
(host comprometido) # route -cn | nc guadalinex 8888
```

Con las líneas anteriores el host remoto tiene a `netcat` escuchando por el puerto `8888`, redireccionando todo lo que reciba a `fichero`. Desde el host comprometido se envían las salidas de los comandos necesarios, mediante un pipe, conectando con el host remoto `guadalinex` por el puerto `8888`. En el caso de querer enviar por la red la imagen del sistema de archivos obtenida con `dd`:

```
(host remoto) # nc -l -p 8888 > hda.dd
(host comprometido) # dd if=/dev/hda | nc guadalinex 8888
```

El acceso a las imágenes para analizar su contenido se realizará montando el dispositivo virtual `loop` existente en Linux. Este dispositivo representa una abstracción que permite acceder a imágenes de sistemas de archivos.

```
mkdir /mnt/hda1
mount -t ext2 -o loop -r hda1.img /tmp/hda1
```

Una vez montado se accede al mismo al igual que otro sistema de archivos convencional.

Es importante también obtener datos del estado de la memoria. El que Linux trate como a un fichero la memoria hace esta labor más fácil. La memoria principal se encuentra alojada en `/dev/men` y la memoria de `swap` en la partición correspondiente.

Mediante comandos como `strings` o `grep` puede recorrerse el contenido de la memoria.

```
strings /dev/mem | more
```

[1] Será necesario compilar las herramientas necesarias de forma estática.

[2] Como curiosidad ver la página del manual de netcat (`man nc`) para ver cuál es la definición de esta herramienta.

Visualizando este fichero pueden descubrirse las librerías que hay cargadas en memoria. Al ser la memoria un dispositivo volatil es imposible verificar, como se hizo con los sistemas de archivos, que los datos capturados se corresponden con los originales.

Otro punto interesante es la red. Es necesario comprobar las conexiones de red existentes y los servicios que se encuentran levantados, así como los procesos que los soportan. El comando `netstat` nos ayudará en este aspecto, siendo posible obtener información sobre los procesos asociados con cada conexión específica:

```
netstat -pan | more
```

Dentro del proceso de recopilación de evidencias es muy importante llevar un completo registro (incluyendo fecha y hora) de las evidencias que se vayan descubriendo y los pasos que se están siguiendo. Un método muy sencillo de hacer esto es con el comando `script`, el cual irá almacenando en un fichero todo lo que se teclee en la consola.

```
script -a fichero
```

Para finalizar la captura de datos será necesario teclear `exit`.

3.2. Analizando datos

Hasta ahora se ha conseguido obtener una foto detallada del sistema en el estado en el que se sospecha está comprometido. Es necesario seguir investigando para obtener más datos que ayuden a descubrir la forma en que se produjo la intrusión.

Se empezará comprobando el fichero `/etc/passwd`. Se comprobará detenidamente en busca de usuarios con permisos de root (UID y GID con valor 0) que no debieran tenerlos. Debe buscarse también directorios home de usuarios en localizaciones no habituales (`/tmp` o `/` son muy habituales en estos casos).

Los ficheros dejados por los intrusos como troyanos y similares también suelen localizarse en el directorio `/dev`. Esto hace que sea otro punto en el que deba investigarse, comprobando los ficheros de dispositivos que se hayan creado más recientemente.

Los troyanos que normalmente deja un intruso tienen como objetivo ocultar información sobre el sistema que haga sospechar al administrador que algo va mal, así como puertas traseras que permiten al intruso acceder al sistema cuando quiera.

3.3. Una ayuda al forense: Sleuthkit

Existen también herramientas que nos permiten realizar este análisis de una forma más eficiente. La herramienta Sleuthkit junto con su interfaz web Autopsy (`http://www.sleuthkit.org`) es un buen ejemplo. Sleuthkit es una colección de herramientas de análisis forense, complementarias a las herramientas que proporciona Linux. Permiten examinar el sistema de ficheros sospechoso de una forma no intrusiva. También permiten analizar el medio, soportando el análisis de particiones DOS, BSD, Mac y Sun entre otros.

Cuando se realiza un análisis completo del sistema de ficheros, es de gran ayuda la interfaz gráfica a estas herramientas. El *Autopsy Forensic Browser*, conocido familiarmente como Autopsy, es la interfaz gráfica a las herramientas de Sleuthkit, permitiendo la organización de la información en casos, comprobando la integridad de las imágenes del sistema, búsquedas, etc.

Como características de los datos de entrada que analiza Sleuthkit cabe destacar:

- Sistemas de ficheros raw e imágenes de discos.
- Soporta sistema de ficheros NTFS, FAT, FFS, EXT2FS y EXT3FS.

Dentro de las búsquedas que permite:

- Nombres de ficheros borrados.
- Muestra los contenidos de todos los atributos NTFS.
- Muestra sistemas de ficheros y detalles de la estructura de meta-datos.
- Crea una línea de tiempo de la actividad de los ficheros, con posibilidades de importación y creación de informes.
- Localiza las huellas (hashes) de los ficheros.
- Organiza los ficheros basámdose en su tipo.

Las herramientas que proporciona Sleuthkit pueden dividirse en cuatro categorías:

1. Información de sistemas de archivos completos: fsstat
2. Acceso a datos almacenados en archivos: dcalc, dcat, dls y dstat
3. Información Meta almacenada en inodos: icat, ifind, ils y istat
4. Tareas de nivel archivos: mactime, file y sorter

Para empezar a trabajar con Sleuthkit[3] es necesario tener las imágenes del sistema (obtenidas anteriormente con `dd`). Se montarán estas imágenes como solo lectura, utilizando el dispositivo virtual `loop`.

El primer punto en el que hay que detenerse es el relativo a los archivos borrados, normalmente serán huellas que el intruso ha querido ocultar. El comando `ils` muestra la información del inodo de un archivo del sistema. Si queremos obtener información de los archivos borrados de la partición almacenada en `hda1.dd` de tipo `ext3`:

