Recomendaciones de seguridad en una instalación de Linux

Gunnar Wolf
Departamento de Seguridad en Cómputo
FES Iztacala

CONSOL 2002

Resumen

Linux es un sistema operativo muy popular, y que día a día gana adeptos. Su fuerte principal está en los servidores, si bien está también ganando más y más fuerza en el escritorio. Su mayor problema, sin embargo, es el enfoque que las principales compañías que lo producen le están dando: Un sistema poderoso, estable y tremendamente fácil de usar. La facilidad de uso conlleva que la seguridad haya sido relegada por debajo de lo que debería ser. Lo que busco con este texto, escrito para el seminario admin-unam, es dar un par de consejos básicos para hacer y mantener una instalación de Linux razonablemente segura.

Este texto completo y actualizado estará disponible en la dirección http://www.gwolf.cx/seguridad/recom_seg_linux

Índice General

0.1 Pre-introducción

El que diga saber todo acerca de Linux es una persona en la que no hay que confiar. Este es un trabajo no muy maduro. He de confesarlo —No muy maduro y hecho a las carreras. Si tienes cualquier corrección, modificación, agregado o lo que sea... Con mucho gusto, y por favor.

Qué aprenderemos?

Primero que nada, en qué podemos confiar y en qué no cuando hablamos de Linux. Platicaremos un poco acerca del modelo de desarrollo —alabado y criticado— conocido como el bazar, y a lo que este ha llevado. Un repaso acerca de las distribuciones, y de los mecanismos que tenemos a nuestro alcance para asegurar a nuestro sistema

Quién debería estar aqui?

Administradores y usuarios en general de sistemas Linux con ya cierta experiencia, con conocimiento de la operación y configuración básica, con inquietud acerca de la seguridad de sus sistemas

Qué es lo que no aprenderás aqui?

El material que te presento no te enseñará a profundidad ninguna de las acciones que comentaremos. Te presentará, sí, la forma básica en la que podrás hacerlo, pero Gunnar cree plenamente en la filosofía RTFM, y juzga imposible entrar en detalles acerca de cada detalle de la configuración. Gunnar cree plenamente que si te interesa un tema, no hay mejor recurso que el aprenderlo por tí mismo.

¿Más recursos?

Claro, no puedo pretender que este tutorial lo incluya todo. Van de una vez varios recursos importantes. Me basé en varios de ellos para hacer este trabajo.

http://www.securityintruder.com/curso.php Curso impartido por Jaime Sánchez en la Universidad Carlos III en Madrid

http://www.seguridad.unam.mx/Tutoriales/tutoriales.html Tutoriales del Departamento de Seguridad en Cómputo

http://www.openbsd.org, del mundo.

http://www.securityfocus.org/, Leyendo artículos en general de los reconocidos grandes de la seguridad siempre trae nuevas ideas.

Tal vez varios de ellos no apliquen directamente a Linux, pero saber de seguridad en general siempre nos mantendrá al día y listos con la mejor información.

1 Introducción

Por fín te convencieron. Linux es fácil. Linux es potente. Linux es grandioso. Linux es la diva bañada en yema de huevo. Te decides a instalarlo. Afortunadamente, a diferencia de los miles de chamaquitos que están dando una terrible fama de inseguro a este sistema operativo, a tí te importa la seguridad – Felicidades. Pero... ¿Qué vas a hacer al respecto? Tendrás que comenzar desde abajo.

1.1 Elige tu Linux

Hay varias distribuciones de Linux, y entre sí pueden ser tan diferentes como dos Unixes sin relación entre sí. Afortunadamente, han mostrado una tendencia a converger sobre un número mucho menor de familias:

Redhateños Casi todas las distribuciones hoy en día están basadas en RedHat [1] – LinuxPPP [2], Conectiva [3], Hispafuentes [4], Mandrake [5], TurboLinux [6], y en cierta medida SuSE [7] y Caldera [8] juegan con las mismas reglas. Todos estos sistemas se centran en una instalación sencilla; todos ellos ofrecen ayuda gráfica para la configuración del sistema por medio de linuxconf, drakconf, yast o algún equivalente. La selección de paquetes se puede hacer por categorías o por paquetes individuales. Hay estilos de configuración pre-establecidos, reduciendo el proceso de instalación a un par de teclazos – o más aún, de clicks de mouse. RedHat y sus derivados son, y por mucho, líderes en el mercado. Últimamente se han orientado cada vez más a usuarios novatos, descuidando importantes aspectos de seguridad a favor de facilidad de uso. Mucha gente indica como recomendable para un usuario nuevo iniciar con una de estas distribuciones, pues son para las que más fácilmente encontrará soporte. Manejan el sistema de paquetes RPM (RedHat Package Manager), que evita que se rompan dependencias y es un muy buen auxiliar para mantener al sistema en un estado consistente. Con una cantidad razonable de esfuerzo, es posible cerrar las principales vulnerabilidades de una de estas distribuciones, logrando una instalación mucho más segura que la que tienen de fábrica.

Debiánidos Debian [9] nace como un proyecto comunitario, fuertemente basado en la ideología de la Free Software Foundation [10]. Tienen un fuerte contrato social [11], en el que se comprometen con la comunidad de software libre, y en el cual está basada la Open Source Definition [12], escrita originalmente por Bruce Perens. Debian es desarrollado y mantenido por una comunidad, no por una empresa. Esto hace que no tenga presiones para sacar versiones (su versión estable actual es la 2.2r3, comparado con las 7.x u 8.x de todos los demás); los sistemas Debian son realmente robustos, los paquetes que aparecen en su rama estable han sido ampliamente probados; el sistema de paquetes de Debian (.deb, manejado con dpkg y apt-get, y varios front-ends como dselect y aptitude) es el más avanzado

en el mundo de Linux; actualizar un sistema Debian completo puede hacerse con sólamente dos líneas de apt-get; la resolución de dependencias y conflictos es muy superior inclusive a la de RPM. sin embargo, en aras de la madurez técnica, se ha sacrificado el lado de la interfaz de usuario, dando como resultado un sistema que a entender de muchos no es muy apto para principiantes.

Slackwaretas Cada día menos comunes, pero con una base de leales usuarios. Slackware [13] es la más veterana de las distribuciones originales de Linux que se siguen manteniendo – SLS e Yggdrasil, los verdaderos pioneros, desaparecieron tras cierto tiempo. Slackware se mantiene fiel a su clientela, que busca un Unix tradicional. Si no me equivoco (corríjanme, por favor!) Slackware es la única distribución de Linux hoy en día que utiliza un sistema de arranque tipo BSD [14] (contrastando con el arranque tipo SysVR4). Slackware no utiliza un sistema de paquetes como los .deb o los .rpm, sino que paquetes .tgz al estilo de los Unixes tradicionales, con un muy débil manejo de versiones, por lo que actualizar normalmente implica recompilar, y desinstalar un paquete puede desencadenar una desagradable cadena de dependencias no resueltas – O peor aún, al intentar actualizar algo puedes, sin darte cuenta, terminar instalando versiones previas. Sin embargo, bien administrado tiene fama de ser muy robusto y estable. Slackware cuenta con una base de usuarios muy limitada, y generalmente son usuarios relativamente experto. Lo que veremos en este tutorial puede a grandes rasgos aplicarse también a sistemas Slackware —de hecho, a cualquier sistema Unix— pero no haremos mayor referencia a esta familia explícitamente.

Minimalistas En Linux existe una gran cantidad de *minidistribuciones*, conjuntos pequeños de programas con un propósito específico hechos para computadoras de propósito específico (p. ej. ruteadores o firewalls), máquinas con prestaciones muy reducidas, o para revivir sistemas dañados. Algunas de las más populares son muLinux [15] (distribución en varios floppies que busca dar lo más cercano a una distribución completa de Linux en floppies y sin necesidad de disco duro), floppyFW [16] (Un sencillo ruteador con capacidades básicas de firewall en un sólo floppie), Linux Router Project [17] (Otro proyecto que busca crear un ruteador. Más extensible que floppyFW), Trinux [18] (Juego extensible de herramientas de seguridad en floppies), Hal91 [19] (Distribución minimalista, muy utilizada para construir aplicaciones específicas sobre de ella), LODS [20] (Derivado de Hal91, incluye un VNCViewer sobre svgalib para tener siempre a la mano una estación de manejo remoto gráfico) y tomsrtbt [21] (Las herramientas básicas para la recuperación de un sistema dañado). Y si bien dista de ser minimalista, en esta categoría también cabe Demolinux [22] (una instalación completa de Linux, con X, GNOME, KDE y StarOffice, en un CD, para poderlo usar sin tener que instalar una computadora completa). Cada una de estas minidistribuciones está hecha para cubrir su nicho específico, y no las tocaremos más en este tutorial.

