

# Seguridad en Linux

**Horst H. von Brand**

**Universidad Técnica Federico Santa María  
Departamento de Informática**

**vonbrand@inf.utfsm.cl**

**Casilla 110-V  
Valparaíso  
Chile**

El tema *seguridad* generalmente se trata como un tema puramente técnico, altamente complejo, con gran cantidad de aspectos a considerar. Daremos una rápida pasada por los temas centrales, el tema en sí es demasiado amplio para poder cubrirlo en su totalidad en espacio tan breve. Se incluyen las referencias (más que nada a la misma red) que personalmente he encontrado me han sido las más útiles.

La tesis del presente trabajo es que la seguridad no es un tema técnico, sino que es fundamentalmente un tema humano. Veremos que las razones para requerir seguridad son los valores que los humanos asignan a ciertos recursos y su control. Los problemas de seguridad más serios dependen a su vez de características humanas, ya sea alguna clase de problema mental de parte de los atacantes típicos como falso sentido de seguridad de parte de los usuarios y administradores. Finalmente está el problema de que a la función de seguridad no se le asigna la importancia (y los recursos) que necesita.

## **1. ¿Qué entendemos por *seguridad*?**

Hay variadas definiciones de seguridad, que difieren más que nada en el énfasis en lo que se define como aquello que debe protegerse. Aquí usaremos la división que puede considerarse clásica:

**Confidencialidad:** Se refiere a que los recursos del sistema están accesibles sólo a partes autorizadas (se conoce también como *secreto*)

**Integridad:** Los recursos sólo pueden ser modificados por partes autorizadas de maneras autorizadas

**Disponibilidad:** Los recursos están disponibles para las partes autorizadas en forma expedita (como definido por los requerimientos del sistema). El no cumplir con esta característica se denomina *negación de servicio* ("denial of service", DoS)

Como se indicó, hay diversas definiciones de seguridad. Son básicamente divisiones y particularizaciones de los anteriores, bastante generales y abstractos. Por ejemplo, se habla de *no repudiación*, significando que quien envió o recibió un mensaje no pueda negar eso a futuro. Igualmente, se habla de *privacidad* como distinto de *confidencialidad*, con significado de proteger la confidencialidad de un usuario (vale decir, su identidad) y no la de los datos.

Los anteriores implican alguna forma de *identificación* (indicar la identidad de las partes) y *autenticación* (verificar que las partes realmente son quienes dicen ser). Nótese que esto va en ambas direcciones, si requiero un servicio (por ejemplo, acceder a una máquina remota) interesa al servicio identificarme y verificar que soy quien digo ser. Al revés, me interesa identificar el servicio al que me conecto (por el nombre o dirección IP de la máquina que quiero usar, e indicando el port del servicio que me interesa), y verificar que es realmente el servicio solicitado (en este caso, que no sea suplantado ni el servicio que requiero ni la máquina en la que corre). Frecuentemente se lista *auditabilidad*, y a veces también *control de acceso* y *autenticidad*.

## 2. Definiendo la seguridad requerida

Sea cuales fueren los requerimientos de seguridad, es importante identificar y explicitar los objetivos de seguridad del sistema, sin importar cómo se agrupan. De esta forma se

estará en condiciones de determinar si se están cumpliendo, y en caso de no ser así, dónde están las falencias.

## **2.1. Algunas máximas**

Antes de entrar en el detalle de lo que se desea, deben tenerse en cuenta un par de puntos importantes:

- Cuando se habla de seguridad, normalmente se considera sólo el caso de un atacante que accede al sistema sin autorización. Siendo éste un caso importante, no debe olvidarse el efecto que pueden tener equipos que fallan, cortes de energía, fenómenos naturales, error humano, ...
- La seguridad absoluta no existe. Lo más que se puede hacer es disminuir la probabilidad o el impacto de situaciones no deseadas.
- Toda seguridad tiene un costo. Este costo no es sólo en términos de equipamiento (máquinas replicadas, alarmas contra intrusos, o sistemas de respaldo), el costo más importante (que suele olvidarse) es en términos de comodidad de uso (mantener sincronizadas las réplicas, tener a alguien que deba presentarse en corto plazo si suena la alarma, o el no poder usar el sistema mientras se hacen los respaldos mensuales).
- La cadena es tan fuerte como su eslabón más débil. Es inútil asegurar exhaustivamente el acceso al sistema donde están los datos a proteger contra acceso no autorizado si éstos aparecen en forma de listados en la basura.