```
ils -f linux-ext3 -r hda1.dd
```

La salida de la ejecución de `ils` muestra en un formato tabular la información. La línea 1 contendrá la cabecera y la línea 2 los datos correspondientes. Otro comando interesante es `fls` que permite recoger información de cualquier archivo que aún existe.

El conjunto de herramientas proporcionado por Sleuthkit es bastante completo y lo anterior es solo una pequeña introducción. Pueden ser difíciles de usar y el formato en que ofrecen los resultados no ayuda, siendo necesario la ayuda de scripts auxiliares que hagan funciones de filtrado para discriminar información que no interese.

En estas circunstancias el uso de una interfaz gráfica mejora de forma considerable las condiciones de trabajo. Autopsy proporcionará una capa abstracta para los comandos, presentando únicamente los resultados, que realmente es lo que interesa. Proporciona también una herramienta con la que documentar y comentar los datos y resultados.

Autopsy permite abrir tantos casos como deseemos, permitiéndonos así almacenar información de varios incidentes. Se creará un directorio por cada caso, que servirá como almacén de los resultados y datos que vayan extrayéndose de las imágenes.

Desde Autopsy se puede realizar también la gestión de las imágenes que dispongamos, almacenándolas en el directorio indicado anteriormente para su estudio o creando un enlace simbólico

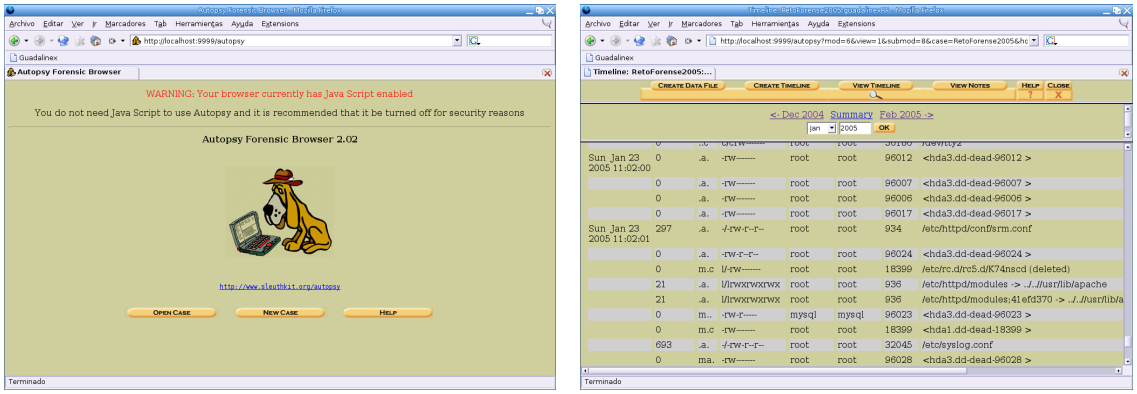
[3]Existen paquetes para la versión inestable de Debian. Instalar paquetes de esa versión puede hacer que nuestro sistema se vuelva inestable y debemos hacerlo con sumo cuidado y bajo nuestra responsabilidad. Si aún así optamos por instalarlos usando el comando `apt-get`, antes hemos de modificar nuestro fichero `/etc/apt/sources.list` añadiendo la línea