1.1.1 Recomendación personal del autor

Si bien elegir una distribución es una labor que debe basarse en las necesidades, la experiencia y la opinión personal de cada uno de ustedes, se me haría muy difícil no dar mi punto de vista al respecto.

Mi recomendación personal, tanto para usuarios experimentados como para usuarios novatos, es definitivamente Debian. Para ambos habrá una curva de aprendizaje; para un novato, esta curva puede ser algo más empinada que para un experto, pero una vez conociendo la manera Debian de hacer las cosas, continuar con la administración del sistema es increíblemente más simple que en las basadas en RedHat.

Debian mantiene tres *ramas* simultáneas de versiones — La estable, la de pruebas y la inestable, cada una de ellas con un nombre código. Actualmente (fines del 2001), las ramas son *Potato* (Stable), *Woody* (Testing) y *Sid* (Unstable). Al ser Debian una distribución mantenida por la comunidad y que busca el máximo de estabilidad y robustez, la rama estable es la indicada pra los servidores y demás máquinas críticas. Los paquetes que incluye no son los últimos disponibles, pero han sido ampliamente probados y no contienen ningún defecto importante que ponga en riesgo la estabilidad o entegridad del sistema. *Potato* fue congelado en agosto del 2000, y desde entonces sólo se le han añadido actualizaciones de seguridad.

La rama de pruebas —Woody— está aún en desarollo, y se espera que sea congelada para convertirse en la próxima rama estable. Cuándo? No está aún determinado, pero probablemente hacia mediados del 2002. La respuesta que escucharán de cualquier desarrollador de Debian es *cuando sea su tiempo.* Woody puede ser ideal para escritorios o estaciones de trabajo. Dados los altos estándares de confiabilidad de Debian, yo no dudaría en recomendar a esta versión como tan estable como una versión definitiva de RedHat, y menos aún en la fase de desarrollo en que ahora estamos, en que poco a poco se va congelando Woody para reemplazar a Potato.

Por último, *Sid* es la rama para los usuarios que gustan estar al último grito en el desarrollo. Las últimas versiones los programas están siempre disponibles a sólo días de haber sido liberadas. Claro, el precio que hay que pagar a cambio es el saber que a veces algo puede tronar, lao puede desestabilizar al sistema entero o a algún subsistema.

Yo corro en mi escritorio con Sid, y muy rara vez he tenido que pelearme con él. Sin embargo, les sugiero fuertemente utilizar —al menos en un principio— sólo Potato o Woody (o las ramas que en determinado momento sean *stable* o *testing*). Un paquete nuevo es admitido rápidamente en Sid, y una vez que ha demostrado no tener problemas importantes es enviado a Woody. Al finalizar el ciclo de congelación de Woody y éste sea nombrado estable nuestro paquete entrará en la rama estable. De otro modo, a menos que sea un parche importante de seguridad o funcionalidad, no entrará a estable.

Todo este rollo respecto a las ramas de Debian es para hacerte ver lo seriamente que toman en cuenta la estabilidad — En Debian jamás habrá versiones estables oficiales tan defectuosas como la RedHat 7.0 o 7.1.

Un detalle más: Si lo que buscas es instalar un sistema operativo *verdaderamente* seguro, Linux no es el camino a seguir. Una distribución y una instalación de Linux pueden ser más o menos seguras, pero nunca serán realmente seguras. OpenBSD es lo que buscas. Un sistema operativo tipo Unix, libre (licencia BSD), diseñado con la seguridad y estabilidad como primer objetivo. Si quieres más información acerca de este sistema operativo, entra a la página oficial del proyecto OpenBSD [23] o a la página del grupo de usuarios de OpenBSD de México [24]. OpenBSD *no* es Linux, por lo que no lo seguiremos tratando en este tutorial.

Dicho lo cual, continuemos.

2 Una instalación segura inicia con una buena instalación

La instalación es lo primero para llevar a cabo el aseguramiento de una computadora. Si tenemos una computadora corriendo desde hace algún tiempo y no la hemos asegurado, probablemente sea más sencillo respaldar, formatear, reinstalar y asegurar antes de que pase nada a intentar asegurarla — buscar una posible intrusión, una puerta trasera oculta en el sistema, alguna configuración modificada puede ser un trabajo demasiado complejo. Es práctica muy recomendable el no conectar la computadora a red recién terminada la instalación, sino que sólo hacerlo tras haberla asegurado.

2.1 Instalando el sistema

Antes de instalar un sistema, el primer paso es analizar detalladamente para qué lo vamos a querer; qué servicios va a prestar, qué paquetes necesitamos para proveer dichos servicios, qué deseamos que pueda hacer. Empezar por el lado contrario (qué no quiero tener en el sistema) no lleva a ningún resultado real, aunque no niego es útil tener identificados los paquetes en los que no confiamos (¿ejemplos clásicos? wu-ftpd, imapd, fingerd, telnetd... Asómate a 2.2.2 para una mayor explicación) para no instalarlos ni por error, y hasta tener una reacción visceral al sólo verlos. Van un par de puntos relativos a la instalación de RedHat y de Debian.

2.1.1 RedHat

Es muy diferente instalar una estación de trabajo e instalar un servidor. RedHat ofrece varias opciones predeterminadas de instalación rápida: Estación de trabajo GNOME, estación de trabajo KDE, servidor, o personalizada. Te sugiero fuertemente *nunca* elegir una opción predeterminada. Hacerlo equivale a solicitarle que instale todo lo que quien preparó la distribución cree que puede valer la pena — tanto para servidor como para estación de trabajo, es altamente recomendable seleccionar instalación personalizada. De no hacerlo así, piensa por un momento — ¿Qué lleva una instalación de servidor estándar? ¿Apache? ¿Samba? ¿Sendmail? ¿Bind? ¿dhcpd? ¿Postgres? ¿rntpd? ¿Squid? ¿POP3 e IMAP? ¿X? ... ¿Viene todo lo que necesitamos? ¿Necesitamos todo lo que viene? O tal vez más importante... Potencialmente, ¿puede hacernos daño algo que instalamos y no requerimos? Invariablemente: No, no, sí.

De hecho, si bien tomará mucho más tiempo, al instalar un servidor —una computadora que será visible a la red externa, que prestará servicios y probablemente sea blanco de ataques— te recomiendo ampliamente seleccionar las categorías de paquetes que requieras, y después de eso indicar al instalador que quieres seleccionar paquete por paquete qué instalar. Vuelvo al ejemplo anterior — Seleccioné instalar el servidor Apache, maravilloso... ¿Viene mod_perl? ¿PHP? ¿Apache_SSL? Los mismos inconvenientes del punto anterior, si bien mucho más locales. Al instalar un servidor es fundamental controlar la lista

exacta de paquetes a ser instalada. No te preocupes demasiado por las dependencias, ya que si te falta algo el mismo programa instalador te dirá qué es cuando le pidas continuar. Si no sabes qué hace un paquete, oprime F1 y el instalador te dará su descripción. Como regla general, si dudas acerca de si un paquete te puede ser útil, puedes omitirlo. Si lo requieres, luego lo instalarás.

Estos mismos comentarios, claro está, se aplican a las instalaciones de estaciones de trabajo. Tal vez muchas veces descuidamos la seguridad de nuestra estación de trabajo... Pero es ahí donde hacemos todo el desarrollo y donde tenemos todos nuestros datos almacenados. No darle su muy buena pensada a los paquetes que instalaremos en nuestra estación de trabajo puede hacer peligrar doblemente nuestra seguridad.