En lo que sigue se discute más que nada el caso en que se pierdan datos. El mismo tipo de análisis puede (y debe) aplicarse a las demás dimensiones de la seguridad que interesen para una situación dada. Por ejemplo, es perfectamente factible que ciertos datos tengan poco valor en sí, pero que deban ser protegidos cuidadosamente contra divulgación o modificación no autorizadas.

El análisis que se haga deberá considerar como parte del "valor" de los recursos en cada una de las dimensiones condiciones adicionales, como pueden ser obligaciones legales

(hay leyes que regulan el uso que pueden hacer las instituciones financieras con los datos de los clientes), contractuales (una empresa de ingeniería estará obligada a guardar reserva de los proyectos de sus clientes), e incluso éticas (los datos de los pacientes de un médico son confidenciales). Un punto de importancia es el problema de imagen pública: En Internet, la imagen que tiene una empresa está dada por su reputación casi exclusivamente. Como el grueso público conoce aún menos de seguridad que los especialistas, incidentes como el reciente cracking de Microsoft, donde el (o los) atacantes habrían tenido acceso al repositorio de código fuente resultan alarmantes para los potenciales clientes, que de muchas empresas sólo conocen su página web. Además, de ser usada una máquina como herramienta para atacar a otras, en algunas legislaciones el responsable de la máquina usada para perpetrar un ataque puede ser legalmente responsable de éste, al menos en parte.

## **2.2. Recursos a proteger**

Es importante tener claro cuáles son los recursos que deben protegerse. Igualmente, es importante determinar los riesgos a que están sujetos, y el impacto que tales riesgos puedan tener.

## **2.3. Valor de los recursos para nosotros**

Típicamente los datos almacenados en un sistema son mucho más valiosos que el sistema mismo. Debe determinarse el costo de reponer los recursos bajo estudio (suponiendo que se pierden totalmente), y alguna idea de la probabilidad de tales desgracias. Esto determinará el costo que estaremos dispuestos a incurrir para protegerlos de posible pérdida.

## **2.4. Valor de los recursos para atacantes**

Asimismo, debe considerarse el valor que estos recursos puedan tener para un posible atacante. Esto determinará en buena parte el costo en que éste estará dispuesto a

incurrir para acceder a ellos. Esto, a su vez, da una pauta complementaria para determinar cuánto esfuerzo invertir en su protección.

### 3. Clasificación de riesgos

Hay varios criterios posibles para clasificar riesgos. Una clasificación simple es según el la fuente del problema:

- Hechos fortuitos
- Ataques externos (a través de la red)
- Ataques internos (usando acceso directo a los sistemas, o privilegios de usuario normal)

Para nuestros efectos, los hechos fortuitos son los menos interesantes (Tal vez por ser menos "computacionales", pero no por eso menos destructivos: Recuérdese la inundación de ECOM, en su época la empresa que manejaba los sistemas de buena parte de Chile en sus máquinas. Una combinación de lluvias intensas, sala de máquinas subterránea, y respaldos guardados en el mismo subterráneo casi acaba con la empresa).

Los ataques externos son los más temidos (el cine ha glorificado la figura del adolescente con un PC, un modem, y demasiado tiempo libre). Sin embargo, no son éstos los más frecuentes, ni lejos los más dañinos.

Los ataques internos son lejos los más frecuentes (las estadísticas indican que son cerca de un 85% de los casos de ataques que se registran).