```
deb http://ftp.fi.debian.org/debian sid main contrib non-free
```

y después

```
#apt-get update
#apt-get install sleuthkit autopsy
```

Figura 3.1: Pantalla de inicio de Autopsy y análisis de datos



a las imágenes, también con origen en el directorio del caso. El procedimiento a seguir será añadir todas las imágenes indicando el punto de montaje correspondiente, el tipo de sistema de ficheros que utiliza y si se desea que se calcule el MD5 de cada imagen. Esto último nos permite comprobar la integridad de las imágenes.

Una vez acabado este proceso de creación del caso puede comenzar la investigación. Inicialmente puede hacerse una recogida de datos de la línea de tiempo de los cambios producidos en el sistema.

Sleuthkit es una herramienta muy potente y es muy utilizada en los análisis forenses realizados tanto sobre sistemas de archivos Linux como Windows. Sin embargo, estas herramientas son sólo ayudas para la persona que realiza el análisis. Por sí solas no van a decirnos qué ha pasado en el sistema. Será el encargado de realizar el análisis forense el que con su experiencia y conocimiento del sistema deduzca qué es lo que ha pasado.

Existen concursos en los que se publican las imágenes de un sistema comprometido y se propone su estudio y análisis[4]. Los concursantes presentarán sus conclusiones así como los pasos y herramientas utilizadas en el proceso. Si se tiene interés en profundizar sobre este tema es interesante empezar con una de estas imágenes publicadas e intentar relizar el análisis. Una vez finalizado es posible compararlo con las conclusiones obtenidas por los concursantas, descubriendo así los puntos que se hayan pasado por alto y las técnicas que siguen distintas personas.

[4]Pueden encontrarse ficheros de desafío forense en `http://project.honeynet.org/challenge/images.html`

Capítulo 4

Detección de virus

> Un virus es una secuencia de código que se inserta en un fichero ejecutable denominado host, de forma que al ejecutar el programa también se ejecuta el virus; generalmente esta ejecución implica la copia del código viral, o una modificación del mismo, en otros programas
>
> *Seguridad en Unix y Redes.* Antonio Villalón Huerta
>
> virus.(Del lat. virus).
>
> 1. m. Biol. Organismo de estructura muy sencilla, compuesto de proteínas y ácidos nucleicos, y capaz de reproducirse solo en el seno de células vivas específicas, utilizando su metabolismo.
>
> 2. m. Inform. Programa introducido subrepticiamente en la memoria de un ordenador que, al activarse, destruye total o parcialmente la información almacenada.
>
> *DICCIONARIO DE LA LENGUA ESPAÑOLA (Vigésima segunda edición) Real Academia Española*

4.1. Virus y Troyanos en UNIX

Podemos cometer errores o descuidos a la hora de configurar nuestro sistema o de programar una aplicación. Esta es la fuente principal de los problemas de seguridad. Sin embargo, existen otras fuentes, los denominados *malware* o software malicioso. Son programas o `scripts` creados con el objetivo concreto de dañar un sistema o lograr información privilegiada del mismo.

Es importante que nos concienciemos de ejecutar en nuestro sistema únicamente software del cual tengamos la certeza que proviene de una fuente fiable. Ésta consideración es especialmente importante cuando hablamos de la administración del sistema. Muchas labores de administración se realizan desde el usuario `root`, que posee privilegios para realizar cualquier acción en el sistema[1].

Supongamos que encontramos en internet un programa que dice cambiar la configuración gráfica de nuestro sistema de X-Window. En la página donde nos podemos descargar este software nos habla de las excelencias del programa, lo fácil de utilizar que es y nos muestra varias copias de pantalla. Nos descargamos el paquete y en las instrucciones nos indica que ejecutemos el *script* `install.sh` con el usuario `root`. Confiamos ciegamente en él y lo ejecutamos. Cual es nuestra sorpresa cuando vemos cómo se empiezan a borrar todos los archivos de nuestro sistema porque hay una línea en este *script* que ejecuta `cd /; rm -rf *`. Este es un caso extremo, que no tiene porqué pasar ¿verdad?, pero nos advierte de comprobar siempre las fuentes de nuestros programas, especialmente si los bajamos de páginas *underground*[2].

Si un usuario, que no sea `root`, ejecuta un programa que contiene un virus o un troyano, únicamente afectará a los archivos a los que el usuario tiene permiso de escritura o modificación. Como acabamos de comprobar, esto se debe aplicar con más rigor en el caso del administrador del

[1] Aunque esto cambiará con mecanismos como SELinux, por ejemplo.
[2] Los bajos fondos de Internet.

sistema. Si es `root` el que ejecuta el programa contaminado, cualquier archivo del sistema puede contagiarse.

Además de descargar el software de fuentes fiables, es recomendable utilizar las "huellas" de todos los programas, generalmente en forma de MD5[3] para verificar que hemos bajado el archivo legítimo. Igualmente, también tenemos la opción, aunque más ardua, de descargar el código fuente y compilar nosotros mismos los programas. Lograremos así la posibilidad de revisar el código fuente en busca de potenciales problemas de seguridad.

4.1.1. El problema de los virus

De forma similar a los virus que atacan nuestro cuerpo y nos provocan enfermedades, los virus informáticos que atacan nuestros ordenadores, nos pueden producir "dolores de cabeza" y graves problemas[4]. Los virus informáticos poseen múltiples vías de transmisión (como la picadura de los mosquitos o los estornudos en los virus biológicos), aprovechando cualquier descuido o resquicio de seguridad tanto del sistema operativo, los programas de aplicación o el propio usuario. Hay páginas web que incluyen virus, programas que descargamos, incluso si no aplicamos los parches de seguridad para mantener nuestro sistema actualizado, algunos intrusos como el Sasser pueden introducirse en él. Pero, sin duda, la difusión de virus a través del correo electrónico debe llevarse todos los honores como principal vía de contagio.

La mejor forma de proteger nuestra red frente a estos correos con virus, es que ni siquiera lleguen a entrar. El que nuestro servidor de correo disponga un mecanismo de detección y eliminación de virus nos protegerá de infinidad de peligros.

Para ello utilizaremos ClamAV, que es un detector de virus con licencia GPL y que integraremos con nuestro agente de transporte de correo para rechazar los mensajes con virus.

El propósito principal de este software es tanto la integración con los servidores de correo (escaneo de datos adjuntos) como el escaneo de sistemas de ficheros que puedan contener virus (p. ej. un servidor samba para clientes windows).

En un sistema antivirus es muy importante la actualización de los ficheros de firmas[5] y de los motores antivirus para adaptarse a las mutaciones y apariciones de nuevos virus. ClamAV dispone de una herramienta para actualizarse automáticamente desde Internet.

El sitio principal de este antivirus es `http://www.clamav.net`, donde podremos ampliar información.

4.2. Antivirus ClamAV

4.2.1. Instalación

Debian

Para instalarlo, usaremos `apt-get`, ejecutándolo como `root`. En esta ocasión suponemos que se ha instalado el servidor de correo Postfix junto con amavis, como se explicó en la tercera entrega de este curso.

En primer lugar necesitamos instalar, en nuestro caso casi todos estarán y, a lo sumo se actualizarán, los paquetes relacionados con los ficheros adjuntos, a fin de poder inspeccionarlos:

```
root@guadalinex:/home/mowgli# apt-get install unrar lha arj unzoo zip unzip
    bzip2 gzip cpio file lzop
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
unrar ya áest en su óversin áms reciente.
```

[3] Fácilmente con $md5sum fichero, podemos obtener la huella y comprobarla con la auténtica, para comprobar que no ha sido modificado por los malos.

[4] Los efectos de los virus informáticos pueden ir desde la simple broma o reivindicación, a la filtración de datos confidenciales o la destrucción de datos en el ordenador.

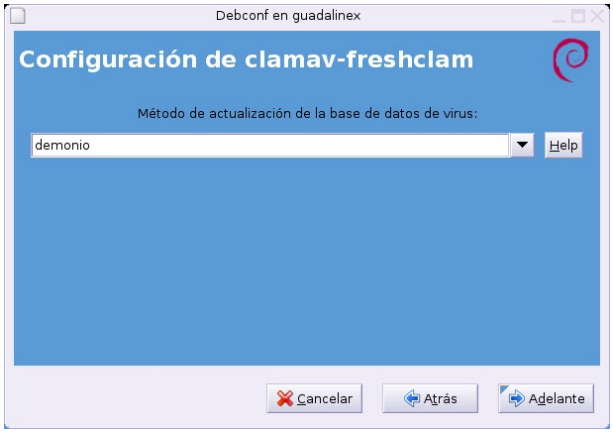
[5] La huella para detectar al virus.

