Universidad de Salamanca | Grado en Ingeniería Informática

Política de Seguridad para un Servidor GNU/Linux de empresa

López Sánchez, Javier
Mateos Pedraza, Alejandro

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Abstract: The main goal of this document is to provide the system administrator with a complete and coherent guide through the different intervention methods that must be applied in order to protect the system from unwanted intrusions, as well as the protocols that must be adopted in the case of finding security holes that compromise the development of its activities.
Keywords: sysadmin, GNU/Linux, Debian, server, security policy

Índice de Contenidos:

Estado del Arte **3**

Análisis de riesgos **3**

Limitaciones y requisitos de seguridad críticos demandados por el sistema informático 3

Tipos de ataques y métodos conocidos para vulnerar la seguridad del sistema 5

Medidas de protección ante los ataques 9

Definición de la política **13**

Primeras nociones: ¿qué aspectos de seguridad debe cubrir nuestra política? 13

Seguridad de los componentes físicos 14

Sistemas antirrobo y de prevención ante manipulación de software 14

Seguridad de los periféricos 15

Seguridad de la BIOS 15

Seguridad del Bootloader 15

Herramientas de seguridad física adicionales: xlock y vlock 18

Detección de vulnerabilidades de seguridad a nivel físico 18

Seguridad local 19

Sobre las cuentas de usuario 19

Sobre la cuenta del root 20

Integridad de la información: Sistema de Ficheros 21

Ficheros especiales: capacitación de acceso a hosts 23

Gestión de permisos especiales a ficheros y directorios: sticky-bit y setuid 24

Herramientas avanzadas de monitorización del Sistema de Ficheros 24

Política de contraseñas y encriptación 25

Seguridad en red 26

Seguridad en el lado del cliente 26

Servicios de protección en el lado del cliente: ssh y stelnet 26

Seguridad en el lado del servidor: inetd 26

Herramientas avanzadas de seguridad en red 27

Sobre los servicios de correo electrónico 27

Seguridad de las aplicaciones 28

Un nivel de seguridad adicional: SELinux y AppArmor 28

Detección y protección frente a intrusiones **29**

Detectando intrusos 29

Medidas de protección 30

Herramientas de análisis y seguridad adicionales **32**

Principios de buenas prácticas **33**

Monitorización de los registros 33

Realización periódica de copias de seguridad 34

'Diccionario' de buenas prácticas: 35

El sistema ha sido comprometido: política de intervención **36**

Bibliografía y referencias **38**

1. Estado del Arte

Cada vez es más difícil controlar la seguridad de los sistemas Linux, a consecuencia de las conexiones a Internet de fácil acceso, el vertiginoso desarrollo del software, y en general, las comunicaciones e intercambio de datos globales que se realizan en nuestros días. Pongamos, por ejemplo, el paso de información de un punto A a un punto B. En dicho proceso, los datos circularán por diferentes puntos en el camino a su destino, de tal manera que en un sistema inseguro cualquier usuario se vería en posición de interceptarlos e incluso, si le place, alterarlos. Más aún, usuarios de nuestro propio sistema podrían también corromper nuestros propios datos. Los intrusos - mejor conocidos en la literatura de la administración de sistemas como "crackers"- podrían conseguir acceso a nuestro sistema sin autorización y emplear conocimientos avanzados para robar información o realizar prácticas de suplantación, pudiendo hasta impedirnos el acceso a nuestros propios recursos. Y esto es sólo la punta del iceberg de un mundo cada día más complejo, el de la seguridad en la administración sistemas. Teniendo en cuenta que ningún sistema es o será nunca inexpugnable, nuestro objetivo como administradores será el de incrementar la dificultad de que se vea comprometido, especialmente en aquellas compañías dedicadas a las telecomunicaciones, por no mencionar a la banca e incluso a las corporaciones que manejan datos gubernamentales.

2. Análisis de riesgos

2.1. Limitaciones y requisitos de seguridad críticos demandados por el sistema informático

Se ha tratado ya en la introducción la importancia y dificultad de incrementar la seguridad de nuestro sistema informático, pero un punto también a considerar es que el balance entre seguridad y usabilidad se ajuste a cada situación. Lo primero que debemos tener presente es el nivel de amenaza ante el que nos debemos de proteger, los riesgos que hemos de afrontar y cómo de vulnerable será nuestro sistema, así como tener pleno conocimiento sobre qué estamos protegiendo, el por qué, el valor de su contenido... No es lo mismo ser un usuario corriente que el administrador de una gran empresa que maneja gran cantidad de datos, y en muchos casos el destino de la empresa depende de la integridad y confidencialidad de los mismos. Merecen ser definidos tres conceptos previos a tener en cuenta en este sentido:

a. Riesgo: se refiere al grado de exposición que tiene el sistema para que un intruso acceda con éxito a una de sus máquinas y los problemas que de ello se derivan: lectura o escritura indiscriminada de ficheros, ejecución de programas maliciosos, borrado de datos importantes, suplantación de identidad, etc. Una cuenta insegura en el sistema puede acabar comprometiendo la red al completo. Otorgar a un usuario acceso a un archivo .rhosts o permiso para usar un servicio poco seguro como tftp han demostrado ser causas suficientes para lo propio. Una vez que el atacante posee una cuenta del sistema, puede emplearla para tener acceso a otro sistema o cuenta, propagando la infección.
b. Amenaza: procede de alguien que busca ganar acceso a nuestra red u ordenador sin autorización, por lo que está en la hoja de ruta del sysadmin decidir en quién confiar a la hora de dar acceso a nuevos usuarios. Existen muchos tipos de intruso: los “curiosos”, que simplemente pretenden encontrar qué tipo de sistema y datos tenemos; los “maliciosos”, cuyo objetivo final es derribar el sistema; los “competidores”, que buscan beneficiarse de nuestra información; los “intrusos de alto perfil”, que quieren usar nuestro sistema para conseguir fama haciendo gala y

promoción de sus habilidades; los "prestatarios", que quieren establecer un negocio utilizando los recursos de nuestro sistema para sus propios fines; o los "*leapfrogged*" que buscan simplemente acceder a otros sistemas utilizando el nuestro como punto intermedio.

c. Vulnerabilidad: es un parámetro que describe cómo de bien protegido está nuestro sistema, así como el potencial para que alguien pueda obtener acceso a él sin autorización.

Debemos considerar siempre que es mejor invertir un poco de tiempo al principio cuidando estos detalles, así como los referentes a las copias de seguridad, antes que tener que recrear todos los datos perdidos.

Por estos motivos debemos generar una política de seguridad sencilla para nuestro sistema que los usuarios puedan seguir. Esta política debe contemplar los datos que queremos proteger, así como la privacidad de los usuarios. Una política de este tipo suele empezar con la frase: "Lo que no está permitido está prohibido". Esto implica que un usuario no debería utilizar un servicio a no ser que le garanticemos acceso al mismo. Un detalle a tener en cuenta es que dicha política debe funcionar sobre la cuenta regular del usuario y no debe permitir ejecutar cosas como root si se desconocen los permisos necesarios -esto puede derivar en serios agujeros de seguridad-. Es importante también no descuidar nuestro sistema simplemente bajo la excusa de que "no se trata de una gran empresa" ya que también podemos ser potenciales víctimas de ataques por aquellos individuos cuyo objetivo es atacar muchos sitios diferentes, independientemente de su tamaño, o acceder a otros sistemas desde el nuestro.

