

Seguridad en GNU/Linux

por Arturo 'Buanzo' Busleiman <buanzo@buanzo.com.ar>
www.buanzo.com.ar, para SoloLinux!

En esta oportunidad tengo el agrado de acercarme a ustedes con una temática mucho mas técnica, parece que mis ruegos a Daniel Fabero han dado resultado, y espero darles un artículo de calidad, como mis anteriores artículos técnicos: Consola y Squid.

Hoy vamos a hablar de un tema terriblemente interesante, que da que hablar tanto a defensores como detractores del Software Libre, y con buenos motivos: GNU/Linux puede securizarse MUY bien, por un simple motivo: tenemos acceso al código fuente. Punto. Fin de la discusión. No podemos confiar en algo que no sabemos (exactamente) qué hace ni cómo lo hace. Bueno, yo al menos no puedo. Muchas empresas se escudan detrás del dicho "Pero yo necesito una empresa detrás, que me de soporte, por eso no utilizo Software Libre.". En verdad, la traducción de esta frase es: "Necesito a quien poder hecharle la culpa de mis errores". El problema es que la Seguridad requiere Responsabilidad, Conocimiento+Experiencia e Investigación, y no todo empleado o empresa considera importante tomar cartas en dichos asuntos, ni dedicar recursos (tiempo, dinero) a los mismos.

Voy a tratar de desmitificar algunas cuestiones acerca de la seguridad en GNU/Linux y aclarar algunos puntos por demás interesantes, como por ejemplo:

- * ¿Es realmente tan seguro?
- * ¿Existen mas vulnerabilidades que en Windows?
- * ¿Los parches salen más rápido?
- * ¿Es mejor Gentoo que Debian?
- * ¿Es más fácil explotar una vulnerabilidad en GNU/Linux?
- * ¿Es el mejor entorno para aprender técnicas de hacking?

Adicionalmente, hablaremos acerca de la securitización básica de nuestra estación de trabajo GNU/Linux, y de algunas tareas que cualquier interesado en la seguridad y la tecnología en general deberían realizar día a día.

Veamos, pues, algunos de esos puntos en mas detalle. Este formato de establecer items de interés/contenido al principio del artículo me resulta muy útil, ¡espero que a ustedes también!

¿Es realmente tan seguro?

Veamos, ¿Cuál es la principal diferencia entre Windows y GNU/Linux? El acceso al código fuente en forma irrestricta. La posibilidad de poder leer el código le permite a los expertos encontrar más fácilmente vulnerabilidades de cualquier tipo y reportarlas a los autores de la aplicación, librería o del Kernel mismo. En el caso de Windows, existen casos en los cuales Microsoft da el acceso al código fuente, pero NO SE PUEDE HACER NADA con el conocimiento adquirido al analizar el mismo, ya que se firman 643813681 acuerdos de confidencialidad, de no divulgación, de no aprovechamiento y demases, que anulan totalmente el propósito de compartir el código fuente. Cuando Microsoft comparte

el código fuente, lo hace generalmente con el propósito de facilitar el desarrollo de aplicaciones que ellos están interesados en promocionar, o sea, lo hacen pura y exclusivamente por beneficio COMERCIAL de ellos, olvidándose de beneficiar a sus clientes durante el proceso. No los juzgo. Al fin y al cabo, es el único modelo comercial que pueden llegar a comprender.

Si nos focalizamos en GNU/Linux, el acceso al código fuente no solo permite encontrar más fácilmente vulnerabilidades, sino que logra que sea aún más sencillo descubrir versiones troyanas de una cierta aplicación, o intentos de troyanizar en el repositorio mismo de la aplicación, como ha sucedido en dos o tres oportunidades. En tales ocasiones, la cantidad de versiones troyanizadas descargadas e instaladas fueron mínimas, gracias a la rápida acción de los miembros de la comunidad.