```
unzip ya áest en su óversin áms reciente.
bzip2 ya áest en su óversin áms reciente.
gzip ya áest en su óversin áms reciente.
cpio ya áest en su óversin áms reciente.
Se áinstalarn los siguientes paquetes extras:
libmagic1
Se áinstalarn los siguientes paquetes NUEVOS:
arj lha lzop unzoo
Se áactualizarn los siguientes paquetes:
file libmagic1 zip
3 actualizados, 4 se áinstalarn, 0 para eliminar y 563 no actualizados.
Necesito descargar 680kB de archivos.
Se áutilizarn 984kB de espacio de disco adicional édespus de desempaquetar.¿
Desea continuar? [S/n] S
```

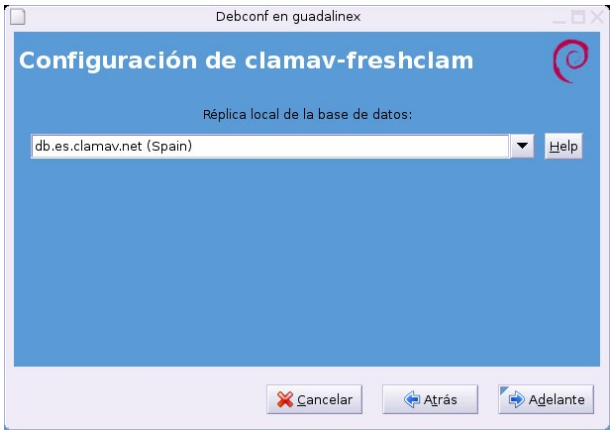
A continuación, instalaremos los paquetes propios de Clam Antivirus:

```
root@guadalinex:/home/mowgli# apt-get install clamav clamav-base clamav-
    daemon clamav-freshclam libclamav1
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Se áinstalarn los siguientes paquetes extras:
libcurl3 libgmp3
Paquetes sugeridos:
daemon libcurl3-gssapi ca-certificates
Se áinstalarn los siguientes paquetes NUEVOS:
clamav clamav-base clamav-daemon clamav-freshclam libclamav1 libgmp3
Se áactualizarn los siguientes paquetes:
libcurl3
1 actualizados, 6 se áinstalarn, 0 para eliminar y 562 no actualizados.
Necesito descargar 3069kB de archivos.
Se áutilizarn 4592kB de espacio de disco adicional édespus de desempaquetar.¿
Desea continuar? [S/n] S
```

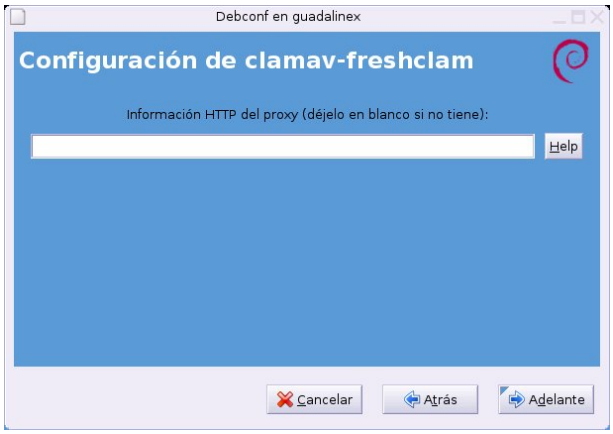
En este momento se inicia de manera automática el asistente de configuración de Clam Antivirus. Seleccionaremos como método de actualización de la base de datos de virus el método de "demonio" (*daemon*) y pulsaremos Adelante:



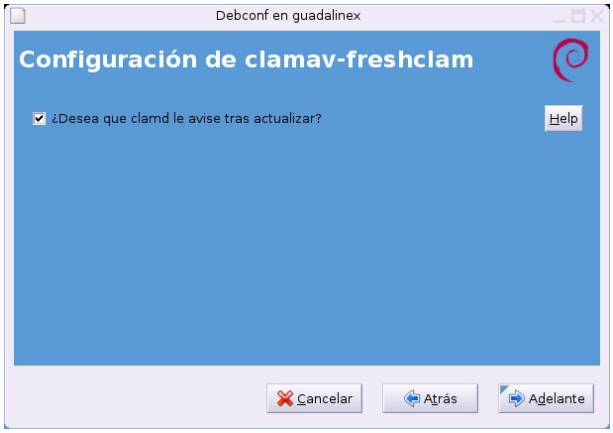
Como *mirror* para la descarga seleccionamos el más cercano a la localización geográfica de nuestro servidor, y pulsamos Adelante:



Si tenemos que usar un proxy para conectarnos a Internet deberemos configurarlo en esta pantalla, o nada con conexión directa a Internet y pulsamos Adelante:



Activamos la opción para que se nos avise tras la actualización y pulsamos **Adelante**, la instalación de los paquetes continuará por el siguiente punto:



Se inicia la instalación y configuración de los paquetes y se añade el usuario y grupo clamav. Por último, se inicia el demonio `clamd`.

Una vez concluida la instalación[6], no es preciso modificar ningún fichero de configuración, pues la instalación ya nos deja todo lo que necesitamos funcionando adecuadamente. Podemos mirar en el fichero `/etc/amavis/amavisd.conf`.

[6]Si todavía los paquetes Debian no se han actualizado a la última versión, puede salir un mensaje de que no es la versión más actual. Esperaremos a que se generen los paquetes y actualizamos con `apt-get`.

El modo de funcionamiento será el siguiente: primero Postfix, nuestro agente de transporte, recibirá los correos y a continuación se los pasará a Amavis para que detecte si contiene virus (dándole el trabajo a Clamav) o es un spam (en este caso el currito será Spamassassin).

Si observamos el siguiente error en el `/var/log/mail.log`:

```
linux amavis[13147]: (13147-01) Clam Antivirus-clamd FAILED - unknown status
    : /var/lib/amavis/amavis-20040818T163812-13147/parts: Access denied.
    ERROR\n
linux amavis[13147]: (13147-01) WARN: all primary virus scanners failed,
    considering backups
```

Entonces, deberemos añadir el usuario `clamav` al grupo `amavis`, una vez que se haya creado, de la siguiente forma:

```
#adduser clamav amavis
```

Y comprobamos que la directiva `AllowSupplementaryGroups` se encuentra en el fichero `/etc/-clamav/clamd.conf`.

Reiniciamos el servidor de correo `postfix`, `amavis` y `clamav`, o el sistema completo.

4.2.2. Probemos la medicina

Una vez instalados los tres paquetes, comprobemos que funciona correctamente.

Primero, con el comando `clamscan`, comprobamos que los ficheros o directorios que le especificamos, están libres de virus o puede que encontremos alguna sorpresa.

```
root@guadalinex:~/# clamscan -r -i -l vir.txt /root/
LibClamAV Warning: ***********************************************************
LibClamAV Warning: *** This version of the ClamAV engine is outdated. ***
LibClamAV Warning: *** DON'T PANIC! Read http://www.clamav.net/faq.html ***
LibClamAV Warning: ***********************************************************
/root/.mozilla/default/0jiyr4ub.slt/Mail/www.midominio.org/Sent: Worm.Gibe.F
    FOUND
/root/Desktop/Descargas/hsgejoq(1).exe: Worm.Gibe.F FOUND
/root/Desktop/Descargas/hsgejoq.exe: Worm.Gibe.F FOUND
/root/hsgejoq.exe: Worm.Gibe.F FOUND
---------- SCAN SUMMARY -----------
Known viruses: 34867
Engine version: 0.84
Scanned directories: 257
Scanned files: 1257
Infected files: 4
Data scanned: 57.43 MB
Time: 70.929 sec (1 m 10 s)
```

El comando anterior (`clamscan`) busca recursivamente (`-r`) a partir del directorio o fichero que le especificamos en último lugar (en este caso desde el home de `root` `/root`).

La opción `-i` indica que solamente nos muestra los ficheros infectados y la opción `-l` nos indica que guardará un informe en el fichero `resultado-virus.txt`. Puede que detectéis algún intruso en vuestro sistema, especialmente si compartís con sistemas windows vía Samba algún directorio.

Otro ejemplo de ejecución es el siguiente:

```
[root@linux tmp]# clamscan -i
/root/tmp/clamav-0.71.tar.gz: ClamAV-Test-Signature FOUND
/root/tmp/msg27986.zip: Worm.SomeFool.Q FOUND
---------- SCAN SUMMARY -----------
Known viruses: 21635
```

```
Scanned directories: 1
Scanned files: 19
Infected files: 2
Data scanned: 3.89 MB
I/O buffer size: 131072 bytes
Time: 6.577 sec (0 m 6 s)
```

En este caso ha detectado dos ficheros con virus y nos presenta el informe final.

También podemos utilizar el comando `clamdscan`.