En el ámbito de la seguridad, distinguimos diferentes ámbitos o campos donde actuar:

a. Seguridad de host o seguridad local: implica que nuestro sistema sea seguro a nivel de máquina, confiando en que los demás subsistemas de la red lo son. El administrador de seguridad local es responsable de muchos aspectos, como la definición de una política de contraseñas de calidad, o asegurar los servicios de la red local de nuestro host. Esto es algo esencial pero puede ser desalentador si la red cuenta con muchas máquinas.
b. Seguridad de red: tan necesaria como la anterior. En una red con muchos ordenadores conectados no puedes fiarte de que todos sean seguros. Asegurar que solo usuarios autorizados tienen acceso a la red, construir firewalls o una fuerte encriptación son parte fundamental de las obligaciones del administrador de la red.

Figura 1. Ilustración utilizada para representar la encriptación de las comunicaciones.

c. Seguridad 'en la sombra': este tipo de seguridad no es enteramente "seguridad" y se recomienda huir de ella en la medida de lo posible. Básicamente, se centra en esperar que el atacante no se percate de las vulnerabilidades de seguridad de determinados servicios. La experiencia demuestra que los agujeros del sistema acaban saliendo a la luz y, por ende, explotados por los delincuentes.

Asimismo, es de vital importancia tener en cuenta aquellos aspectos de seguridad que son necesarios en nuestro sistema, siendo los siguientes los requisitos de seguridad más relevantes:

a. Autorización: permisión de acceso a los datos únicamente a aquellos que lo necesitan.
b. Autenticación: verificación de que todo usuario del sistema es quien dicen ser.
c. Privacidad o confidencialidad: consiste en garantizar que la información personal no pueda verse comprometida.
d. Integridad: asegurar que los datos no han sido dañados o manipulados.
e. No repudio: confirmar que los datos son recibidos con capacidad para probarlo ante un tribunal.
f. Disponibilidad: garantizar que el sistema puede llevar a cabo la función requerida.

2.2. Tipos de ataques y métodos conocidos para vulnerar la seguridad del sistema

Existen una gran cantidad de métodos empleados en los ataques a la integridad del sistema, con dependencia tanto de los elementos de hardware como del software -que debe mantenerse actualizado a fin de aplicar las últimas correcciones de seguridad que vayan surgiendo-. Algunas de las principales técnicas de ataque empleadas en la actualidad son:

a. *Bug exploits*: los atacantes aprovechan agujeros o errores de un hardware, software, protocolo, servicio o incluso del propio sistema operativo para colarse en el sistema. Cada cierto tiempo se van descubriendo agujeros o "*bugs*" en los diferentes elementos informáticos, que nunca son perfectos. Dichos agujeros, pueden ser empleados por los atacantes con el fin de acabar con la seguridad de los sistemas. Las técnicas empleadas para este tipo de ataque pueden ser genéricas o específicas, en función del elemento a infectar. En lo que respecta a los elementos particulares, las personas responsables de los mismos son las encargadas de llevar a cabo nuevas versiones o parches para solucionar los posibles problemas. De cualquier modo, nosotros como administradores debemos mantenernos bien informados y establecer una política de actualización responsable con el fin de evitar riesgos de este tipo. También puede ser que no haya ninguna solución disponible por el momento, en cuyo caso, o bien buscamos alternativas a este componente, o bien lo deshabilitamos hasta que se solucione el problema.

b. Virus: son generalmente programas adjuntos a otros que emplean mecanismos de autocopia y transmisión. Podrían ser considerados la mayor plaga de seguridad que existe actualmente en muchos sistemas. Son comunes los virus anexos a programas ejecutables, mensajes de correo electrónico o aquellos que aparecen incorporados en programas o documentos que permiten algún lenguaje de macros no verificado. Los sistemas GNU/Linux están muy protegidos contra los virus porque:
   - Los programas ejecutables tienen un acceso muy limitado al sistema, y concretamente a la cuenta del usuario administrador -con excepción del root, que a consecuencia de su poder debe actuar con mayor responsabilidad-.
   - Ni el correo ni los documentos suelen emplear lenguajes de macros no verificados (aunque algunas de las suites de ofimática de GNU/Linux comienzan a usar lenguajes de macros y los clientes de correos van incorporando soporte de *html* empotrado con soporte de *JavaScript*, lo que es una fuente de bastante problemática).

Debemos tener en cuenta también que conforme aumenta el uso de la plataforma, más atractiva se va volviendo para los posibles atacantes y en un futuro podrían aparecer virus específicos para GNU/Linux.

Un punto de vital importancia a tener en cuenta es que aunque nosotros no generemos un virus, los podemos transmitir, por ejemplo si disponemos de un sistema que actúa como *router* (*relay*) de correo. En este caso podemos implementar alguna política de detección y filtrado de virus si en nuestro sistema existen entornos de *Windows* en los que es posible propagar virus que hayamos recibido del exterior. Un ejemplo a destacar en la categoría de virus es:

- *Spam*: el *spam* o correo 'basura' no suele emplearse como vector de ataque, aunque puede combinarse con técnicas como el *phising*, pero podríamos considerarlo como un problema más bien por su virulencia de aparición así como por el coste económico que pueden provocar al causar pérdidas de tiempo y recursos.

  Destacando algún virus informático importante tenemos el "*Melissa*" del año 1999, virus que venía introducido en un archivo llamado "List.doc" que contenía una enorme cantidad de contraseñas y registros para acceder gratis a sitios de contenido inapropiado. Este virus aprovechó el correo electrónico de modo que nada más abrir el documento, el virus accedía a los contactos y reenviaba el correo a 50 personas, infectando además todos los archivos de Word de la víctima, siendo un virus destructivo y letal. Otro virus muy relevante es el virus "*I love you*", que eliminaba todos los archivos jpg del ordenador, afectó a 50 millones de usuarios y produjo pérdidas de unos 5500 millones de dólares como consecuencia.