2.1.2 Debian

La instalación básica de Debian es muy diferente para el nuevo usuario, pero tras un par de puntos verás que es prácticamente lo mismo. Lo primero que llama la atención al instalar Debian es la interfaz del instalador... ¿Estamos aún en 1995? ¿Y por qué la interfaz es de texto? Bueno... Antes de aventar en un ataque de ira los CDs de Debian, ponte a pensar... ¿Qué ventaja te da una instalación gráfica? Ninguna... Muy al contrario. Debian puede aún instalarse desde floppies en computadoras que no cuenten con unidad de CD (o si tienes una buena conexión a red y no quieres bajar el ISO completo), cosa que en RedHat es bastante más compleja, y el proceso es casi igual. Y —aclaro, esto es meramente opinión personal— no existe una instalación más simple de efectuar que la de OpenBSD, cuya interfaz es un simple y llano script de shell. ¿para qué queremos pantallitas bonitas en un proceso que vamos a realizar una sóla vez? Mejor sencillez, simplicidad, funcionalidad.

Debian se instala en dos fases: Primero se instala el sistema base, fase durante la cual no hay mucho que acotar... Partición, módulos básicos, en fin, lo de siempre. Ya después de ello, con un sistema Unix funcional y tan completo como una instalación mínima lo permite, nos presenta los primeros detalles de configuración: Contraseña de root, cuenta de usuario, modo de manejo de las contraseñas. Entro más en detalle más adelante (2.3). Después de esto, al igual que en RedHat, nos da a escoger si queremos seleccionar los paquetes con la interfaz de experto (paquete por paquete) o simplificada (por categoría). Claro está, recomiendo fuertemente ir paquete por paquete. Recuerda que en este momento puedes elegir ya paquetes que tengas disponibles sobre la red... ¡Pero no bajes la guardia al instalarlos!

La interfaz que utilizas para seleccionar los paquetes es `dselect` — Aprende a quererlo, que será probablemente uno de tus mayores aliados. Claro, hay varios programas más amigables que reemplazan a éste, pero comparten los principios básicos.

Al seleccionar paquetes, para instalar uno márcalo con `+`. Para eliminar un paquete, `-`. Si quieres eliminar también los archivos de configuración, `_`. Si quieres que no sea modificado el paquete cuando haya alguna actualización, `=`. Dselect (que no es más que una interfaz para `apt-get`) se encargará del manejo

de dependencias.

2.2 Primer arranque

El sistema ya está instalado, y arrancas tu computadora con una gran sonrisa en la boca. ¿¡Pero qué haces con...!? ¡Desconecta eso! ¡Sí, el cable de red! ¿Que por qué? Porque la máquina está demasiado abierta todavía. Analicemos:

Acabas de instalar el sistema operativo. No has cerrado ningún servicio que viene activo por default (ver 2.2.1). No has parchado aún posibles errores del sisetma (ver 2.4). No hay aún herramientas de seguridad instaladas (3.1, 3.3). A más de una persona le han logrado comprometer su sistema en los primeros minutos de vida, cuando más confiaba en él, y pocos se dan cuenta hasta semanas o meses más tarde. Hasta que tengas la computadora en un estado presentable, no conectes ese bonito cablecito. Y como veremos en 4.2 y 4.3, las amenazas son mucho más fuertes de lo que imaginamos.

Ahora, claro, probablemente se te ocurrió ya una falla fundamental en mi argumento: Varias distribuciones te permiten la instalación por red. Si, como yo, tienes una muy buena conexión a Internet, puedes juzgar innecesario bajar y quemar imágenes ISO para hacer instalaciones básicas, ¿cierto? Bueno... Lo importante es aislar a tu computadora de la red hostil tanto como sea posible. ¿Cómo? Hay muchas maneras... Probablemente puedas poner a tu computadora en una dirección IP no homologada (enmascarada), o puedas restringir con tu firewall que no reciba conexiones entrantes. El grado de conexión/no-conexión depende exclusivamente de tus necesidades y posibilidades.

2.2.1 Paranoia inicial: A cerrar, a cerrar, que el mundo se va a acabar

Un concepto fundamental en seguridad es que *no* debes mantener abiertos servicios que no sean indispensables. Cualquier distribución viene, desafortunadamente, con varios puertos abiertos por default. Es muy raro que muchos de esos servicios sean utilizados hoy en día, y partiendo de que cualquier pedazo de código tiene altas probabilidades de tener un defecto interesante (y por tanto, explotable) escondido por ahí, entre más servicios estemos dando, más riesgo corremos de que nos peguen por aquella vulnerabilidad.

Hay varios protocolos que presentan un riesgo adicional: Por más seguro que sea un demonio, si requiere que la contraseña sea transmitida en texto claro (como lo hacen FTP, telnet, POP3, IMAP, etc.) abre nuestras máquinas a que un atacante olfatee la red esperando encontrar una contraseña, y la utilice para suplantar la identidad de su dueño. Estos servicios, claro está, deben ser también cerrados, o por lo menos, entubados en una conexión cifrada, como lo veremos en 3.2.2.

Una herramienta invaluable para asistirte al controlar acceso a los servicios que proporcionamos en nuestro sistema es un firewall, ya sea como un sistema dedicado o como reglas locales de filtrado, como lo veremos en 4.4.

inetd /xinetd Vayamos primero sobre del *superdemonio*, inetd. Este superdemonio está encargado de levantar una potencialmente gran cantidad de programas servidor para varios servicios, típicamente de baja complejidad pues funciona mejor con programas que no cueste mucho tiempo inicializar. Bajo el riesgo de equivocarme en alguno, en inetd vienen abiertos varios servicios, como chargen, echo, telnet, ftp, rsh, rexec, rwho, talk, finger, ident.

Además de ser innecesarios hoy en día, pueden ser muy peligrosos. ¿Ejemplos? Con identd, finger, rwho y chargen es posible averiguar información acerca de quién está usando y qué está corriendo una comptuadora —información, claro, muy valiosa para un atacante. Varios de estos demoniostambién han presentado casos de buffer overflows.

Muy bien, ya eché mi rollote... Ahora bien, ¿Cómo puedo cerrar servicios innecesarios? Fácil: El archivo de configuración de inetd es el `/etc/inetd.conf`, y su sintaxis es muy sencilla. A mí me toca todavía más fácil: No tengo que enseñarte cómo agregar servicios a inetd (que para eso está la página del manual, claro ;-)), sino que simplemente cómo cerrarlo.

Este archivo se puede analizar línea por línea. Primero que nada, ¿qué es un comentario? Cualquier cosa después de un signo `#`. Si el signo `#` está al principio de la línea, toda la línea es considerada un comentario y su significado es nulo para inetd.

Todas las líneas que no son comentarios y no están en blanco inician con ya sea una palabra o un número. Éste es el que define qué servicio ejecutará —por ejemplo, la línea:

```
telnet stream tcp nowait telnetd.telnetd /usr/sbin/tcpd /usr/sbin/in.telnetd
```

indica que estamos hablando de telnet. Claro, la computadora tendrá que traducir esto a un número de puerto; para esto extá el archivo `/etc/services`, donde nos indica que telnet corresponde al puerto 23 de TCP. Si queremos deshabilitar ese servicio, únicamente tenemos que comentar la línea en cuestión, de esta manera:

```
#telnet stream tcp nowait telnetd.telnetd /usr/sbin/tcpd /usr/sbin/in.telnetd
```

Después de hacer los cambios pertinentes, reiniciamos inetd. Esto puede ser –dependiendo del tipo de Linux que tengamos– de una de las siguientes maneras:

`/etc/init.d/inetd stop; /etc/init.d/inetd start` para Debian

`/etc/rc.d/init.d/inetd stop;/etc/rc.d/init.d/inetd start` para RedHat y derivados anteriores al 7.0

Cabe mencionar que la versión 7.0 y superiores de RedHat (y es de esperarse que otras varias distribuciones sigan este ejemplo) incluye a un reemplazo para inetd llamado xinetd. La configuración general, localizada en el archivo `/etc/xinetd.conf` es:

```
defaults
{
instances               = 60
        log_type                = SYSLOG authpriv
```

```
        log_on_success = HOST PID
        log_on_failure = HOST
cps = 25 30
}

includedir /etc/xinetd.d
```

Con xinetd tenemos un control más granular acerca del comportamiento de cada uno de los servicios. Por ejemplo, en la configuración anterior vemos que xinetd limitará a 60 instancias (conexiones simultáneas), lo cual puede reducir dramáticamente los ataques de negación de servicio. En este ejemplo vemos también cómo las conexiones serán registradas en las bitácoras del sistema. Y en vez de definir cada servicio en una línea, en el directorio `/etc/xinetd.d` tenemos un archivo por servicio (toman el nombre de las líneas definidas en `/etc/services`) como el siguiente, `/etc/xinetd.d/telnet`:

```
service telnet
{
flags = REUSE
socket_type = stream
wait = no
user = root
server = /usr/sbin/in.telnetd
log_on_failure += USERID
disable = yes
}
```

La mayoría de los campos son análogos a los que vemos en inetd.conf — La única diferencia clara es el campo `enable`. En vez de comentar la línea, basta con marcar `disable=yes` para que el servicio no sea iniciado.