```
root@guadalinex:~/curso-linux/entrega6/images# clamdscan /root/hsgejoq.exe
/root/hsgejoq.exe: Worm.Gibe.F FOUND
----------- SCAN SUMMARY -----------
Infected files: 1
Time: 0.152 sec (0 m 0 s)
```

4.2.3. Freshclam

El proceso `freshclam` se encarga de mantener actualizado el fichero de firma de virus. Podemos ejecutarlo directamente desde la línea de comandos, aunque lo mejor es incluirlo como tarea periódica (en el sistema `cron`) para que se actualice automáticamente.

El siguiente comando comprueba los ficheros de firmas de virus, y nos da como resultado que estamos actualizados.

```
[root@linux tmp]# freshclam
ClamAV update process started at Sat May 22 14:54:04 2004
Reading CVD header (main.cvd): OK
main.cvd is up to date (version: 23, sigs: 21096, f-level: 2, builder: ddm)
Reading CVD header (daily.cvd): OK
daily.cvd is up to date (version: 325, sigs: 539, f-level: 2, builder:
    ccordes)
```

En la dirección `http://news.gmane.org/gmane.comp.security.virus.clamav.virusdb`, podemos ver las versiones de los ficheros de firma de virus y si son los que nosotros tenemos.

En esta ejecución se produce una actualización del fichero `daily.cvd` a la versión 326.

```
ClamAV update process started at Sun May 23 11:02:31 2004
Reading CVD header (main.cvd): OK
main.cvd is up to date (version: 23, sigs: 21096, f-level: 2, builder: ddm)
Reading CVD header (daily.cvd): OK
Downloading daily.cvd [*]
daily.cvd updated (version: 326, sigs: 554, f-level: 2, builder: ccordes)
Database updated (21650 signatures) from database.clamav.net (80.69.67.3).
Clamd successfully notified about the update.
```

4.2.4. Funcionamiento

Para comprobar que está funcionando, mandemos un correo de prueba al usuario `alumno1@midominio.org`[7] con nuestro cliente de correo preferido. Éste es el resultado del correo una vez entregado, mostrando las cabeceras.

```
From alumno1@midominio.org Tue May 31 01:30:20 2005
Return-Path: <alumno1@midominio.org>
X-Original-To: alumno1@midominio.org
```

[7] O el dominio que tengamos configurado en nuestro sistema.

```
Delivered-To: alumno1@midominio.org
Received: from localhost (guadalinex [127.0.0.1])
by guadalinex (Postfix) with ESMTP id 8AA77A781
for <alumno1@midominio.org>; Tue, 31 May 2005 01:30:07 +0200 (CEST)
Received: from guadalinex ([127.0.0.1])
by localhost (www.midominio.org [127.0.0.1]) (amavisd-new, port 10024)
with LMTP id 08847-01 for <alumno1@midominio.org>;
Tue, 31 May 2005 01:28:59 +0200 (CEST)
Received: from [192.168.200.4] (unknown [192.168.200.4])
by guadalinex (Postfix) with ESMTP id 672DE97A8
for <alumno1@midominio.org>; Tue, 31 May 2005 00:35:39 +0200 (CEST)
Message-ID: <429B953A.7070901@midominio.org>
Date: Tue, 31 May 2005 00:35:38 +0200
From: alumno1 <alumno1@midominio.org>
User-Agent: Mozilla/5.0 (X11; U; Linux i686; es-ES; rv:1.7.6) Gecko/20050324
    Debian/1.7.6-1
X-Accept-Language: en
MIME-Version: 1.0
To: alumno1 <alumno1@midominio.org>
Subject: Re: df
References: <429B9356.6040009@midominio.org>
In-Reply-To: <429B9356.6040009@midominio.org>
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: 8bit
X-Virus-Scanned: by amavisd-new-20030616-p10 (Debian) at midominio.org
Status: O
X-UID: 19
Content-Length: 27
X-Keywords:
alumno1 óescribi:
> Hola
```

Comentemos las líneas más interesantes. Primero el camino recorrido por el correo:

```
Received: from [192.168.200.4] (unknown [192.168.200.4])
by guadalinex (Postfix) with ESMTP id 672DE97A8
```

Estas líneas indican que la primera escala ha sido la recogida por parte de Postfix como agente de transporte de correo (puerto `25` de la máquina `192.168.200.4`)

```
Received: from guadalinex ([127.0.0.1])
by localhost (www.midominio.org [127.0.0.1]) (amavisd-new, port 10024)
with LMTP id 08847-01 for <alumno1@midominio.org>;
```

Postfix de nuestra máquina local (`127.0.0.1`) lo ha pasado al puerto `10024` que es donde está escuchando `amavisd-new`. Aquí se aplican las reglas especificadas en el fichero de configuración `/etc/amavis/amavisd.conf`.