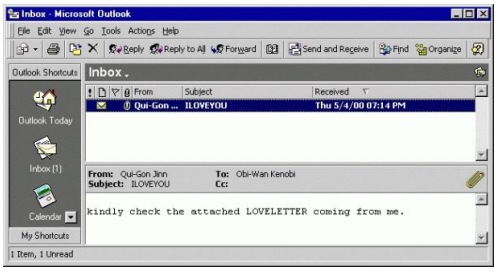


Figura 2. Virus "I Love You", virus letal | Fuente: BlogThinkBig

c. Gusanos: son programas que aprovechan agujeros del sistema para ejecutar código sin permiso, haciendo uso de los recursos de la máquina -como la CPU- cuando se detecta que el sistema no está en uso o en funcionamiento. También pueden ser malintencionados y robar recursos o bloquear el sistema. Emplean técnicas de transmisión y replicación. Un gusano de gran impacto en la historia fue el gusano "*Morris*", surgido en el año 1988 y que infectó al 10% de los usuarios de la época, causando daños de hasta 96 millones de dólares.

d. Troyanos: Programas que incorporan funcionalidad de utilidad pero que ocultan otras empleadas para comprometer el sistema u obtener información del mismo. Como ejemplo podemos hablar de los códigos de tipo móvil de aplicaciones web (*Java*, *JavaScript*, *ActiveX*...) que piden el consentimiento para ejecutarse o cuentan con modelos limitados de lo que pueden hacer. Se propagan fácilmente a través de los agujeros.

e. *Back doors*: las puertas traseras son una forma de acceso a un programa oculto que puede otorgar el acceso al sistema o a nuestros datos sin que lo sepamos. Pueden cambiar la configuración del sistema o permitir la introducción de algún virus. Puede venir tanto en un software común

como a través de un troyano que produzca puertas traseras.

f. Bombas lógicas: Programas incluidos en otros que comprueban determinadas condiciones como acciones del usuario o condiciones temporales con el fin de activarse y llevar a cabo acciones no autorizadas.

g. *Keyloggers*: Programa especial que captura las interacciones del usuario con el teclado o ratón con el fin de obtener la interacción del mismo con aplicaciones, sus sitios visitados en la red y sobre todo contraseñas o credenciales introducidos por el usuario. Pueden ser tanto programas individuales como troyanos incorporados en otros programas y lo normal es que se introdujesen en un sistema abierto al que se tuviese acceso.

Figura 3. Ilustración - Keyloggers | Fuente: Avast

h. Escaneo de puertos: Esto podría ser considerado más que un ataque, un paso previo de recolección de posibles objetivos. Se basa en el empleo de herramientas que ofrecen la posibilidad de examinar la red buscando máquinas con puertos abiertos (por ejemplo los puertos TCP o UDP). Un ejemplo de herramienta para este propósito es *nmap,* cuyo funcionamiento se puede explicar con el siguiente fragmento de código:

- Ejecutamos la orden: `Nmap 45.43.49.119`
- Obtenemos la siguiente salida:

```
Starting Nmap 7.70 (
    https://nmap.org ) at
    2019-10-01 18:12
CEST
Nmap scan report for ack.nmap.org
    (45.33.49.119)
Host is up (0.19s latency).
Not shown: 993 filtered ports
PORT STATE SERVICE
22/tcp open ssh
25/tcp open smtp
70/tcp closed gopher
80/tcp open http
113/tcp closed ident
443/tcp open https
31337/tcp closed Elite
Nmap done: 1 IP address (1 host up)
    scanned in 10.86 seconds
```

- Si analizamos dicha salida podemos observar que ha encontrado algunos puertos abiertos como ssh, smtp o http.

i. *Sniffers*: los *sniffers* se emplean para la captura de paquetes que circulan por una red. Si disponemos de herramientas apropiadas, seremos capaces de analizar comportamientos de máquinas, identificando servidores, clientes y protocolos utilizados así como en muchas ocasiones contraseñas de servicios no seguros. Algo importante de estas herramientas es que no solo tienen por qué emplearse para el ataque, sino también para analizar nuestro tráfico, evaluar nuestras redes o identificar fallos. Los atacantes suelen emplear técnicas de escaneo de este tipo para conocer datos del sistema desconocido o analizar la interacción interna y, en definitiva, encontrar las vulnerabilidades de un sistema.

j. *Hijacking*: se basa en intentos de configurar una máquina de forma que intercepte o sea capaz de reproducir el funcionamiento de un

servicio de otra máquina a la que ha pinchado la comunicación. Su empleo es más habitual en correo electrónico, transferencia de ficheros o sitios web (donde se puede capturar una sesión de usuario y reproducir lo que está haciendo). Puede darse esta situación de 'secuestro' tanto a nivel de red -captura o escucha de paquetes en comunicaciones- como de forma offline -ejecución arbitraria de códigos de *script*-. Podemos tomar el ejemplo de sistemas de correo electrónico por webmail o aquellas aplicaciones web que incluyen html en alguna de sus ventanas, en las que es posible insertar código de lenguajes como *JavaScript* o *PHP*. Si esto no está bien controlado se pueden robar datos del usuario -identificadores de sesión, *cookies*- lo que permite que el atacante pueda secuestrar la identidad y sesión del usuario a través de la reproducción a posteriori de sus sesiones de correo o de la aplicación web sin tener que conocer su contraseña.

k. *Buffer overflows*: la técnica de desbordamientos aprovecha errores de programación de las aplicaciones. Básicamente, como su nombre indica, aprovecha los desbordamientos de los *buffers* de la aplicación. El hecho de no controlar los límites del *buffer*, puede ser utilizado por un programa atacante para generar un dato o mensaje más grande de lo que se esperaba, produciendo fallos al escribir más allá de los límites permitidos, y sobrescribiendo de este modo la memoria adyacente. Un ejemplo de esta situación se da en los programas de C que contienen *buffers* mal escritos: si tenemos un *array* y sobrepasamos el límite que se le ha establecido, podemos sobrescribir el código del programa, provocando un funcionamiento incorrecto o incluso la caída del servicio o de la máquina. También se puede aprovechar para introducir en el propio ataque trozos de programas compilados en C o shell scripts de modo que se permita la ejecución de cualquier código que el atacante desee. Variaciones de esta técnica se pueden llevar a cabo para atacar por ejemplo la pila del programa o la gestión de peticiones dinámicas de memoria realizadas por el ejecutable. Cualquiera de estas alteraciones puede dar al atacante la posibilidad de ejecutar código arbitrario añadido, o directamente producir la caída de dicho ejecutable (o del servicio).

Figura 4. Ilustración - Desbordamiento de Arrays

l. *Denial of Service (DoS)*: la denegación de servicio es un ataque que ocasiona la caída de la máquina o la sobrecarga de uno o más servicios, inutilizándolos. Existe también la *Distributed Dos (DDos)* cuyo fin es usar un conjunto de máquinas distribuidas para que se produzca el ataque o la sobrecarga del servicio. Ambos ataques se suelen emplear contra sitios web o servidores DNS y se solucionan con actualizaciones de software y una configuración adecuada de los parámetros de control de carga del servicio. Un ejemplo a destacar de este tipo de ataque es *SYN FLOOD,* que genera paquetes TCP que abren una conexión, sin usarla posteriormente -solo la abren- consumiendo así recursos del sistema en estructuras de datos del núcleo y recursos de conexión por red. La repetición de este ataque un gran número de veces puede resultar a la larga en la ocupación de todos los recursos sin utilizarlos. Otro ejemplo que podemos destacar es el *mail booming*, técnica que consiste en bombardear con correos, normalmente de emisor falso, hasta saturar las cuentas de correo, esto produce que caiga el sistema de correo o se vuelva inutilizable por su lentitud. En estos casos

los ataques son "sencillos" de realizar y difíciles de solucionar al aprovecharse del funcionamiento interno de protocolos y servicios.