Por ejemplo, si un elemento central o muy utilizado en Windows, tanto por el sistema como por aplicaciones de terceros, tiene un problema, todas las aplicaciones que lo utilizan se verán afectadas, y la propagación del exploit puntual será descubierta cuanto se llegue a una masa crítica de servidores o sistemas infectados mucho mayor a la "aceptable". El verdadero problema es que Microsoft, o el vendedor en cuestión, se preocupará por intentar corregir el problema en ese momento: cuando la masa crítica sea... verdaderamente crítica. Se ha sabido de varios casos en los cuales el vendedor sabía del problema, pero no lo consideraba una "amenaza para la seguridad" hasta no ver exploits pululando por Internet. Claro, como no hay acceso al código fuente, y nadie SABE que la vulnerabilidad está allí, ¿por qué preocuparse en corregirla YA? Esto es totalmente inaceptable si estamos interesados en la calidad de nuestro software. Claro, publicar el código fuente es un acto de valentía que aplaudo de pie. Considero que abrir el código es un paso gigantesco hacia la seguridad en el software. Y digo esto desde un punto de vista puramente técnico, y no filosófico.

En resumidas cuentas, la seguridad es un estado mental. No podemos darnos el lujo de ni siquiera pensar que el Open Source es seguro intrínsecamente, pero si podemos afirmar que el acceso al código fuente otorga una gran ventaja sobre el software de código cerrado. Una auditoría permanente por cientos, miles de expertos del mundo no se compara con el que pueda hacer en forma privada una empresa o un programador independiente de software propietario.

¿Existen mas vulnerabilidades que en Windows?

Esta no es la pregunta que deberíamos hacernos, ya que no tiene sentido. Es una "causa perdida" esta cuestión, ya que tener más o menos vulnerabilidades depende de dos factores:

- 1) El conocimiento de los programadores acerca de las técnicas de programación seguras
- 2) El conocimiento que se posea acerca de vulnerabilidades y formas de explotarlas al momento de diseñar y desarrollar dicho software.

Si, ambos factores podemos tomarlo como uno mismo, pero en el mundo real suelen darse las dos situaciones por separado: gente que sabe programar, y gente que sabe explotar/vulnerar. Son pocos los casos de gente que sabe hacer las dos cosas bien, y es por este motivo que separo en dos factores.

Existe gran documentación acerca de deficiencias a la hora de diseñar y escribir software, con prácticas comunes para evitar diferentes tipos de problemas. David Wheeler nos tiene ya acostumbrados a documentación de altísima calidad, que invito a todos ustedes a leer en su sitio, <http://www.dwheeler.com/>

Cualquier programador interesado en escribir software, o cualquier docente de la materia programación, debería prestar singular importancia a este tipo de documentación.

Ahora, continuando con nuestra pregunta, decíamos que no es la que deberíamos hacernos. Yo creo que la pregunta es la siguiente: ¿Se descubren más vulnerabilidades en GNU/Linux que en Windows?

La respuesta es si. ¡Claro que se descubren más vulnerabilidades! ¡TENEMOS el código fuente! Lo puede analizar más gente, y el parche puede ser distribuido a las pocas horas de descubrirse el problema. Sinceramente, prefiero que se descubran vulnerabilidades, ya que significa que hay un grupo de gente grande preocupado por encontrarlas, y corregirlas.

Una vez leí un artículo acerca de cual es el proceso completo, de principio a fin, en Microsoft, para llegar a un parche desde el momento en que se reporta la vulnerabilidad. Ellos dicen que están tardando tanto en liberar un parche porque lo deben hacer para muchas arquitecturas diferentes, versiones de windows diferentes, etc, porque deben evitar que los parches provoquen inestabilidad (no, no es un chiste) en los sistemas donde sean aplicados, o incluso que abran nuevos agujeros. El problema es que esto sigue sucediendo, ya que tal vez el parche funcione correctamente cuando se lo testea contra Windows en si mismo, pero la mayoría de los problemas actuales surgen de la interacción de la dupla "parche" y "software de terceros", como por ejemplo, un antivirus.

Hace poco, con la salida del último pack de correcciones acumulativas para Windows 2003, al instalarlo los usuarios descubrieron que el login tardaba varios minutos. Al poco tiempo, se descubrió que era un problema que se le generaba al antivirus instalado en dichas estaciones de trabajo.

Esos son los riesgos del software de código cerrado, especialmente en software de base como sistemas operativos.

¿Los parches salen más rápido?

Si, salen más rápido porque el proceso no comienza solo cuando una vulnerabilidad es reportada. A veces simplemente se descubre una vulnerabilidad y se crea el parche cuando un programador se encuentra mejorando la aplicación, o analizándola con otros propósitos. O sea, se emiten parches con mayor frecuencia, y no necesariamente son todos parches a problemas de seguridad.