Un tipo de ataque particularmente insidioso es lo que se llama *ingeniería social*. Se refiere a engañar a una persona para que entregue información (por ejemplo, llamando por teléfono e indicando que es el supervisor, solicitando alguna clave "olvidada" con voz autoritaria) o inicie la acción destructiva (como en el caso del infame **I LOVE YOU**, ¿quién es capaz de resistir leer una supuesta declaración de amor de parte de algún miembro del sexo opuesto al que conoce en la oficina?).

Una clasificación adicional se puede hacer por los objetivos de los atacantes:

- Ataques dirigidos a un blanco específico. Esto incluye desde espionaje industrial hasta el simple desafío por ingresar a sitios protegidos, pasando por vandalismo puro (destruir páginas web, borrar o destruir datos, etc).
- Ataques al azar. En esta clasificación cae la creación de virus y gusanos.
- Reclutamiento para ataques dirigidos. Una de las técnicas nuevas (se estrenó a principios del 2000) es penetrar un gran número de máquinas independientes, con la intención de usarlas para lanzar ataques de saturación coordinados ("Distributed Denial of Service", DDoS) contra blancos escogidos. De éstos fueron víctimas entre otros Amazon (<http://www.amazon.com>) y eBay (<http://www.ebay.com>).

Otra clasificación útil considera las técnicas usadas. Se puede distinguir entre:

- Verdaderos *crackers*, con algún conocimiento de los sistemas atacados. Son los más peligrosos, pero afortunadamente los menos. Sin embargo, las vulnerabilidades que éstos descubren, y las herramientas que desarrollan para aprovecharlas, normalmente se difunden rápidamente entre la segunda categoría.
- Los llamados *script kiddies* (traducción libre sería "mocosos que usan scripts"; análisis detallado de un incidente que permitió obtener luces de los motivos y grado de conocimiento de los atacantes se encuentra en Know Your Enemy: Motives (<http://www.enteract.com/~lspitz/motives/>)), son típicamente adolescentes cuya entretención consiste en coleccionar y usar herramientas que detectan vulnerabilidades y las explotan. La técnica básica es revisar y clasificar grandes números de sistemas (generalmente al azar), con la intención de crear bases de datos de máquinas potencialmente vulnerables. Al descubrirse alguna nueva vulnerabilidad, usan la base de datos para identificar blancos prometedores. Normalmente nada saben de los sistemas atacados (clásico fue el caso en que uno de éstos se hizo **root** en una máquina Sun (corriendo Unix), para dar el comando **format c:**). Se caracterizan por el uso de herramientas automatizadas para identificar víctimas potenciales, e incluso para penetrarlas (de allí el apelativo).

Nótese que ésta es una clasificación más que nada por *modus operandi*, un *cracker* buscando montar un ataque DDoS probablemente use técnicas de *script kiddie* para reclutar máquinas con las cuales llevar a cabo su propósito.

## 4. Flora y fauna

Ninguna visión general de la seguridad estaría completa sin una breve definición de algunos términos.

### 4.1. Virus

Un virus es un programa que no tiene existencia propia. Depende de su funcionamiento de algún programa huésped cuyo control toma. Los virus se propagan a través de infectar otros programas en la misma máquina, y a través de copias de programas infectados de máquina a máquina.

### 4.2. Gusanos

Un gusano ("worm") es un programa independiente, que infecta máquinas (no programas individuales). Se propaga a través de la red, y toma el control de la máquina víctima usando alguna vulnerabilidad particular. Desde ésta monta ataques a otras máquinas.

### 4.3. Troyano

En rigor, el nombre es Caballo de Troya. Es un programa que aparenta tener una función útil, pero en realidad tiene otra siniestra, tal como el ejemplo mitológico del que toma el nombre. Su difusión se debe a usuarios que lo usan o comparten por su (supuesta) utilidad. Frecuentemente aparecen como versiones nuevas de programas ampliamente usados.

Como el caballo original era de madera, sirve de (débil) pretexto para el nombre de la sección actual.