m. Spoofing: el enmascaramiento de identidad se basa en la falsificación de la información y de los participantes de una transmisión. Uno de los métodos empleados es el *IP spoofing*, que consiste en la falsificación de una máquina para generar tráfico falso o escuchar aquel dirigido a otra máquina. Otro ejemplo es el *ARP spoofing*, que emplea *DDoS* e intenta falsificar direcciones de fuentes y destinatarios en una red a través de ataques de las cachés de ARP de las máquinas, sustituyendo las direcciones reales por otras en múltiples puntos de la red. Por último, el método de correo electrónico basado en generar correos falsos (tanto en contenido como la dirección de origen). Esta última técnica es la empleada en la llamada "ingeniería social" cuyo fin es engañar al usuario de manera 'educada'. Un ejemplo de este engaño es conocido como *phising* y se basa por ejemplo en solicitar al usuario (administrador, cliente del banco,...) que introduzca información personal en una página falsa haciéndole creer que es la verdadera.

2.3. Medidas de protección ante los ataques

Antes de pasar a tratar específicamente las medidas frente a cada ataque específico, podemos mencionar algunas recomendaciones muy generales para la seguridad del sistema frente a ellos. Son medidas que pueden ser poco productivas pero si no aseguramos el sistema, no tenemos garantizado ningún control sobre lo que puede pasar, de hecho, nadie nos puede asegurar que un programa malicioso no burle nuestra seguridad si lo ejecutamos con los permisos apropiados. Estas medidas son:

a. Control del comportamiento de los usuarios, concretamente, por la confidencialidad de las contraseñas. Este es uno de los puntos que más puede facilitar a los atacantes actuar desde el interior del sistema, de hecho es de donde vienen la mayor parte de ataques. Con respecto a estos usuarios podemos señalar aquellos que escriben contraseñas predecibles, o que directamente las olvidan cada pocos días, o las comparten con otras personas. También debemos tener en cuenta que aunque sea una contraseña apropiada, existen mecanismos para encontrarlas. Aparece aquí el concepto de "cultura de seguridad entre usuarios", que incluye obligarles a cambiar la contraseña, que no sean o contengan palabras típicas o que cumplan con una determinada longitud por ejemplo. Muchas empresas están también empezando a llevar a cabo firmas de contrato donde se le obliga al empleado a cumplir determinadas medidas de seguridad para sus contraseñas.

b. No utilizar o ejecutar programas de origen desconocido (generalmente los distribuidores emplean mecanismos de comprobación de firmas para garantizar que los paquetes de software son los especificados). Tampoco utilizar programas o servicios cuyo funcionamiento desconocemos ni probar a adivinarlo a partir de ejecutarlo de forma repetida.

c. No utilizar las cuentas de usuarios privilegiados (root) para trabajar de forma normal con la máquina, dado que cualquier programa tendría permiso para entrar a cualquier zona del sistema de archivos o cualquiera de los servicios que se encuentra en ejecución. Del mismo modo tampoco se debe acceder de forma remota directamente como usuario privilegiado, y más aún, ejecutar programas con permisos de administración si no sabemos los niveles de

seguridad que posee el sistema o sus conexiones (no utilizar conexiones críticas en redes inseguras).

A continuación, conviene tratar las diferentes medidas de prevención y detección de intrusos que se pueden llevar a cabo frente a cada tipo de ataque indicado anteriormente:

a. *Password cracking*: dado que este ataque -cuyo objetivo es el de romper contraseñas y acceder al sistema comprometiendo la seguridad del usuario- se suele llevar a cabo por *logins* realizados de forma repetida, una posible solución sería reforzar la política de contraseñas como se mencionó previamente (longitud mínima, cambios periódicos, cierta complejidad, ...). En concreto el punto más importante a tener en cuenta es que no sean muy obvias dado que estos ataque se suelen llevar a cabo a través de un diccionario que posee términos en el idioma del usuario, nombres propios o incluso se pueden realizar estas pruebas con datos propios del usuario como su DNI o su dirección. Una manera apropiada de establecer estas contraseñas es que cuenten como mínimo con 6, 7 u 8 caracteres y que dichos caracteres incluyan signos alfabéticos, numéricos y algún que otro carácter especial. No obstante por muy segura que sea la contraseña, puede ser vulnerable si es empleada en sitios inseguros, por lo que es recomendable llevar a cabo refuerzos mediante técnicas y servicios de cifrado para proteger tanto las contraseñas como las comunicaciones. Algunos servicios que no cuentan con dicho cifrado y que por lo tanto podrían ser considerados como no seguros son telnet, rlogin, ftp,...

b. Explotación de agujeros (*'bug exploits'*): se deben evitar aquellos programas obsoletos debido por medio de parches y actualizaciones de seguridad a fin de garantizar la seguridad del sistema operativo y las diferentes aplicaciones que pueda contener. Un consejo es emplear herramientas avanzadas que detecten estas vulnerabilidades.

c. Virus: la solución se basa en el empleo de antivirus, sistemas de filtrado para mensajes sospechosos y evitar la ejecución de sistemas de macros, ya que no pueden ser verificados. Debemos tener en cuenta que cada día los virus se perfeccionan, y especialmente si tenemos entornos de *Windows* es importante examinar los virus que pueden afectar a estos sistemas. Algunos antivirus de Unix y Linux como *ClamAV* evitan que los virus se propaguen a nuestros sistemas internos. Una idea interesante es combinar productos antivirus como el mencionado con productos *antispam* como *SpamAssasin*.

d. Gusanos: la solución es controlar el uso de nuestras máquinas o usarlos en horas no previstas. En resumen con patrones de comportamiento variables, controlando el tráfico de entrada y salida.

e. Troyanos: debemos controlar la integridad de nuestros programas periódicamente haciendo uso de mecanismos de sumas de verificación (*checksum*) o firmas y a través de comprobaciones para detectar si existe tráfico anómalo de entrada o salida al sistema (podemos usar cortafuegos para bloquear dicho tráfico). Los *rootkits* son una versión peligrosa de los troyanos, que utilizan varias herramientas para realizar más de una función. Algunas herramientas que podemos utilizar para la verificación de la integridad mencionada son *md5* y *gpg* para mecanismos de sumas, o directamente *Tripwire* ó *AIDE,* que automatizan el proceso.

f. Puertas traseras: debemos obtener la certificación por parte de nuestros

proveedores de software de que no hay puertas traseras escondidas y no documentadas, aceptando únicamente el software de sitios que nos den determinadas garantías. En caso de software de terceros o de fuentes que podrían haber alterado el software original, muchos fabricantes integran los ya mencionados en el apartado anterior, códigos de suma o firmas digitales como forma de verificación del mismo. Deberíamos verificarlas siempre que estén disponibles como paso previo a la instalación del software. Una alternativa es probar el sistema de un modo intensivo antes de establecerlo como un sistema de producción. La modificación del software a posteriori también puede ser un problema, en cuyo caso los sistemas de firmas o sumas, para generar códigos sobre software que ya está instalado con el fin controlar que no haya cambios vitales, son muy recomendables. Otro punto a destacar son las copias de seguridad, gracias a las cuales, podemos detectar cambios por medio de comparaciones.