Cuál de los dos usar? Es completamente cuestión de preferencia personal. xinetd te da la ventaja de mayor granularidad, inetd presenta un panorama más claro de qué está activo y qué no. Mi recomendación, personal y subjetiva, va por inetd, ya que siendo hoy cada vez menos (y menos importantes) los servicios proporcionados a traves de este mecanismo, se me hace más importante tenerlos centralizados en un solo archivo que el poner controles específicos a cada uno de ellos — Labor muy fácil de hacer, de todos modos, ya sea a través de TCPWrappers o de reglas de filtrado de paquetes.

Otros demonios independientes Por otro lado, tenemos los demonios *per sé*, los programas que están contínuamente corriendo en nuestro sistema. En un sistema Linux típico basado en SysVR4 (no Slackware), encontraremos qué programas demonio tenemos corriendo examinando los directorios de inicio. Para esto, consultamos en qué *runlevel* estamos ejecutando (busca la línea `id:`x`:initdefault:` en tu archivo `/etc/inittab`; el número que esté en la x será el runlevel en el que entra tu sistema por default. Este será típicamente 2 para Debian, y 3 o 5 para RedHat.

Con esto en mente, hay dos sopas: Hacerlo a mano o a través de un programita que nos ayude. Si tienes RedHat y quieres irte por el camino fácil, ejecuta `/usr/sbin/ntsysv`.

Si sigues leyendo esto, eres de los míos: Quieres saber cómo funcionan las cosas. Te felicito de antemano. Vamos a entrar en materia. A propósito, si utilizas Debian y quieres hacer esto a su más puro estilo, tienes a `update-rc.d`, que de todos modos asume que sabes manejar tu sistema.

Dentro de `/etc` (o de `/etc/rc.d`, si estás usando un sistema RedHat 6.2 o anterior) encontrarás los directorios `init.d`, `rc0.d`, `rc1.d`, `rc2.d`, `rc3.d`, `rc4.d`, `rc5.d` y `rc6.d`. Algunos sistemas tendrán también a `rcS.d`. En `init.d` están los scripts de arranque/finalización de todos tus demonios, y en los demás directorios (uno por cada runlevel) hay ligas simbólicas hacia dichos scripts, con una convención especial en el nombre: `{K|S}##xxxx`, donde:

K Terminar el demonio al entrar al runlevel indicado

S Iniciar el demonio al entrar al runlevel indicado

El órden en el que será iniciado/terminado (ascendente)

xxxx Nombre del demonio

Por ejemplo, si entramos en runlevel 2 por default y tenemos los siguientes archivos en /etc/rc2.d:

```
gwolf@mipc:/etc$ ls rc2.d/
S10sysklogd  S20dhcp  S20inetd    S20lpd         S20snort  S89cron       S99wdm
S12kerneld   S20exim  S20ipac     S20makedev     S20ssh    S91apache
S14ppp       S20gpm   S20logoutd  S20postgresql  S89atd    S99rmnologin
```

Esto significa que al encender el sistema o entrar a runlevel 2, éste inicia (en órden) la ejecución de `sysklogd`, `kerneld`, `ppp`, `dhcp`, `exim`, `gpm`, `inetd`, `ipac`, `logoutd`, `lpd`, `makedev`, `postgresql`, `snort`, `ssh`, `atd`, `cron`, `apache` y `rmnologin`.

Para evitar que un demonio inicie en el runlevel default basta con quitar la liga hacia él; por ejemplo, si decides que ya no requieres dar servicios de Apache, basta con la siguiente línea:

```
rm -f /etc/rc2.d/S91apache
```

O bien, si usas Debian, puedes hacer:

```
update-rc.d -f apache remove
```

El crear la liga con la K nos sirve para indicar que al entrar a este runlevel, en caso de que cierto demonio esté corriendo, lo mate. Recuerda que podemos cambiar de runlevels sin reiniciar el sistema, con el comando

`init` ***x***

Siendo ***x*** el runlevel al que queremos entrar.

Portmap Hay algunos servicios que funcionan sobre un mecanismo llamado *Remote Procedure Call* (RPC). Algunos de estos son NFS, NIS/YP, y otros. La mayor parte de los usuarios hoy en día no requieren a ninguno de ellos, y creen que basta con detener al demonio indicado. Portmap es, sin embargo, un demonio que guarda información acerca de todos estos demonios; mucha gente ignora la existencia de Portmap, sin embargo, y lo deja corriendo.

Una gran cantidad de exploits han aparecido aprovechando debilidades en el diseño de Portmap. Te sugiero fuertemente que lo desactives y elimines del sistema a menos que realmente lo requieras. Muchos sistemas lo levantan desde rcS.d, otros varios desde cada runlevel en específico.

Portmap es iniciado como demonio independiente, y escucha por el puerto 111.

Los infames r-commands Mención especial entre los servicios nefastos merecen los llamados *r-commands* - Comandos remotos. Estos servicios aparecieron cuando Internet era todavía una red académica y confiable; están hechos para permitir hacer ciertas operaciones fácilmente entre computadoras. Los principales r-commands son:

rsh, en una máquina remota, o iniciar una sesión en ella

rcp Copia archivos de una computadora a otra

rwho Revisa qué usuarios están conectados a un servidor remoto

Estos comandos son triplemente peligrosos:

- Toda la interacción es transmitida en claro y puede ser facilmente detectada por un sniffer, y modificada con herramientas que cualquier cracker interesado puede fácilmente conseguir.
- Para la autentificación, la contraseña es transmitida en claro, como en el caso de Telnet.
- Tienen un mecanismo de confianza, manejado a través del archivo `/etc/hosts.equiv` y los archivos `.rhosts` en el home de cada usuario, en el que especifican en qué servidores confían. Esta confianza se basa únicamente en su dirección IP, por lo que si alguien envía paquetes que parezcan venir de dichas máquinas, éste pasará sin requerir autentificación.

Los r-commands son típicamente activados desde inetd, por lo que basta comentar las líneas que los invocan (y claro, reiniciar inetd) para cerrarlos. Típicamente vienen declarados en inetd como shell (rsh), login (rlogin), exec (rexec) y rwho (rwho).

Todos estos comandos pueden, además, ser reemplazados con sus equivalentes seguros con Secure Shell, con la misma sintaxis y prácticamente el mismo nombre (ver 3.2.1

Revisando todo: netstat Nuestra principal preocupación en este momento son típicamente los ataques provenientes de la red[1]. netstat es un comando que podemos encontrar en todo Unix, y que nos permitirá cerciorarnos de que no

[1] Recuerda que esto no siempre es cierto: La mayor parte de los intentos de ataque son internos, de gente que sí tiene acceso legítimo a nuestros sistemas.

estamos dando más servicios de los estrictamente requeridos. Esta, claro, es sólo una de las funciones que tiene netstat - siendo una herramienta tan útil, te sugiero leer su página de manual si no estás familiarizado con él.