## **5. Calidad de software y seguridad**

Para que se vea comprometida la seguridad de un sistema algún agente debe usar sus privilegios en forma no autorizada. El caso más frecuente es el en el que algún programa es engañado para que efectúe acciones no autorizadas.

Ejemplos típicos de esto han sido los virus que se han distribuido por correo, aprovechando las facilidades de ciertos sistemas de correo para ejecutar automáticamente código que viene en mensajes, y los similares con aplicaciones de oficina. Éstos caen más bien en el ámbito de diseño básicamente mal hecho, que debiera rehacerse. Claro que el costo de tal cambio es inmenso.

La otra gran fuente de problemas han sido errores de programación (un buen resumen es *Secure Programs HOWTO* (<http://www.dwheeler.com/secure-programs/>), que incluye discusión de las últimas clases de vulnerabilidades descubiertas, y *Secure UNIX Programming FAQ* (<http://www.whitefang.com/sup/>), que incluye referencias a una gran cantidad de FAQs adicionales). La pregunta obvia entonces es la relación entre calidad de software (ausencia de fallas) con su vulnerabilidad frente a ataques externos.

Como ya dijo Dijkstra: “ Pruebas de software pueden demostrar la presencia de errores, no su ausencia ” simples pruebas nunca permitirán lograr software seguro. El problema es que los atacantes buscan activamente provocar situaciones extremas (que nunca ocurrirían bajo uso normal) precisamente con la intención de aprovechar errores en situaciones pasadas por alto en el diseño, la programación, y las pruebas. Las técnicas a emplear entonces para asegurar programas y sistemas son de auditoría más que pruebas en el sentido tradicional. Herramientas al efecto incluyen *ITS4* (<http://www.cigital.com/its4/download.html>), aunque se pueden usar también con buenos resultados opciones de los compiladores para obtener más mensajes de advertencia, y el uso de bibliotecas especiales de verificación. Sin embargo, la herramienta principal sigue siendo la persona que revisa el código. En este sentido, los



sistemas de fuentes abiertas (<http://www.opensource.org>) tienen una ventaja, dado que hay más posibles auditores. Por el otro lado, los fuentes pueden ser también revisados por los malandrines. En todo caso, esto no es tanta desventaja como parece a primera vista, muchas de las vulnerabilidades se pueden encontrar simplemente tomando los programas ejecutables y sometiéndolos a entradas absurdas (nombres de archivo inmensamente largos, o que incluyen caracteres de control, etc).

Por ejemplo, al enterarme del tipo de problemas de seguridad que podría introducir el rebalsar arreglos en programas que corren con mayor privilegio, y teniendo a mano la en ese entonces última versión del programa de conexión vía telefónica `dip` (<ftp://metalab.unc.edu/pub/Linux/system/network/serial/dip/dip337o-uri.tgz>) revisé el uso de strings y arreglos en éste. Descubrí varios usos dudosos, los corregí y envié un parche al encargado, quien lo integró a la versión oficial. Poco después se descubrieron problemas de seguridad en este paquete, que mis parches resolvían. No habría sabido cómo construir un ataque usando los problemas encontrados, y tampoco sé (ni me interesa particularmente saber) cuál de los errores llevaba al problema de seguridad.

Un punto de interés es que el tipo de errores de programación que se reflejan en problemas de seguridad más comunes suelen dar lugar a caídas de los programas ante pruebas con datos al azar, como en `Fuzz` (<http://www.cs.wisc.edu/~bart/fuzz/fuzz.html>) (hay una versión actualizada de la herramienta del caso en `Source Forge` (<http://fuzz.sourceforge.net>)). En el año 1990, se probaron entre 49 y 85 utilitarios de cada uno de 5 sistemas Unix comerciales, con el resultado de 25 a 33% de fallas. Las pruebas se repitieron en 1995 en 9 sistemas, incluyendo una distribución de Linux y los utilitarios de GNU (<http://www.gnu.org>). Se probaron entre 47 y 80 programas en cada ambiente, con tasas de fallas entre 15 y 43% en los Unix comerciales. Esto se compara con un 9% para Linux y un 6% para GNU. Lo preocupante es que en los casos donde se pudieron determinar los errores en versiones del mismo sistema en 1990 y 1995, muchos de los errores originales seguían en el código, a pesar de estar ampliamente disponibles los resultados y las herramientas (<ftp://grilled.cs.wisc.edu/fuzz/>) usadas. Eso sí, estos resultados son a lo más indicativos de la calidad general del software de cada sistema, no indican necesariamente el nivel de seguridad de otro software que se distribuye con él. Las pruebas de 1995 no hicieron mella en los programas servidores probados.