g. Bombas lógicas: se suelen ocultar tras acciones del usuario o activaciones por tiempo. Lo que podemos hacer es verificar que no haya trabajos no interactivos introducidos de tipo *cronbat*, *at* o procesos como los de tipo *nohup* en el sistema, que cuenten con una ejecución periódica o que directamente hayan estado ejecutándose en segundo plano un largo periodo de tiempo (para ello podemos usar los comandos *w* y *jobs*). Las acciones que podemos tomar son el empleo de medidas preventivas para impedir trabajos programados no interactivos a los usuarios - *cronbat* - o solo permitir aquellos que sean realmente necesarios.

h. *Rootkits* y registradores de tecleo: como se trata de un proceso intermediario que intenta capturar nuestras pulsaciones de tecla, para almacenarlas o comunicarlas, lo que tenemos que buscar son procesos extraños pertenecientes a nuestro usuario o ficheros abiertos con los que no estemos trabajando (a través del comando *lsof*) o conexiones de red si estamos ante un registrador de tecleo con envío externo. El *rootkit* por otro lado es un paquete que contiene varios programas con diferentes técnicas que permiten al atacante tras entrar en una cuenta, utilizar registradores de tecleo, puertas traseras, troyanos para obtener información así como puertas de entrada al sistema. Este tipo de ataques pueden ir acompañados hasta de programas que ayuden a eliminar las pruebas borrando los registros. Los rootkits más peligrosos son los que aparecen como módulos del núcleo ya que pueden actuar a nivel del mismo. Las herramientas *chkrootkit* y *rkhunter* pueden ser útiles a la hora de verificar los *rootkits*.

i. Escaneo de puertos: como ya hemos mencionado previamente esta forma de ataque se basa simplemente en detectar puertos abiertos y servicios en funcionamiento que podrían ser más susceptibles de ataque.

j. Sniffers: si evitamos intercepciones de nuestros paquetes podemos impedir que se introduzcan escuchas. Una posible técnica es construir el hardware de red dividido por segmentos, permitiendo que el tráfico no circule más allá de la zona donde se va a utilizar. Si añadimos cortafuegos para unir los segmentos podremos controlar más fácilmente el tráfico de entrada y el de salida. También podemos manejar técnicas de cifrado para evitar que los mensajes sean leídos por alguien que pueda estar escuchando la red. Algunas herramientas útiles son *Wireshark* y *Snort* si lo que queremos es comprobar nuestra propia red o *nmap* si vamos a escanear los puertos. Concretamente, los husmeadores pueden ser detectados en la red a través de la búsqueda

de máquinas que se encuentren en el modo *Ethernet* promiscuo, es decir, aquellas que están escuchando cualquier paquete que circula. De forma general, la tarjeta de red solo captura aquel tráfico dirigido a ella, ya sea broadcast o multicast.

k. Hijacking: implementar mecanismos de cifrado para los diferentes servicios, requerir autenticación (y que esta sea renovada periódicamente), emplear cortafuegos para controlar tráfico entrante y saliente o monitorizar la red para identificar flujos de tráfico que sean sospechosos son algunas de las posibles contramedidas que podemos tomar frente a los secuestros. Para aplicaciones web o correo electrónico una buena medida sería limitar todo lo posible el uso de códigos de *script* o directamente eliminar la posibilidad de que se puedan ejecutar *scripts* internos a ventanas de html en caso de que las fuentes de donde provengan, no sean de fiar.

l. Desbordamientos: normalmente aparecen como *bugs* del sistema y se pueden solucionar con una actualización del software o el paquete en cuestión. A través de los registros del sistema se pueden observar situaciones donde un servicio que debería estar funcionando, cae. Maximizar los controles de procesos y accesos a recursos es una buena medida para aislar el problema si se produce en entornos de acceso controlado como *SELinux*. *Grsecurity* o *PaX*, son parches de los núcleos oficiales de Linux que otorga protecciones adicionales para los desbordamientos, ya sean de pila (*heap*) o de los *buffers*.

m. Denegación de servicio y otros como SYN Flood o bombardeo de correo: una de las medidas a tomar es el bloqueo de tráfico que sea innecesario por nuestra red, de nuevo haciendo uso de cortafuegos. Además debemos controlar como es lógico tamaños de *buffer*, número de clientes pendientes por atender, los *timeouts*,... en todos aquellos servicios donde se pueda.

n. Falseamiento de identidad: Sea cual sea el tipo de falseamiento de los especificados en los tipos de ataque, la solución es llevar a cabo un cifrado fuerte de los servicios, realizar un control de los cortafuegos, emplear mecanismos de autenticación basados en más de un aspecto, usar mecanismos que controlen qué sesiones se han establecido, basándose en varios parámetros de la máquina al mismo tiempo, monitorizar sistemas DNS, cachés de ARP entre otros, con el fin de detectar determinados cambios en la información que hagan que los anteriores no sean válidos.

o. Ingeniería social: Principalmente las medidas están orientadas a educar a los propios usuarios, que como ya hemos mencionado, deben ser responsables de sus credenciales y conocer las técnicas básicas que puedan aumentar su seguridad. También es muy importante controlar quién puede tener acceso a la información de seguridad y quién no.

Figura 5. Simulación de la técnica de Phising | Fuente: BBVA.com

3. Definición de la política

3.1. Primeras nociones: ¿qué aspectos de seguridad debe cubrir nuestra política?

Como si de una afección se tratase, el protocolo de seguridad de todo sistema informático debe disponer de medios para la prevención, detección y recuperación del propio sistema, sin fisuras críticas de manera innegociable, pues de ellos depende en gran parte la integridad, inmunidad, estabilidad y confidencialidad de la información; y, a largo plazo, el éxito de la empresa, indistintamente de su sector de ocupación. Debemos ser conscientes de que el único sistema seguro es aquél que no se abre al mundo; y con ello no sólo nos referimos a un supuesto sistema que, obviamente, debiera estar desconectado de la red, sino también apagado y encerrado bajo llave, pues incluso en sus entrañas el sistema puede ser inseguro, y no puede ser nunca protegido por completo, si bien podemos intervenir para hacerlo evolucionar, adaptarse y fortalecerse. Unas medidas de prevención, detección y recuperación adecuadas le ayudarán a salir indemne de los riesgos a los que se enfrente a raíz de su apertura al vasto universo empresarial. Y aún así, la aplicación de estas medidas no nos garantiza tener el sistema intacto, pues quizá el escudo haya sido levantado tarde: atacantes con acceso previo al sistema, o los mismos usuarios pueden intentar poner en jaque las restricciones impuestas.