Por otra parte, el equipo de gente que puede estar trabajando en solucionar una vulnerabilidad es mayor. Incluso se han dado casos de personas totalmente independientes que encuentran un mismo problema y lo solucionan de maneras diferentes. En esa instancia, el autor solicita a los miembros de la lista de correos de desarrolladores que prueben los parches y hagan comentarios al respecto. Es más, es muy

común ver el código del parche como parte del cuerpo de un emilio, siendo comentado como si de un parrafo de alguien se tratara. Definitivamente, somos muy geeks.

¿Es mejor Gentoo que Debian?

En verdad aquí estoy hablando de utilizar software viejo con menor funcionalidad, pero más testeado contra problemas, que versiones nuevas con mayor funcionalidad. En MI caso particular, prefiero utilizar software más nuevo, siempre, excepto que estemos hablando de dos generaciones totalmente diferentes de un mismo software, donde tal vez el diseño interno haya cambiado y evolucionado de forma tal que no sea de nuestro agrado, y preferamos la versión anterior.

Ahora, a nivel de la seguridad, el utilizar software viejo más testeado trae la aparente ventaja de la estabilidad y seguridad, pero con menor funcionalidad.

El software más nuevo potencialmente tiene en su haber el conocimiento adquirido día a día por el o los programadores, sumado a los reportes de seguridad y parches enviados por otros miembros de la comunidad, y mayor funcionalidad.

Es el famoso caso de la balanza de la Funcionalidad versus la Seguridad. Se suele suponer que cuanto mayor funcionalidad posea una aplicación, menor Seguridad poseerá, ya que la mayor funcionalidad se equipara a una mayor cantidad de código, por ende, mayores posibilidades de encontrar una vulnerabilidad. Como ven, se está hablando de una simple cuestión de estadística, dónde no se tiene en cuenta el proceso de auditoría continuo al que es sometido el código a medida que se lo va extendiendo.

Ahora, piensen lo siguiente: al momento de auditar una aplicación, se aplica el conocimiento que se posea en ese momento acerca de como explotar las vulnerabilidades que se encuentren. Si una no se puede explotar, no es una vulnerabilidad a la que se le de prioridad por corregirse. Eso suele quedar sin corregir. Pasan los meses, o años en algunos paquetes de Debian, y ahora existen técnicas que permiten explotar esa previamente inexplotable vulnerabilidad. ¿Que el software sea viejo garantiza la seguridad? No. ¿Y que sea nuevo, la garantiza? Tampoco, pero al menos tenemos mayor funcionalidad, e integrar parches a versiones nuevas es más sencillo que realizar un backport de parches hacia versiones más viejas.

Es por esto que, en mi opinión, yo prefiero utilizar Gentoo o cualquier distribución basada en fuentes, como Sourcemage, o distribuciones más actualizadas, como Ututo-e.

Claro, ustedes pueden decir: Puedes seleccionar otro repositorio para utilizar con el APT de Debian. ¿Y cual sería el sentido de utilizar Debian, entonces, si no utilizamos el repositorio oficial de aplicaciones? ¿Enviamos a la basura el trabajo que realizan?

¿Es más fácil explotar una vulnerabilidad en GNU/Linux?

Aquí en verdad no hablo de la complicación intrínseca de desarrollar un exploit contra un agujero puntual, sino de las técnicas disponibles de análisis y explotación propias de cada plataforma, en comparación con GNU/Linux.

Hace poco, durante un evento de seguridad informática en Buenos Aires donde tuve el

agradado de participar como ponente, presencié una charla acerca de las técnicas para analizar y explotar vulnerabilidades en Windows, y me sorprendí de descubrir que hay herramientas que facilitan mucho el proceso, y que incluso el diseño interno de Windows facilita la explotación. ¡Es tan simple "engancharse" de los diferentes mensajes y eventos de Windows! Pero claro, en contrapartida, en GNU/Linux contamos con el código fuente del Kernel y librerías. Un amigo mio, experto en el desarrollo de exploits, siempre me comentaba el proceso de analizar el código mm (Administración de Memoria) de cada nueva versión del kernel Linux que se liberara.