Para completar el cuadro, a principios del 2000 se repitió este tipo de pruebas sobre

Windows NT. La tasa de fallas fue de 100%.

## **6. ¿Razones para dormir sobre los laureles?**

Diariamente se escucha de virus que atacan computadores personales, pero no se ha sabido de virus en estado salvaje que ataquen a Linux (o a alguna versión de Unix, si es por eso). Las razones son más bien simples:

- En Unix, los usuarios están protegidos el uno contra el otro, y el sistema está protegido contra los usuarios. Este es un prerequisite básico de todo sistema multiusuario.
- Más que nada por usar el disco en forma eficiente, hay una única copia de los ejecutables de uso general. Rara vez los usuarios tienen ejecutables de uso continuo, y aún menos los comparten con los demás.
- Hay gran variedad de sistemas Unix, en particular, muchas distribuciones y versiones de Linux en uso corriente. Esto complica la labor del atacante.
- Las máquinas Unix suelen tener administración más cuidadosa que otras alternativas (Mal que mal, Linux normalmente lo instala uno mismo, aún es raro ver sistemas preinstalados).

Sin embargo, el uso de Linux está creciendo en forma muy importante. Muchas máquinas Linux están en labores de servidor, conectadas en forma permanente a la red. Esto las hace vulnerables, y las transforma en blancos deseables (tomar el control de una máquina a la que no se tiene acceso no tiene mucha gracia).

Sin embargo, Linux se hace cada vez más popular. Un estudio de IDC (<http://www.idc.com>) sobre difusión de sistemas operativos de servidores citado por CNET (<http://news.cnet.com/news/0-1003-200-1549312.html>) da un 16% a Linux en 1998, y un 25% en 1999. Esto lo hace cada vez más interesante como blanco a ser explotado. Como Linux es fácil de conseguir, y corre en máquinas de bajo costo, un atacante potencial puede montar una máquina en la cual estudiar el sistema sin ninguna

interferencia sin mayores problemas. Como ya se indicó, la disponibilidad de los fuentes del sistema es un arma de doble filo.

Sea como sea, la comunidad criptográfica (<http://www.faqs.org/faqs/cryptography-faq/>) hace tiempo llegó a la conclusión que la única manera de construir un sistema criptográfico seguro es que éste sea conocido en su funcionamiento, y que la seguridad dependa de una parte pequeña, fácilmente cambiabile (la clave). Esto asegura una amplia revisión de parte de los profesionales en la materia, con lo que potenciales debilidades serán encontradas y corregidas. Los "sistemas secretos" (lo que se ha dado en llamar "security through obscurity", seguridad a través de la obscuridad) son seguros mientras sean realmente secretos en su funcionamiento, dejan de ser seguros en el momento en que dejan de ser secretos. Las mismas observaciones son aplicables a los sistemas computacionales que nos interesan.

## **7. ¿Divulgar o no divulgar?**

Una de las discusiones continuas en el área es qué hacer al descubrir una vulnerabilidad. Los hay quienes insisten que debe dar detalles completos del problema (incluso programas que aprovechan la vulnerabilidad en forma automática) como una manera de presionar a los proveedores para que corrijan las vulnerabilidades.