En este caso, voy directo sobre una de las formas de invocar a netstat. Nos interesa saber qué puertos tiene abiertos nuestro sistema. Vamos primero sobre los puertos TCP:

```
mipc:~# netstat -nap |grep -w 'LISTEN\|udp'
tcp        0      0 0.0.0.0:25              0.0.0.0:*               LISTEN      828/inetd
tcp        0      0 0.0.0.0:6000            0.0.0.0:*               LISTEN      227/X
tcp        0      0 0.0.0.0:1024            0.0.0.0:*               LISTEN      219/wdm
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      209/apache
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      194/sshd
tcp        0      0 0.0.0.0:515             0.0.0.0:*               LISTEN      169/lpd
udp        0      0 0.0.0.0:177             0.0.0.0:*                           219/wdm
udp        0      0 0.0.0.0:67              0.0.0.0:*                           150/dhcpd-2.2.x
```

Los modificadores `nap` (`p` está disponible únicamente en Linux, `n` y `a` son estándares en otros Unixes) y los filtros aplicados nos indican:

n No resuelve las direcciones a sus nombres DNS. Esto hace mucho más rápida la ejecución. Recuerda que la computadora sigue desconectada de la red.

a Muestra todos los sockets, no sólo los que estén conectados. Esto es muy importante, porque estamos —justamente— buscando los sockets que están escuchando.

p Muestra el número y nombre de proceso dueño de dicho socket. Gracias a esta opción, única a Linux, podrás darte cuenta rápidamente qué proceso está abriendo dicho socket - Por ejemplo, si no sé de qué se trata el socket 5432, aquí me doy cuenta que es propiedad de postmaster, el demonio de PostgreSQL.

|\|udp' netstat nos da la información de todos los sockets, conetcados o en espera, de dominio local Unix o Internet. Nos interesa en este momento únicamente ver a aquellos que estén escuchando (y por eso el LISTEN). Sin embargo, UDP es un protocolo que no maneja estado, por lo tanto listamos todos los puertos que estén abiertos para UDP... Y de todos modos, recuerda que aún no conectas tu cable de red ;-)

2.2.2 Los grandes y eternos errores de RedHat

Históricamente, RedHat se ha empecinado en incluir paquetes que han demostrado ser altamente vulnerables a ataques, probablemente por no haber tomado en cuenta la seguridad en su etapa de diseño... Y los siguen incluyendo hoy en día.

Coincidentemente (o tal vez no), estos paquetes provienen en su mayoría de la Universidad de Washington. Para todos ellos, te sugiero desactivarlos y desinstalarlos. Si requieres la funcionalidad que ofrecen, inetnta instalar un programa substituto, casi todos tendrán uno disponible.

wu-ftpd Demonio de FTP. Si requirieras dar servicio de FTP (que puede muy bien ser suplido por http para transferencias anónimas, y no debe ser utilizado para transferencias que requieran autentificación), te sugiero instalar ProFTPD[25]. De todos modos, siempre que puedas reemplázalo por sus equivalentes seguros (SSH 3.2.1).

wu-imapd Demonio que provee IMAP, POP2 y POP3, protocolos populares para que usuarios remotos puedan consultar su correo sin entrar al servidor. Puedes instalar Cyrus-IMAP o QPopper en vez de este paquete. Te sugiero fuertemente envolver a toda conexión para estos protocolos en un túnel cifrado con stunnel 3.2.2.

wu-pine Cliente de correo en modo texto muy popular. No hay un reemplazo exacto a pine, pero hay una gran cantidad de clientes de correo adicionales — Uno de los más populares es mutt [26]. Muchos de tus usuarios pueden solicitarte que mantengas pine en el sistema, lo que deberás cuidar celosamente es estar siempre corriendo la última versión.

bind Este demonio no es muy prescindible – bind es por mucho el servidor de DNS más utilizado en Internet. Sin embargo, el historial de seguridad de bind es tan malo como el peor. Hoy en día no hay verdaderas opciones libres para reemplazarlo. Está DJBDNS [27], escrito por D. J. Bernstein, que parece ser una muy buena y muy viable alternativa a bind, pero desafortunadamente no tiene una licencia libre — No permite redistribución de binarios modificados. Hay algunos otros proyectos, como bind 9 y dents, pero no están en un estado listo para producción.

2.3 Manejo de los archivos de contraseñas

Hoy en día, casi todas las distribuciones presentan la posibilidad de utilizar *shadow passwords* y *MD5*. ¿Qué es esto?

Tradicionalmente, en los sistemas Unix manejamos todas las cuentas del sistema en el archivo `/etc/passwd`, donde tenemos los datos generales del usuario así como su contraseña cifrada con el algoritmo DES del NIST. Este es un algoritmo de cifrado de 64 bits, excelente en su momento, pero casi trivial de tronar con el poder de cómputo actual — ¡Y el archivo de contraseñas requiere ser legible para todos los usuarios del sistema! Esto se solventa con dos mecanismos.

2.3.1 Shadow passwords

2.3.2 MD5

2.4 Parches, parches, parches

Conforme pasa el tiempo, nuevos errores de programación se van encontrando en prácticamente todos los programas. Muchas veces, estos errores se ven reflejados en vulnerabilidades que pueden comprometer nuestro sistema completo. Es por tanto indispensable que nos mantengamos al día con los parches.

2.4.1 RedHat y derivados - RPMs

Actualizar un sistema RedHat es muy sencillo. Claro, el primer paso para actualizar es estar informado. La mejor fuente para hacerlo será una lista de correo específica de tu distribución; de todos modos; de todos modos, la lista de anuncios de RedHat [28] es siempre un buen punto de referencia.

Claro está, durante la instalación y al dar los primeros pasos –insisto– la computadora no está conectada. Te sugiero hacer un CD con las actualizaciones que hayan salido hasta el momento –entre 50 y 200MB normalmente– e instalar todo lo que requieras con el comando

```
rpm -Fvh /mnt/cdrom/*
```

Claro, con el directorio en donde tengas los RPMs en vez de `/mnt/cdrom` si es el caso. La opción F de RPM es para que actualice únicamente los paquetes que requiere actualizar. Por ejemplo, si en mi computadora no instalé bind, instalar la actualización de bind sería inútil o hasta contraproducente (pues puede introducir vulnerabilidades que no existían previamente en el sistema); con F no se instalaría, aunque sí se instalaría cualquier otro paquete requerido.

2.4.2 Debian y derivados – apt-get

Debian nos presenta una gran ventaja respecto a cualquier otra distribución: El sistema apt-get. Conviene tener la siguiente línea en el archivo `/etc/apt/sources.list`, que indica a apt-get dónde buscar los programas a instalar:

```
deb http://security.debian.org stable/updates main contrib non-free
```

Actualizar el sistema completo puede ser hecho simplemente con los comandos:

```
apt-get update; apt-get dist-upgrade
```

apt-get tiene una gran cantidad de opciones; si bien correr esto a diario te ayudará a mantener el sisetma al día, hay varias salvedades para las cuales es imprescindible leer la documentación.

Como nota al pie, apt-get es un sistema tan poderoso que nos permite actualizar el sistema completo a una versión más nueva de la distribución con tan sólo indicarlo, sin romper dependencias ni crear conflictos. apt-get es gran parte de la razón por la que los miles de usuarios de esta distribución la aprecian tanto por sobre de RedHat y otras distribuciones.

3 Apretando los tornillos a nuestro sistema: Paquetes adicionales y proyectos interesantes

Prácticamente todas las distribuciones incluyen ya algunas de las herramientas de seguridad que en su momento parecieron tremendamente innovadoras y hoy en día son ya dadas por hecho, como es el caso de los maravillosos TCP-Wrappers de Wietse Venema. sin embargo, hay varias herramientas que pueden serte muy útiles. Instalarlas es lo de menos... Y de ahí en más, verlas correr, verlas analizar tu sistema, verlas velar por tí es sencillamente un deleite. Algunas de ellas son:

3.1 Psionic: Portsentry y Logcheck

No sé cómo estas dos herramientas no vienen instaladas por default en todo sistema Unix. Ambas son parte del proyecto Abacus de Psionic [29].