Claro, si intentamos explotar una vulnerabilidad en una aplicación, en GNU/Linux tenemos tanto aplicaciones propietarias como abiertas, y en resumidas cuentas, la facilidad de explotar depende de la disponibilidad de acceso al código.

Bueno, también depende de la capacidad de quien quiera realizar el exploit, pero ese ya es otro tema, ¿no?

¿Es el mejor entorno para aprender técnicas de hacking?

Bueno, veamos:

- * Es el entorno que más se puede investigar, por tener acceso al código fuente
- * Unix está diseñado para trabajar en redes, en múltiples protocolos.
- * Es el que mayor cantidad de expertos en la materia utilizan.
- * Es el que mayor cantidad de software para programación, análisis forense, depuración y comunicaciones posee.
- * Es el mejor documentado.

Definitivamente, GNU/Linux es el mejor entorno para aprender técnicas de hacking, PERO un buen hacker conoce varios sistemas operativos, por lo que recomiendo que no se limiten a GNU/Linux, ni al software libre, si a lo que investigación se refiere.

SECURITIZACIÓN BASICA

A la hora de aplicar una metodología de securitización, me gusta seguir el modelo TCP/IP, para recordar todas y cada una de las capas en las cuales se debe aplicar planeamiento, seguridad y revisión. Claro, no voy a entrar en detalles aquí acerca del valor de la información, riesgo, etc, ya que vamos a hablar de securitización de nuestro ordenador. Por supuesto, la idea es conocer que herramientas tenemos en GNU/Linux para cada etapa de seguridad, analisis, revisión.

Las capas del modelo TCP/IP, recordemos, son las siguientes:

- * Capa de Aplicación (HTTP, SMTP, FTP, Telnet, etc)
- * Capa de Transporte (TCP, UDP)
- * Capa de Red (IP)
- * Capa de Acceso a la Red (Ethernet, Token Ring)
- * Capa Física (Cable Coaxil, Par Trenzado)

Entonces, lo conveniente es comenzar por la capa inferior, o sea, la Física, y subir hasta llegar a la capa superior, la de Aplicación. Este modelo nos recordará que la seguridad debe aplicarse en todos los niveles, y también nos brinda una metodología para encontrar y solucionar problemas, ya que **guía** nuestros pensamientos.

Veamos, entonces, algunos detalles de cada capa.

Capa Física

¡Bien! No vamos a seguir el modelo TCP/IP tal cual lo he descripto, sino que lo utilizaremos, o abusaremos, como "ayuda-memoria". La Capa Física debe recordarnos que la seguridad física es tan o más importante que la lógica. Si securizamos nuestro servidor con un firewall, si nos roban el ordenador dicho firewall ya no es útil.

Claro, algunos ataques que vengan por el Mundo Real, el Físico, pueden tener su contramedida, o protección, en el mundo lógico. Por ejemplo, hace unos días se hizo pública una vulnerabilidad que afecta a Microsoft Windows en algunas de sus últimas versiones. En este caso, se descubre una vulnerabilidad en la forma en que los controladores de dispositivos USB, como pen-drives, manejan la información interna de dichos dispositivos, de tal forma que un pen-drive correctamente modificado puede provocar la ejecución de código arbitrario con privilegios de SYSTEM, en el ordenador donde se coloque dicho pen-drive.

La contramedida principal es corregir los controladores, claro, pero fíjense que también se podrían deshabilitar los puertos USB desde el BIOS, quitarlos del gabinete, o incluso habilitarlos desde el sistema operativo desde un usuario con privilegios cuando sea necesario utilizarlos.

Entonces, al pensar en securizar la capa física, debemos analizar todas las posibilidades, considerando lo que no sea estrictamente necesario para la función que haya que cumplir. Esto es especialmente cierto en ordenadores de una empresa, o en servidores, donde existe, o debería existir, una política de uso válido.

Entonces, en la categoría "Física", podemos hablar de:

- * Controlar / Restringir el acceso físico al ordenador
- * Quitar / Deshabilitar dispositivos de almacenamiento removible, como disketteras, lectoras de CD/DVD, Zip Drives, USB, etc.
- * Verificar el recorrido de cables de red, tensión, etc. (Una lluvia puede arruinarnos).
- * Instalar filtros de picos de tensión, y estabilizadores. (Un rayo puede sobrecargar la línea telefónica o la de Cablemodem).
- * Verificar que no existan dispositivos adicionales, como por ejemplo, un emisor de FM entre el teclado y el puerto del teclado, que sirve para transmitir toda tecla presionada por radiofrecuencia.