La seguridad debe evaluarse desde tres planos, a grandes rasgos: a nivel local, de conexión en red y de los propios servicios.

En cuanto a las medidas de actuación a nivel local, deberán definirse con claridad los mecanismos de autenticación, así como los permisos concedidos a usuarios y grupos de usuarios para la gestión de la información y los recursos, de tal manera que los datos de la compañía no asuman riesgos de ninguna índole.

A nivel de red, garantizar la seguridad de los recursos debe verse desde dos lados del mismo prisma: la empresa como cliente y la empresa como servidor. Como es lógico, llegado el caso -sumamente común en nuestros días- de contratar servicios a terceros, debemos tener un control fidedigno, nuevamente, de los procesos de autentificación y acceso a los servidores oportunos, realizando los controles precisos para la detección de posibles técnicas de suplantación –‘*phishing’*, en la literatura *hacker*-, que por lo general resultan difíciles de detectar a nivel usuario. En cuanto a la organización como servidora de sus propios recursos en red -si lo fuere-, debemos tener la certeza de que aquello que se proporciona es confidencial para los intereses de la propia empresa y también de su cliente, en armonía con la jurisdicción vigente. Asimismo, conviene asegurar que cada cliente haga uso de los servicios que exclusivamente ha contratado y que, por descontado, nuestros recursos no son utilizados por terceros indeseados. A fin de cuentas, el administrador debe controlar que únicamente se proporcionan y se utilizan los servicios necesarios y conocidos, sin concesiones.

Respecto a las aplicaciones, debe vigilarse la explotación de ‘bugs’ del software. Las fisuras en el desarrollo de los diferentes programas resultan especialmente frecuentes y proporcionan una vía de acceso a los intrusos: técnicas como los troyanos o las puertas traseras –‘*backdoors*’- no reciben su nombre por casualidad y aprovechan estos fallos para realizar sus invasiones, generalmente de forma transparente al propio sistema. Frecuentemente, controlar y conocer la identidad de los servicios instalados en el sistema y apoyarse en herramientas de actualización suele ser lo más conveniente, aunque también debe existir un compromiso

entre actualización y estabilidad de las propias aplicaciones. No obstante, la mejor manera de evitar los ataques suele ser siempre la prevención y la aplicación de una serie de buenas prácticas, que en capítulos posteriores se detallarán. Visitar fuentes de prestigio en materia de fallos de software y mantenerse informado de las nuevas vulnerabilidades detectadas es un consejo más que recomendable en este sentido.

3.2. Seguridad de los componentes físicos

La primera capa de seguridad que todo administrador debe tomar en consideración es la seguridad de los propios componentes físicos comprados por su empresa: en especial, la integridad de las máquinas y del propio entorno de trabajo. En este sentido, deben darse respuesta a diferentes preguntas: ¿quién o quiénes tienen acceso físico directo y autorizado a cada máquina de cada departamento? ¿Debe considerarse un uso compartido de los equipos? ¿Quiénes deben compartirlos? ¿Cómo protegemos a cada dispositivo de ser manipulado por agentes externos e incluso por los propios empleados?

No obstante, la seguridad física que demanda el sistema empresarial depende evidentemente del contexto y el presupuesto. Las políticas de buenas prácticas vuelven a ser una herramienta fiable, al menos, para protegerse de puertas para adentro, y en este sentido, cada empresa puede fijar diferentes directrices a sus empleados en función de su situación: mientras que en algunos casos dejar el equipo desbloqueado al finalizar el trabajo puede considerarse aceptable, en otros puede llegar a ser motivo de despido. Los métodos de seguridad física y electrónica más obvios como colocar cerraduras en las puertas, proteger los cables y asegurar los armarios de cableado, e incluso la videovigilancia son buenos aliados para alcanzar la integridad física del sistema.

Sistemas antirrobo y de prevención ante manipulación de software

La mayor parte de los sistemas informáticos fijos para el sector empresarial disponen en la actualidad de diferentes mecanismos para bloquear físicamente los equipos y dispositivos asociados. Estos mecanismos de bloqueo resultan útiles con objeto de evitar que cualquier individuo, de la empresa o ajeno a ella, pueda directamente manipular o incluso robar el hardware. Además, resultan especialmente aconsejables para evitar que se puedan producir reinicios de sistemas locales desde discos de instalación no controlados o a través de cualquier otra herramienta de hardware con este propósito. El mecanismo utilizado para implementar dichas cerraduras depende en muchas ocasiones de la arquitectura de la placa madre: en muchos casos, se diseña de tal manera que el equipo debe romperse para manipularse físicamente, siendo esto lo más aconsejable. En algunos casos, tampoco se permite enchufar nuevos periféricos como teclados o ratones. Algunas máquinas -como las SPARC o las MAC-, la arquitectura es tal que será necesario romper la caja para poder abrirla, por lo que no existen métodos de cerrajería que se puedan aplicar para vulnerar el equipo a nivel de electrónica. Sin duda un buen disuasivo para aquellos que pretendan robar o alterar el material de oficina.

Seguridad de los periféricos

Si los dispositivos disponen de una cámara web o un micrófono conectado, debemos considerar si hay algún peligro de que un atacante tenga acceso a esos dispositivos. Cuando no se utilizan, resulta conveniente desconectarlos, inhibir su función por medios físicos o incluso eliminarlos, si es posible. De lo contrario, se debe tomar extrema precaución acerca de todo software al que se concede acceso a los periféricos.

Seguridad de la BIOS

La BIOS constituye el nivel más bajo de software que configura el hardware basado en arquitecturas x86. Los diferentes gestores de arranque en Linux -LILO ("Linux Loader"), en particular- acceden a la BIOS para determinar cómo arrancar el dispositivo Linux. Como tal, la BIOS puede ser utilizada con objeto de impedir que los atacantes reinicien nuestras máquinas y manipulen el sistema Linux de la empresa. Asimismo, muchas BIOS de PC permiten establecer una contraseña para acceder a la propia herramienta. Aunque esto no proporciona un nivel de seguridad óptimo, dado que la BIOS puede ser restablecida por el atacante llegado el caso, podría ser un buen elemento disuasorio: más allá de porque lleva tiempo, porque los ataques de diccionario típicos suelen dejar un rastro una vez que el atacante atraviesa las puertas del sistema, pues el sistema puede y debe llevar registro de los accesos, como se verá en apartados posteriores.

Otro riesgo derivado de confiar en las contraseñas de la BIOS para asegurar el sistema físico es el problema de las contraseñas por defecto. La mayoría de los fabricantes de BIOS incorporan contraseñas predeterminadas que funcionan independientemente de la contraseña elegida para acceder a la herramienta, a fin de contemplar el caso de olvido de la clave establecida.