Portsentry es un detector de barridos de puertos e intentos de conexión a puertos cerrados, que no sólo previene sino que toma acción correctiva bloqueando toda comunicación entre el posible atacante y nuestro sistema. Prácticamente todos los ataques comienzan con una fase de recopilación de información, en que el atacante intenta encontrar todos los datos posibles acerca de nuestro sistema... Y Portsentry lo manda a volar sin darle oportunidad de aprender prácticamente nada respecto a nosotros. Para un tutorial más a fondo de Portsentry, consulta [**?**].

Revisar las bitácoras es una de las obligaciones más importantes, pero más tediosas y menos queridas, de un administrador de sistemas. Esto es en buena parte porque tenemos que acordarnos de hacerlo, tenemos que recordar en qué línea nos quedamos, no hay manera automática de prioritizar los mensajes, y nos toca analizar bloques bastante grandes (te gustan unas 600 líneas diarias en un sistema con actividad moderada?). Logcheck te simplifica esta tarea, enviándote a tu buzón con el intervalo que le especifiques (te sugiero una hora), acomodado por prioridades, lo que llegue a tu bitácora. Si quieres más al respecto, asómate a [**?**]

3.2 Cifrado a todo nivel

Las redes hoy en día son —y afortunadamente lo sabemos— inseguras, y nos imposibilitan confiar en lo que transportan. El cifrado a todos niveles nos permite asegurar por lo menos los siguientes puntos, fundamentales para virtualmente toda operación:

Privacidad Nadie que no sea el destinatario debe poder ver la información que enviamos.

Integridad Debemos poder tener la certeza de que la información que recibimos es exactamente la información que nos fue enviada.

Irrefutabilidad Debe haber mecanismos claros para determinar que la comunicación que recibimos viene realmente de quien dice provenir.

Tenemos las siguientes herramientas principales para cifrar nuestra comunicación usando sistemas Linux:

3.2.1 Secure Shell

Una gran ventaja de los sistemas Unix es su capacidad de ser administrados remotamente. Por muchos años, telnet fue una maravillosa herramienta para hacer esto. Adicionalmente, FTP nos permitía la transferencia de archivos entre sistemas, y el sistema de ventanas X desde un principio permitió el manejo transparante de conexiones remotas. Sin embargo, como ya saben, eso ya no es suficiente gracias a la facilidad y frecuencia de los olfateos en una red.

Secure Shell nos permite hacer todo esto, de una manera segura y cifrada. Pensado originalmente como reemplazo de telnet/rlogin/rsh/rexec (para manejar sesiones remotas), hoy incluye a scp y sftp, para la copia de archivos remotos, y dentro del mismo ssh, el mecanismo para crear túneles de puertos sobre canales cifrados, lo que nos permite, entre otras muchas cosas, utilizar sesiones X remotamente sin preocuparnos por intrusos espiando nuestra conexión — y sabiéndolo utilizar, ssh nos puede dar toda la infraestructura necesaria para una VPN completa.

No hay excusa hoy en día para no tener Secure Shell instalado en nuestros sistemas, al igual que no hay excusa para utilizar conexiones no cifradas.

3.2.2 Para los protocolos inseguros, stunnel

Algunas aplicaciones hoy en dia, pese a lo que recién comenté, no pueden ser manejadas por Secure Shell. Un ejemplo muy clásico es el de las máquinas Windows cliente que consultan su correo en nuestro servidor — y hay muchos más. Si bien no podemos exigir a nuestros usuarios que utilicen Secure Shell para crear canales cifrados antes de abrir una sesión de correo (imagínalo simplemente...), sí podemos poner a su disposición versiones cifradas con el estándar SSL de los protocolos inseguros que manejan. La mayor parte de los clientes de correo reconocen los puertos 993 y 995 para IMAP y POP3 (respectivamente) sobre SSL. Stunnel nos permite cifrar cualquier protocolo, siempre que el programa cliente comprenda también SSL.

3.2.3 Y para el correo, PGP/GnuPG

Probablemente el programa de cifrado más conocido en el mundo sea PGP. Desde su polémica aparición hace 10 años ha permitido a todo mundo tener acceso a criptografía fuerte junto con un robusto esquema de redes de confianza para llaves públicas, lo que en español significa que nos permite mantener nuestras comunicaciones cifradas y seguras.

GnuPG, la versión GNU de PGP, esta ya perfectamente al nivel de su predecesor, garantizándonos todas las libertades que nos da trabajar con software GNU, siendo compatible con el PGP original.

3.3 Herramientas clásicas de seguridad

Dirán lo que quieran, pero las herramientas clásicas de seguridad siguen siendo una importantísima herramienta para asegurar y verificar nuestro sistema. Puedes encontrarlas en el FTP del Departamento de Seguridad en Cómputo – ftp://ftp.seguridad.unam.mx, y muchas veces dentro de tu misma distribución de Linux.

Una instalación default de Linux incluye ya a algunas de estas herramientas completamente integradas al sistema, notablemente a passwd+, shadow y TCP Wrappers.

Estudiar las principales herramientas de seguridad está más allá del propósito de este texto. Asómate a los tutoriales del Departamento de Seguridad en Cómputo [32]. En general, las herramientas de seguridad pueden clasificarse en:

Sistmas(IDS) Monitorean contínuamente a un sistema o una red buscando comportamientos anómalos y toman acción inmediata al respecto. ¿Ejemplos? snort, portsentry, swatch.

Analizadoreslidades (o, simplemente, de puertos abiertos). Pueden servir tanto a un administrador como a un atacante. ¿Ejemplos? nmap, nessus, satan, saint, tara, etc.

Analizadores posibles vulnerabilidades o errores de configuración, o revisan la integridad de sus archivos. ¿Ejemplos? Cops, tiger, tripwire, etc.

Herramientas un equipo en diferentes rubros. ¿Ejemplos? logcheck, npasswd, passwd+, crack, etc.

3.4 Bastille Linux

Bastille Linux [33] es un conjunto de scripts diseñados para ayudar a cerrar los posibles problemas en las distribuciones derivadas de RedHat, y recientemente han aparecido paquetes de Bastille en la rama de desarrollo de Debian. Un punto muy positivo de Bastille es que cada paso que da lo explica al operador, evitando crear agujeros en su conocimiento – tanto o más importantes que los del sisetma.

Bastille Linux es un proyecto muy ambicioso y vital; casi a diario aparecen noticias en su página, y es de esperarse que en un futuro cercano o medio sea integrado a más de una distribución de Linux.

3.5 OpenWall

El proyecto OpenWall [34] creó un grupo de parches disponibles para el kernel del Linux, y una muy buena forma de prevenir ataques como Buffer Overflows y similares. Son una coleccion de mejoras de seguridad para integrar en bloque en el kernel, configurables todas ellas desde una nueva sección de Seguridad que es añadida al menu de configuración del kernel, y que verás a la hora de reconfigurarlo. Estos parches estan disponibles para diferentes versiones del

kernel, pero hay que recalcar que *no* son parte de la distribución central. algunos de sus puntos son:

- Stack no ejecutable: La mayor parte de los ataques por buffer overflow buscan sobrescribir la direccion de retorno en la pila, apuntando a codigo arbitrario introducido por el atacante, que tambien es puesto en la pila. Si el area de la pila no es ejecutable, estos buffer overflows se vuelven mas complicados de explotar.

- Ligas y FIFOs en /tmp restringidas: Ciertos programas son vulnerables a que el intruso cree una liga simbólica en /tmp, la cual el programa no comprueba, y el atacante usa para sobreescribir u obetner datos de otra región del sistema. Activando esta opcion se reduce el impacto de este tipo de ataques, no permitiendo a un proceso seguir un archivo que es un enlace en un directorio temporal

- /proc restringido: restringe el acceso a los directorios en /proc, de tal forma que los usuarios solo pueden ver sus procesos en el sistema y ningun tipo de actividad de conexiones de la red, salvo que se encuentren en un grupo especifico. Tambien impide el uso del comando 'dmesg' a los usuarios.