Creo que pueden darse una idea del análisis a realizar. Pasemos a capas superiores, entonces.

Capa de Acceso a la Red

En esta capa debemos preocuparnos por la red física propiamente dicha. Por ejemplo, ya es conocidísima la técnica denominada "sniffing", donde un ordenador con el software necesario (por ejemplo, ettercap, ethereal o tcpdump) puede capturar el tráfico de nuestra red, incluyendo contraseñas, sin mayores complicaciones.

La detección de este tipo de ataque va a depender de varios factores, pero si nos limitamos a la Capa actual, debemos al menos asegurarnos de no estar utilizando hubs para nuestra red, sino switches, que, aunque no imposibilitan el sniffing (ettercap puede capturar tráfico en redes switcheadas), facilitan la detección del mismo. Herramientas como arpwatc detectarán el tipo de ataque utilizado por ettercap para capturar el tráfico de una red switchheada, el cual es conocido como ARP Poisoning.

De la misma forma, la separación en diferentes redes físicas o lógicas (por 802.11q) puede resultar más que útil. GNU/Linux permite trabajar muy cómodamente reglas de filtrado o enrutado para estos casos, mediante iproute2, ebtables e iptables.

Si utilizamos tecnologías wireless, les recuerdo que el filtrado por dirección MAC no es tan útil si un atacante "esnifea" la red, obtiene alguna mac existente, y modifica la propia, con el consiguiente acceso a la red. La seguridad no lo puede dar algo tan trivial como un parámetro tan fácilmente modificable como la MAC.

Capa de Red

En esta capa vamos a hablar de direcciones IP, subredes, rutas y NATs, dejando los puertos para la próxima capa.

En primera instancia, les recuerdo que el comando route e ifconfig ya son obsoletos, y les recomiendo que instalen el paquete iproute2, y aprendan a utilizar el comando "ip", que permite administrar tanto interfaces como rutas.

A nivel seguridad en Capa de Red, es útil conocer cuanto ancho de banda se está utilizando en cada interfaz de red, lo cual nos permitirá descubrir si se está utilizando nuestra conexión a internet con fines non-sanctos, o incluso para determinar problemas de configuración de red o aplicaciones. Las herramientas bwmon, bmon o ntop nos permiten obtener un panorama bastante interesante acerca de estas cuestiones, sino ntop la más completa, ya que ofrece gráficos por interfaz, por protocolo, por puerto, por ip/red, etc.

De la misma manera, no puedo dejar de repetir lo importante que es crear correctamente nuestras reglas de firewall y NAT con Netfilter. No es propósito de este artículo hablar de Iptables/Netfilter, pero les recomiendo que lean la página completa del manual para iptables, lo cual les dará una idea de que cosas pueden hacer, lo cual les servirá más que una receta de cocina. Como regla básica, recomiendo que ustedes hagan sus propios scripts de firewall, que separen el tráfico por interfaz y por entrada/salida/forward.

Especialmente protejan la cadena INPUT, ya que algunas distribuciones de GNU/Linux habilitan varios servicios por defecto que no queremos que puedan verse desde Internet, o incluso desde otros ordenadores de nuestra red interna.

Asimismo, verifiquen que no existan rutas hacia redes que no deban estar, verifiquen el contenido de /etc/hosts y /etc/resolv.conf periodicamente, ya que algunos ataques de muy mala calidad se logran modificando el contenido de dichos archivos.

Capa de Transporte

De esta capa, lo principal es conocer que servicios se están ofreciendo, y desde dónde se los puede acceder. A tal efecto, nada mejor que las herramientas netstat, lsof y nmap, y un par de amigos que utilicen GNU/Linux o sistemas que puedan ejecutar nmap en otro proveedor de internet.