Estas contraseñas son bastante fáciles de conseguir en los sitios web de los respectivos fabricantes, por lo cual no podemos considerar las contraseñas de la BIOS como una protección efectiva, al menos no ante atacantes con buena preparación. No obstante, muchas BIOS x86 permiten especificar configuraciones de seguridad de mayor complejidad. Conviene, por lo tanto, revisar detenidamente el manual de la BIOS para jugar con estas configuraciones. Por ejemplo, algunas BIOS impiden taxativamente arrancar el sistema desde unidades extraíbles -discos y USB de instalación, principalmente- y algunas demandan nuevas contraseñas para acceder a funciones críticas de la BIOS. Otro punto en contra de las contraseñas lo encontramos en el caso de los servidores, que no podrán arrancar de forma autónoma si no tienen a alguien con los privilegios necesarios para ingresar la clave correspondiente.

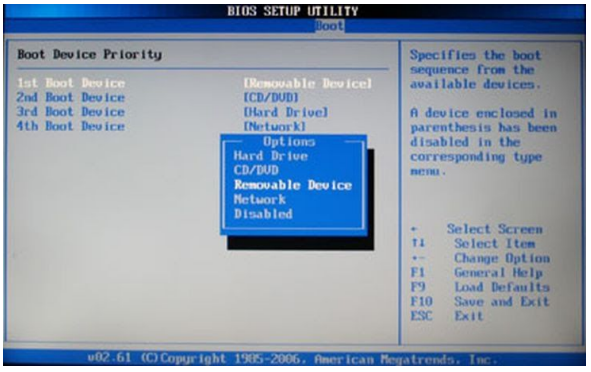


Figura 6. Configuración de los modos de arranque de la BIOS.

Seguridad del Bootloader

Los diversos gestores de arranque -bootloaders- de Linux también pueden tener una contraseña de arranque. LILO ("Linux Loader"), por ejemplo, cuenta con contraseña y determinadas configuraciones de arranque restringidas. La protección del bootloader es crítica si no se desea tener un agujero de seguridad de dimensiones astronómicas. De no hacerlo, cualquier usuario con conocimientos no muy avanzados podría

emprender las siguientes acciones contra el sistema:

a. Acceso al equipo con privilegios de superusuario (“root”), simplemente editando los parámetros del gestor de arranque. Si un atacante consigue acceder al sistema con tales privilegios, adquiere control total sobre el equipo, los usuarios y la información presente en él.
b. Posibilidad de abrir un intérprete de órdenes para recabar información del dispositivo, sin necesidad de contraseñas o requerimientos de seguridad.
c. Arranque del sistema desde sistemas operativos no controlados y por lo tanto inseguros.

En caso de tener GRUB (‘Grand Unified Boot Loader’) como gestor de arranque -como es nuestro caso en la máquina Debian 10 que hemos tomado para realizar algunas pruebas-, también se nos proporciona la posibilidad de fijar una contraseña, para que sólo los administradores puedan iniciar operaciones interactivas (es decir, editar las entradas del menú de GRUB e ingresar a la interfaz de línea de comandos).

Como ya se ha dicho, es importante que el administrador recuerde toda esta batería de contraseñas -como dice la sabiduría popular, es necesario predicar con ejemplo- y recordar que éstas sólo servirán para retrasar la ‘enfermedad’, pero que nunca son garante de seguridad total o una barrera de detención infalible. El fichero para la gestión del arranque (/etc/lilo.conf o en su defecto los ficheros de los directorios /boot/grub y /etc/grub.d) deberán estar en modo "600" -lectura y escritura sólo para el root-, si no queremos ver el sistema comprometido en exceso.

Se ha realizado una prueba de configuración de seguridad de una máquina virtual Debian 10 con GRUB como gestor de arranque. Los pasos a seguir incluyen tres aspectos a tener en cuenta: bloqueo de acceso a la línea de comandos del GRUB, bloqueo de edición de las entradas del GRUB y bloqueo de la posibilidad de ejecución de todas las entradas del GRUB a usuarios no autorizados. Los pasos que se han seguido para dejar el gestor de arranque con un nivel de protección aceptable -recuerde que las contraseñas deben considerarse simplemente una medida disuasoria- son los siguientes:

1. Realizar una copia de seguridad de los ficheros a modificar de los directorios /boot/grub y /etc/grub.d, y disponer de un medio de instalación del sistema operativo en caso de que cualquiera de los pasos posteriores se realice de forma incorrecta:

```
sudo cp
    /boot/grub/grub.cfg
    ~/grub.cfg.old
sudo cp
    /etc/grub.d/00_header
    ~/00_header.old
sudo cp
    /etc/grub.d/10_linux
    ~/10_linux.old
sudo cp
    /etc/grub.d/30_os-probe
    r ~/30_os-prober.old
```

2. Añadir al final del fichero 00_header la relación de usuarios que tendrán la posibilidad de usar la línea de comandos y editar las entradas del propio bootloader. Es conveniente que las contraseñas definidas no coincidan con las contraseñas de los propios usuarios en el sistema, por razones obvias. Se permite también definir nuevos usuarios no existentes en el propio sistema:

```
sudo pico /etc/grub.d/00_header

        cat << EOF

        set superusers="root,alema"

        password root adminsys2020

        password alema user2020_1

        EOF
```

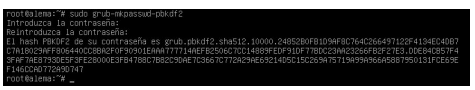
```
cat << EOF
set superusers="root,alema"
password root adminsys2020
password alema user2020_1
EOF
```

3. Cifrar las contraseñas de los usuarios indicados en el paso anterior: las nuevas contraseñas, por el momento, aparecerán indicadas de forma explícita en el */etc/grub.d/00_header*.

```
root@alema:~# tail /etc/grub.d/00_header

if [ "x${GRUB_BADRAM}" != "x" ] ; then
  echo "badram ${GRUB_BADRAM}"
fi

cat << EOF
set superusers="root,alema"
password root adminsys2020
password alema user2020_1
EOF
root@alema:~# _
```

Esto no es una característica deseable en absoluto, por lo que debemos generar un hash que permita ocultar las contraseñas que se acaban de configurar:

```
sudo grub-mkpasswd-pbkdf2
```



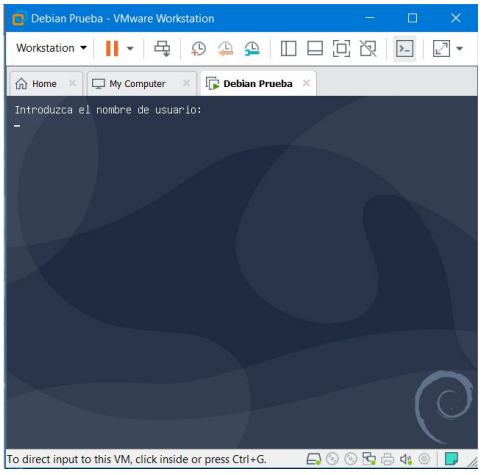
4. El sistema solicitará la contraseña a encriptar en dos ocasiones, por cada clave que se desee encriptar. Debemos repetir la operación para los dos usuarios indicados en el fichero de header.
5. Una vez obtenido el hash, regresamos al fichero correspondiente para modificar las contraseñas explícitas por los hashes generados:

```
cat << EOF

set superusers="root,alema"

password_pbkdf2 root [HASH root]

password_pbkdf2 alema [HASH
alema]

 EOF
```

6. Finalmente, para que los cambios tengan efecto se debe actualizar la configuración del GRUB mediante la orden:

```
sudo update-grub2
```

7. Dado que nuestra versión de GRUB es posterior a la 2.0 (*grub-install --version*) nuestro sistema está convenientemente protegido frente al acceso a la línea de órdenes del bootloader, la edición de sus entradas y la ejecución de las entradas del GRUB a usuarios no autorizados. Finalmente, comprobamos que los cambios han tenido lugar reiniciando el sistema.