- Destruir segmentos de memoria no utilizados: Unix te permite especificar cuanta memoria puede consumir un proceso. Desafortunadamente los segmentos de memoria compartidos pueden existir sin estar asociados a ningun tipo de proceso. Esto destruira segmentos de memoria que ya no se encuentran en uso, que no han sido vinculados a ningun proceso.

4 Retos diarios

Bueno... Tu sistema está ya cuidadosamente instalado, parchado, asegurado y listo para el mundo. Hora de dejar que tu esfuerzo rinda frutos — Estás ya listo para conectarlo a Internet.

Claro está, el esfuerzo no termina aquí. Día a día aparecen nuevas amenazas, día a día tenemos que protegernos y estar mejor preparados que nuestro contrincante. Es imposible resumir todo lo que esto implica, pero menciono algunos puntos importantes.

4.1 Nuevos agujeros, siguiendoles la pista

Hay varios lugares con información acerca de nuevos problemas de seguridad, incluyendo vulnerabilidades. Lo primero que debes hacer, como lo mencioné en 2.4.1, es subscribirte a la lista de anuncios de tu distribución; ahí siempre tendrás la última información tan pronto sea liberada. Otra lista muy importante es Bugtraq [35], de SecurityFocus, donde se discuten posibles vulnerabilidades, y sirve muchas veces como base para la reacción de las diferentes distribuciones.

Hay varios sitios importantes que merecen visitas periódicas. ¿Cada cuánto? Entre más seguido, mucho mejor. De nuevo, son demasiados. Los primeros que recuerdo son:

- Departamento de Seguridad en Cómputo, http://www.seguridad.unam.mx y http://www.unam-cert.unam.mx/ – Información general de seguridad en español; equipo de atención a emergencias de cómputo para el dominio .unam.mx
- CERT-CC, http://www.cert.org – Centro de Coordinación de los grupos CERT internacionales. Aquí tienes los últimos boletines de seguridad, así como prácticamente toda la información relativa a qué hacer para evitar y cómo reaccionar ante intrusiones.
- Security Focus, http;//www.securityfocus.org – Portal general de seguridad en todos los sistemas operativos
- Linux Security, http://www.linuxsecurity.com – Dedicado especialmente a Linux

Y, claro está, muchos más.

4.2 La nueva moda – Gusanos

Los gusanos no son un nuevo concepto; el primer incidente documentado sobre Internet fue, justamente, el famoso gusano de Morris, en 1989. Sin embargo, la moda actual de los gusanos data de hace relativamente poco tiempo – En enero

del 2001 apareció el gusano Ramen y poco después tuvimos a Lion. Ambos explotaban fallos de seguridad en las distribuciones de RedHat 6.2 y 7.0. En julio, agosto y septiembre del 2001 tuvimos la proliferación de gusanos atacando a las diferentes versiones de Windows NT: Code Red, Code Blue, Code Green, Code Rainbow, Nimda... Siguiendo, todos ellos, el mismo patrón. Lo que es peor, explotando las mismas vulnerabilidades de sistemas mal configurados.

Estos gusanos atacan indiscriminadamente, buscando de manera automática un host al que atacar buscando dirección por dirección en enormes segmentos de red, y conectándose únicamente al puerto al que saben explotar. Hay a la fecha gusanos que explotan FTP (21/tcp), DNS (53/udp), Portmap (111/tcp), LPR (515/tcp) y HTTP (80/tcp). La cura? Simple: Tener el sistema al día. Los gusanos no son inteligentes, saben atacar únicamente a una versión de un demonio. Si actualizas la versión vulnerable por una más nueva, quedarás a salvo de sus ataques.

Un punto importante: Ten mucho cuidado de parchar todas las máquinas de tu segmento, no sólo los servidores principales. Aún si no te duele que una máquina sea atacada pues no hace nada importante, ésta comenzará a atacar a todas las computadoras que pueda, chupando todo el ancho de banda del que dispongas y posiblemente metiéndote en problemas de responsabilidad legal, pues el comportamiento de tus computadoras es responsabilidad tuyo.

4.3 Barridos, y cómo responder a ellos

Es muy común encontrar que nuestro sistema está siendo barrido (¿Instalaste portsentry, verdad?). Las primeras veces siempre te alarman, hasta que te des cuenta que es algo normal. Sin embargo, cada que recibas un barrido... Revisa una vez más que no haya nada raro en tu sistema, y toma nota de la dirección orígen. Si recibes barridos con frecuencia de la misma dirección, revisa quién es el responsable de dicha red, y repórtale el incidente, de la manera más detallada posible, incluyendo horas de inicio y término de los barridos registrados.

Para averiguar los datos de contacto del responsable de una red, utiliza el servicio whois de ARIN (http://www.arin.net/whois/index.html, dominios del continente americano), InterNIC (http://www.internic.net/whois.html, dominios genéricos), RIPE (http://www.apnic.net/, dominios europeos) o APNIC (http://www.apnic.net/, dominios de Asia y el Pacífico), estos engloban a la gran mayoría de las direcciones asignadas.

4.4 Reglas de filtrado de paquetes (firewall)

Prácticamente toda computadora Unix es capaz de manejar reglas de firewall. Esto es muy útil aún cuando esta no sea el firewall central de la organización — El tener reglas de filtrado de paquetes para determinar a quién se le proporciona cada uno de los servicios añade un muy gran nivel de control a nuestro equipo.

Linux maneja para el filtrado de paquetes diferentes sistemas, dependiendo de la versión de kernel. Con la versión 2.0.x o anteriores, el sistema manejado era ipfwadm; con la versión 2.2.x se cambió a ipchains, y en la versión 2.4.x

se maneja iptables. Toda nueva versión (hasta ahora) soporta la sintaxis de la versión anterior, si bien agrega capacidades suficientemente interesantes para hacer que valga la pena aprender a manejar la propia.

La sintaxis de las reglas de firewall, desafortunadamente, no es tan simple como en la manejada por IPF, disponible para los demás Unixes, o PF, disponible en OpenBSD. Sin embargo, la sintaxis básica es muy simple.

Trabajaremos aquí únicamente con ipchains, si bien la sintaxis de iptables es bastante similar, y sólo con un nivel que nos permitirá crear las reglas más básicas que requiera nuestra computadora. Para un firewall en forma, te remito al tutorial que preparó uno de sus principales autores, Rusty Russell [36]. La información que yo presento aquí viene exclusivamente de la página de manual de ipchains [37].

Cuando el código de firewall del kernel está activo, hay tres *cadenas* principales por las que pasa un paquete: INPUT, OUTPUT y FORWARD. Además de esto, podemos crear cadenas definidas por el usuario.

Para definir a política que seguiremos con los paquetes que nos interesen, definimos *reglas* sobre cada una de las cadenas. Cada una de estas reglas especifica el criterio para que un paquete sea tratado de diferente manera, e indican cómo será tratado el paquete. Estas reglas pueden tener uno de los siguientes objetivos:

ACCEPT Permite que el paquete pase

DENY Evita que pase el paquete, soltándolo "en el piso" — No avisa al emisor que el paquete fue rechazado.

REJECT Evita que pase el paquete, pero responde con un mensaje ICMP al emisor indicando que el paquete fue rechazado

MASQ Sólo es válido para FORWARD y cadenas definidas por el usuario, y cuando el kernel fue compilado con la bandera CONFIG_IP_MASQUERADE. Los paquetes pueden pasar, pero aparentan haberse originado en este servidor y no en quien realmente los envió. Al recibir respuesta, pasa automáticamente por el desenmascaramiento.

REDIRECT Sólo es válido para INPUT y cadenas definidas por el usuario, y cuando el kernel fue compilado con la bandera CONFIG_IP_TRANSPARENT_PROXY. Causa que un paquete sea "secuestrado" y redirigido al puerto y host especificado.

RETURN Equivale a cuando un paquete llega al final de las reglas y no fue tratado explícitamente — Será tratado con el objetivo default para esa cadena.

Al definir nuestras reglas podemos agregarlas (-A), eliminarlas (-D), reemplazarlas (-R), y otros varios métodos para los cuales los remito a la página del manual.