Exponente saliente de esta tendencia es la lista de correo BugTraq (<http://www.securityfocus.com/forums/bugtraq/faq.html>). Esto tiene la lamentable consecuencia de poner en manos de personas inescrupulosas el equivalente de un arma cargada y con el seguro quitado. Se indica que el dar sólo los detalles de la vulnerabilidad sin demostrar su posible uso permitiría soslayar el problema, con lo que tienen razón. Sin embargo, el entregar una herramienta de penetración lista para ser usada es de dudosa ética, en mi opinión.

Otro extremo lo da CERT (<http://www.cert.org>), cuya política es dar a conocer vulnerabilidades en términos muy generales, y sólo una vez que el proveedor haya puesto una solución al problema.

En todo caso, las políticas de ambos están convergiendo, en que BugTraq solicita que sólo se publiquen las vulnerabilidades después de que esfuerzos serios para alertar al

proveedor hayan resultado infructuosos, y después de que haya transcurrido un plazo razonable para que se encuentre una solución. CERT tiene ahora una política similar de publicar detalles de vulnerabilidades para las cuales el proveedor no haya dado solución en un plazo prudente.

Ambos mantienen extensas colecciones de reportes de vulnerabilidades y también resúmenes de vulnerabilidades más explotadas. Son recursos importantes para tener una visión general de los problemas y sus soluciones.

## **8. Algunas referencias útiles**

Un excelente resumen de los tópicos involucrados en seguridad se encuentra en las páginas de temas básicos de (<http://www.securityfocus.com>). Más información se encuentra en Security Portal (<http://www.securityportal.com>). Referencias específicamente sobre Linux tiene Linux Security (<http://www.linuxsecurity.com>). La página semanal de noticias Linux Weekly News (<http://www.lwn.net>) (aparece los jueves) incluye una sección sobre anuncios de actualizaciones y una sección dedicada a la seguridad. Buenos resúmenes, con gran cantidad de referencias, son los Security-HOWTO y Secure-Programs-HOWTO, ambos se encuentran en Linux Documentation Project (<http://MetaLab.unc.edu/LDP/>) (del cual mantiene traducciones al castellano el proyecto LuCAS (<http://lucas.hispalinux.es/>), además de bastantes documentos propios). Las distribuciones en sus páginas mantienen links a temas de seguridad y actualizaciones.

Herramientas para asegurar sistemas se encuentran por ejemplo en la distribución Bastille Linux (<http://www.bastille-linux.org/>). Hay discusiones también en Armoring Linux (<http://www.enteract.com/~lspitz/linux.html>) y en Armoring Solaris (<http://www.enteract.com/~lspitz/armoring.html>), y en el FAQ (<ftp://rtfm.mit.edu/pub/faqs/computer-security/most-common-qs>) de `comp.security.unix` (`news:comp.security.unix`). Las distribuciones de Linux incluyen herramientas como `tcp_wrappers` y `tripwire`, además de incluir versiones más seguras de algunas herramientas básicas que las que suelen distribuir los Unix comerciales. Además, suelen ser mucho más ágiles en la distribución de

actualizaciones, particularmente las debidas a problemas de seguridad.

Un tema muy relevante para lo que es confidencialidad (y también relacionado con autenticación y no repudiación) es la criptografía. Este es un tema extremadamente técnico, y un área donde abundan los charlatanes. Lectura obligada es *Snake Oil Warning Signs: Encryption Software to Avoid* (<http://www.interhack.net/people/cmcurtin/snake-oil-faq.html>). En general, hay muchísima información en los FAQ ("Frequently Asked Questions", preguntas frecuentes) de Usenet. Estos (y mucha otra documentación de referencia general) se encuentran en Internet FAQ Consortium (<http://www.faqs.org>).

Temas de análisis forense y herramientas al efecto (cómo determinar qué pasó durante un incidente) se encuentran entre otros en *fish* (<http://www.fish.com/security/>). Un ejemplo de uso de técnicas forenses se encuentra en *Know your enemy: A forensic analysis* (<http://www.enteract.com/~lspitz/forensics.html>).