Por ejemplo, veamos el uso de netstat y la utilidad equery de Gentoo para determinar a que paquete corresponde un binario que tiene un puerto abierto en nuestro sistema:

Utilizaremos netstat de la siguiente manera: **netstat --inet -nlp**

De esta forma, obtendremos un listado como el siguiente:

```
noname ~ # netstat --inet -nlp
Active Internet connections (only servers)
Proto Local Address      Foreign Address  State    PID/Program name
tcp    127.0.0.1:64109    0.0.0.0:*        LISTEN   7520/wish
```

He removido algunas de las columnas, a efectos de mejorar la lectura de la salida. En este caso vemos que el puerto 64109 esta abierto por el programa "wish" cuyo pid es 7520, y esperando conecciones (State en LISTEN). Lo interesante de utilizar netstat en vez de nmap es que no se pierde nada de tiempo, el resultado es inmediato. Claro, lo utilizamos en nuestro propio ordenador, para investigar nuestro propio ordenador.

Ahora, con el comando ps, podremos ver la línea de comandos completa del programa "wish", buscando el PID.

```
noname ~ # ps ax|grep 7520
 7520 ?          S          0:29 wish /usr/bin/amsn
```

Mmm. La línea de comandos no nos dice mucho. Veamos a que paquete pertenece "wish". (Instrucciones para Gentoo Linux):

```
noname ~ # which wish
/usr/bin/wish
noname ~ # equery belongs /usr/bin/wish
[ Searching for file(s) /usr/bin/wish in *... ]
dev-lang/tk-8.4.9 (/usr/bin/wish -> wish8.4)
```

Bien, de aquí vemos que wish pertenece al paquete tk, el cual es el interprete para aplicaciones gráficas desarrolladas con el lenguaje TCL. Por lo tanto, quien abre el puerto debe ser la aplicación que es interpretada, en este caso /usr/bin/amsn. Utilizando el mismo procedimiento, descubriremos que el paquete en Gentoo se llama amsn, y con "emerge -s ^amsn\$" veremos que es un cliente de la red MSN Messenger. Y, efectivamente, tengo mi cliente de MSN Messenger abierto y conectado a la red. Pero... atención! Ya que SOLO está escuchando en 127.0.0.1, que corresponde a la interfaz loopback, la cual solo es accesible desde el propio ordenador, y no por la red. Si dijera 0.0.0.0 se vería desde cualquier lado el puerto abierto, siempre y cuando no haya reglas de firewall bloqueandolo, o si dijera alguna otra IP, sólo desde la interfaz vinculada a dicha IP. Para confirmar, podríamos ejecutar nmap desde algun otro ordenador, contra nuestra o

nuestras direcciones ip.

Suele ser útil configurar los servicios y el firewall para que mantengan una coherencia. Cuestión de que si alguien logra saltarse el firewall, o deshabilitarlo, la aplicación siga escuchando en las interfaces/IP correspondientes.

Capa de Aplicación

Esta es la capa más compleja, porque llegamos al nivel de la seguridad de cada aplicación, con lo que hablamos de conocer su funcionamiento interno, parámetros de configuración, realizar un seguimiento en las listas de correo, mantener contacto con usuarios de la misma, e investigar formas de optimizar su funcionamiento o seguridad. Claro, en programas simples, esta tarea es relativamente sencilla, pero en programas como Apache o Samba, debemos prestar especial atención.

Recomiendo utilizar Nessus periódicamente, actualizándolo con el comando `nessus-update-plugins`, y registrándonos como usuarios del mismo en www.nessus.org. Nessus nos ayudará a realizar una simple auditoría, lo cual nos proveerá de punteros y consejos para mejorar la seguridad de nuestra estación de trabajo, que generalmente los linuxeros también utilizamos como mini-servidor, sea de archivos, web o correo electrónico.

Palabras Finales

De más está decir que la seguridad es una cuestión importantísima, compleja y terriblemente interesante. En GNU/Linux encontraremos un aliado para securizar redes, realizar auditorías y aprender, al mismo tiempo pudiendo disfrutar de un sistema interesante, estable y... divertido (para el geek, claro).

Y sobre nuestro interrogante principal... ¿Es GNU/Linux seguro?, yo respondo que SI. Porque... "seguro podremos securizarlo", porque tiene tanta documentación, tantos expertos que lo aman y tanto código para leer, que todo eso sólo puede ser beneficioso para nosotros, los usuarios, administradores y... geeks.