Y por último, los parámetros: Definimos cuándo se aplica cada una de las reglas cuando se cumple. Podemos negar uno de estos criterios con !:

- -p (–protocol) *[!] protocolo*
- -s (–source) *[!] dirección[/máscara] [!] [puerto[:puerto]]*
- –source-port *[!] [puerto[:puerto]]*
- -d (–destination) *[!] dirección[/máscara] [!] [puerto[:puerto]]*
- –destination-port *[!] [puerto[:puerto]]*
- –icmp-type *[!] tipo*
- -i (–interface) *[!] nombre*

Y, de nuevo, otras varias opciones que no trato aquí.

Por ejemplo, el siguiente comando agrega una regla que bloquea todos los paquetes provenientes de la red 192.168.0.0/24 al puerto de Web (80):

```
ipchains -A INPUT -p tcp -s 192.168.0.0/24 --destination-port
80 REJECT
```

Y el siguiente permite que llegue a este puerto únicamente la computadora 192.168.0.25:

```
ipchains -A input -p tcp -s 192.168.0.25 --destination-port
80 ACCEPT
```

Crear reglas es, por qué no aceptarlo, bonito y divertido... Pero llega mucho más allá de lo que persigo con este tutorial.

4.5 Sistema de detección de intrusos

Un sistema de detección de intrusos (IDS) es muchas veces visto como parte de un firewall. Hay quien cree que por estar enviando a la bitácora notificación de los paquetes rebotados ya tiene un sistema de IDS. Gran mentira. Un IDS es justamente la herramienta que detectará comportamiento hostil dentro de una conexión legítima - Por ejemplo, mi firewall va a permitir que cualquiera hable con el puerto 80 (http) de mi servidor de Web. Sin embargo, ¿qué pasa si recibo la siguiente solicitud?

`GET ../../../../etc/passwd`

¡Esto enviaría mi archivo de contraseñas a quien lo solicitara! (claro está, ningún servidor de web hoy en día accedería a un ataque tan primitivo) Y este ataque, sea o no exitoso, debe ser detectado a tiempo y, de ser posible, detenido.

Hay dos tipos de IDSs: Los que están dedicados a monitorear actividad sospechosa en el host mismo (IDSs orientados a host) y los que monitorean la red buscando huellas de intentos de ataque, no necesariamente hacia la computadora en que están alojados (IDSs orientados a red).

4.5.1 IDSs orientados a host

Hay una gran cantidad de programas que cumplen estas funciones: TripWire [38] detectará modificaciones a los archivos que le especifiquemos, HostSentry (parte del proyecto Abacus, [29]) nos reportará anomalías relativas al uso de cuentas de usuario en nuestros sistemas.

Hay muchos otros IDSs orientados a host, y de hecho, muchos programas que en realidad no fueron explícitamente diseñados como uno pueden ser clasificados como tales.

La importancia de los IDSs orientados a host es mucho mayor de lo que mucha gente piensa: Hay una infinidad de maneras de conseguir acceso elevado en casi cualquier sistema, y una vez que el atacante lo tiene, cualquier cosa que haga parecerá —ante un IDS orientado a red— actividad normal y legítima.

4.5.2 IDSs orientados a red

Hay un claro primer lugar en IDSs orientados a red libres: Snort [39]. Este detector de intrusos cuenta con una gran cantidad de firmas para identificar los diferentes ataques, es relativamente sencillo de mantener, y se mantiene bastante al día. Ahora, Snort por sí solo lo único que hará es registrar los intentos de ataque. Para que este IDS realmente nos sea útil, requerimos herramientas que procesen de alguna manera útil los datos que reciben de Snort. Algunas de ellas son:

- Snorticus [40] es una colección de scripts en shell diseñados para simplificar el manejo y la administración de Snort, analizar sus resultados y mantener las definiciones de reglas.
- ACID (Analysis Console for Intrusion Databases) [41] toma los datos que fueron almacenados por Snort en una base de datos y los presenta con una interfaz Web, permitiéndonos encontrar y analizar detalles, tendencias y problemas que a simple vista no encontraríamos
- Demarc [42] Un proyecto similar a ACID
- Hogwash [43] es un sistema de respuesta en tiempo cuasireal, tirando o modificando los paquetes que considera malignos para que no lleguen a su destino. Según algunas definiciones, Hogwash junto con Snort crean un *Signature-based Firewall* — Un firewall basado en firmas. Está diseñado para correr en capa 2 y ser invisible — Corre sin siquiera tener configurado el stack de TCP/IP del kernel, es extremadamente eficiente (con una PC estándar se da abasto para una red a 100Mbps). Está aún *muy* en desarrollo, pero es una idea maravillosa.

5 Por fin...

Tras quién sabe cuánto tiempo y esfuerzo dedicados a leerme o escucharme hablar —esfuerzo muy loable— te tengo una maravillosa noticia: Ya acabamos. Si tienes alguna duda que no haya resuelto, si quieres más detalle en algún punto, ahora es cuándo para solicitarlo.

Quedo a tus órdenes. Mi dirección de correo es gwolf@campus.iztacala.unam.mx, y te repito, el texto completo de este tutorial lo tendrás disponible y (hasta donde pueda mantenerlo) actualizado en http://www.gwolf.cx/seguridad/recom_seg_linux

Referencias

[1] RedHat, http://www.redhat.com

[2] LinuxPPP, http://www.linuxppp.com

[3] Conectiva, http://es.conectiva.com

[4] Hispafuentes, http://www.hispafuentes.com

[5] Mandrake, http://www.mandrakesoft.com

[6] Turbolinux, http://www.turbolinux.com

[7] SuSE, http://www.suse.com

[8] Caldera, http://www.caldera.com

[9] Debian, http://www.debian.org

[10] Free Software Foundation, http://www.fsf.org

[11] Contrato social de Debian, http://www.debian.org/social_contract.es.html

[12] Open Source Definition, http://www.perens.com/OSD.html

[13] Slackware, http://www.slackware.com

[14] Descripción del sistema de arranque de Slackware, http://www.slackware.com/config/init.php

[15] muLinux, http://sunsite.dk/mulinux/

[16] floppyFW, http://www.zelow.no/floppyfw/

[17] Linux Router Project, http://www.linuxrouter.org

[18] Trinux, http://trinux.sourceforge.net/

[19] Hal91, http://www.itm.tu-clausthal.de/~perle/hal91/

[20] LODS, http://www.freecolormanagement.com/lods/

[21] tomsrtbt, http://www.toms.net/rb/

[22] DemoLinux, http://www.demolinux.org/

[23] OpenBSD, http://www.openbsd.org

[24] Grupo de usuarios de OpenBSD de México, http://www.openbsd.org.mx

[25] Proyecto ProFTPD, http://www.proftpd.net/

[26] El cliente de correo Mutt, http://www.mutt.org

[27] Proyecto DJBDNS, http://cr.yp.to/djbdns.html

[28] Lista de anuncios de RedHat, redhat-announce-list-admin@redhat.com

[29] Proyecto Abacus, de Psionic, http://www.psionic.com/abacus

[30] Tutorial de Portsentry, http://www.gwolf.cx/seguridad/portsentry/

[31] Tutorial de Logcheck, http://www.gwolf.cx/seguridad/logcheck/

[32] Tutoriales del Departamento de Seguridad en Cómputo, http://www.seguridad.unam.mx/Tutoriales/tutoriales.html

[33] Proyecto Bastille Linux, http://www.bastille-linux.org/

[34] Proyecto OpenWall, http://www.openwall.com/

[35] lista Bugtraq, bugtraq@securityfocus.org

[36] Tutorial de IPTables de Rusty Russell, http://www.telematik.informatik.uni-karlsruhe.de/lehre/seminare/LinuxSem/downloads/netfilter/iptable HOWTO.html

[37] Página de manual de IPChains, http://www.rt.com/man/ipchains.8.html

[38] Tripwire, http://www.tripwire.org/

[39] Snort - The Open Source Network IDS, http://www.snort.org/

[40] Snorticus, http://snorticus.baysoft.net/

[41] Analysis Console for Intrusion Databases (ACID), http://www.andrew.cmu.edu/~rdanyliw/snort/snortacid.html

[42] Demarc, http://www.demarc.org/

[43] Hogwash, http://hogwash.sourceforge.net/