```
Received: from localhost (guadalinex [127.0.0.1])
by guadalinex (Postfix) with ESMTP id 8AA77A781
for <alumno1@midominio.org>; Tue, 31 May 2005 01:30:07 +0200 (CEST)
```

Si nuestro valeroso correo pasa todas las pruebas de virus, spam y banned, llegará exitoso a los brazos de la persona destinataria[8].

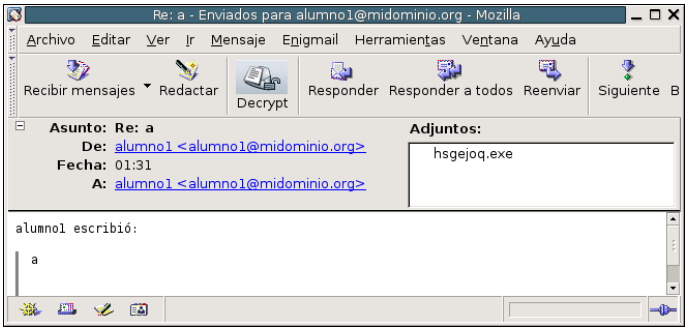
```
X-Virus-Scanned: by amavisd-new-20030616-p10 (Debian) at midominio.org
```

[8]Quién sabe si es que nos ha llegado la devolución de Hacienda. Para despidos o marrones, podemos aplicar algún filtrillo en Amavis ;-)

Nos dice que el correo ha sido escaneado y comprobado por `amavisd-new`.

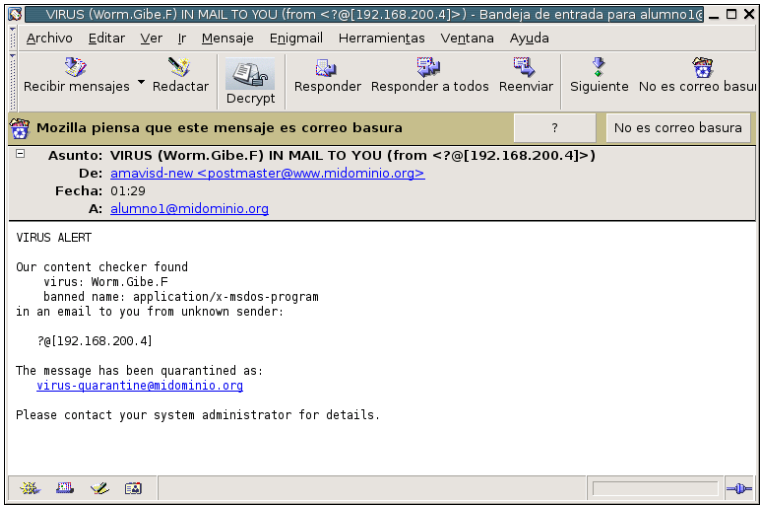
4.2.5. Ejemplo

Comprobemos cómo funciona enviando un virus de verdad. No os debe ser muy difícil capturar alguno, ¿verdad?. De un correo cogemos el fichero `hsgjoq.exe`, que contiene un virus. Lo guardamos y vamos a incluirlo como adjunto en un correo a nuestro sistema.



Lo enviamos y al usuario `postmaster` que esté definido en nuestro sistema le llegará un mensaje de aviso. Podemos personalizar qué hacer cuando se detecta un virus. Lo recomendable es que sólo se avise al `postmaster`, porque ayudará a ver de dónde procede el virus y desinfectar la máquina responsable. Al destinatario normalmente no le interesa saber que un virus iba dirigido a él (hay tantos). El supuesto remitente normalmente es engañoso, el virus se encarga de poner el remitente que le interese para no despertar sospechas y lo que conseguimos es que una persona que no tiene nada que ver, reciba un mensaje diciendo que ha enviado un virus, siendo falso en la mayoría de los casos.

En caso de que configuremos para que nos avise si un virus iba dirigido a nosotros, recibiremos un correo como éste.



Bibliografía

[1] *Seguridad en Unix y redes v2.0* Antonio Villalón Huerta

[2] *Seguridad práctica en Unix e Internet* Garfinkel y Spafford

[3] *Hacking Exposed Linux* Brian Hatch y James Lee

[4] Información de la NSA sobre SELinux en http://www.nsa.gov/selinux/

[5] HOWTO - Getting Started with new SELinux