Una vez configurada convenientemente la BIOS para impedir el acceso a la propia herramienta y los arranques por medios extraíbles (como se ha indicado en el capítulo anterior), y una vez asegurado el bootloader con los pasos anteriores, podemos considerar un nivel de protección física bastante aceptable. No obstante, podemos ir un paso más allá, cifrando las particiones del disco duro para evitar que desde un LiveCD se puedan montar las particiones de nuestro equipo, o incluso, llegado el extremo, limitar el acceso físico al propio ordenador por parte del usuario, a fin de asegurar físicamente cualquier tipo de acceso por medios extraíbles no autorizados.

Herramientas de seguridad física adicionales: xlock y vlock

Si el usuario debe ausentarse de su puesto de trabajo a menudo, puede ser una buena idea también bloquear la consola -en caso de que el usuario deba tenerla habilitada, pues en caso contrario lo común es incluso inhibir el propio programa- a fin de evitar posibles manipulaciones o accesos intrusivos. Dos pequeños programas que cumplen esta función son *xlock* y *vlock*. *xlock* es un sistema de bloqueo con interfaz gráfica (basado en el conocido sistema de ventanas X) y está disponible para cualquier distribución de Linux que soporte X. Ejecutar xlock desde cualquier terminal bloqueará la pantalla y requerirá la contraseña correspondiente para desbloquearla. Por su parte, vlock es un pequeño y simple programa que nos permitirá bloquear algunas o todas las consolas virtuales de nuestro Linux.

Por supuesto, bloquear la consola evitará que el trabajo en curso pueda ser editado, eliminado manipulado, pero no impide que la máquina pueda reiniciarse por medios ajenos, causando la pérdida del trabajo en curso. Nada impide tampoco acceder a la máquina desde otro dispositivo conectado a la misma red. Y más importante aún, no impide que un atacante pueda apagar completamente el sistema de ventanas X e iniciar un proceso de login desde la consola virtual a partir de la cual se inició el sistema de ventanas, suspenderlo y realizar así una escalada de privilegios. Por esta razón, se debe considerar autorizar los accesos de los usuarios bajo control de XDM (X Display Manager), convenientemente configurado.

Detección de vulnerabilidades de seguridad a nivel físico

La primera cuestión a tener en cuenta al respecto de las vulnerabilidades en la capa física debe ser al respecto de las ocasiones en que la máquina ha sido apagada o reiniciada. Linux es un entorno seguro y fiable, por lo que los únicos momentos de 'debilidad' que es posible explotar se producen cuando el equipo se reinicia por motivos de actualización del propio sistema operativo o por cambios de hardware. Si una máquina presenta reinicios no controlados, podemos tener signos de que el sistema ha sido comprometido, pues gran cantidad de los ataques a la integridad del sistema requerirán que el intruso deje su huella, reiniciando o apagando el dispositivo. Asimismo, deben realizarse inspecciones frecuentes para comprobar si existen signos de manipulación del material de oficina.

Aunque los intrusos -que por lo general saben lo que hacen- limpian los rastros de su presencia de los registros (eliminando sus huellas de los logs del sistema), es una buena idea comprobarlos todos ante cualquier signo de alarma y anotar cualquier discrepancia. En este sentido puede ser una buena idea almacenar los datos de los registros en un lugar seguro, como un servidor de registros específico dentro una red bien protegida. Una vez que una máquina ha sido comprometida, los datos logs en la mayoría de los casos pierden su utilidad, ya que probablemente también habrán sido modificados por el intruso en cuestión.

Una idea mucho más interesante reside en la configuración del demonio syslog. Podemos

ordenar a este demonio el envío automático de los datos de registro a un servidor central de syslog. Sin embargo, esta información no puede enviarse de forma pública sin encriptar, permitiendo a un intruso ver los datos a medida que se transfieren, pues sería como tapar la cabeza y descubrir los pies. Por suerte, existen demonios de syslog que son capaces de encriptar los datos a medida que se envían. Aun así, debemos tener en cuenta que los atacantes disponen de medios para falsificar los mensajes del syslog mediante programas denominados 'exploits', por lo que el acceso a la configuración de este daemon debe restringirse al máximo en lo posible.

Entre la información que debe comprobarse en los registros, proporcionamos las siguientes claves:

a. Completitud de los registros: un registro incompleto es señal inequívoca de modificación no controlada.
b. Registros que contengan extrañas marcas temporales.
c. Registros con permisos o propietarios incorrectos.
d. Datos registrados sobre reinicios de servicios o del sistema completo.
e. Registros perdidos.
f. Entradas de superusuario o logins de dudosa procedencia.

3.3. Seguridad local

El siguiente escalón que se debe superar es la seguridad del sistema a nivel local, protegiéndolo de los ataques -intencionados o por completo desconocimiento- que nuestros propios usuarios puedan realizar contra el sistema. Obtener acceso a una cuenta de usuario local es el primer paso para un atacante en su camino para explotar la cuenta del administrador. En un sistema con seguridad local poco estricta, un usuario podría en pocos pasos realizar una escalada de privilegios hasta convertirse en el root, con todas las consecuencias. En cambio, si nos aseguramos de que nuestro protocolo de seguridad a nivel local no da manga ancha a las acciones de un usuario común, entonces los intrusos tienen un nuevo obstáculo que saltar. Un usuario local puede causar -por difícil que parezca de concebir- estragos en el sistema, especialmente si dicho usuario goza de la confianza del propio administrador. Proporcionar cuentas a personas que se desconocen o en las que no se confía es una idea totalmente desaconsejable. La seguridad local es por tanto un factor crítico, pues la mayoría de los ataques terminan por hacer uso de recursos internos del sistema.

Sobre las cuentas de usuario

Debemos asegurarnos de que a cada usuario se le conceden los recursos mínimos para que pueda realizar la tarea que tiene asignada. La distribución de usuarios en grupos en función de los recursos que demandan es una buena práctica en esta dirección, y en Linux esta distribución resulta especialmente sencilla. Se citan a continuación un conjunto de reglas que debemos adoptar como dogma para permitir el acceso legítimo a una máquina Linux de nuestro sistema:

a. Conceder a los usuarios la cantidad mínima de recursos y privilegios de acceso que necesitan para desempeñar su tarea.
b. Llevar un registro de todos los inicios de sesión y de los intentos de acceso que tienen lugar en el sistema, y compararlos con el registro oficial de cuándo estos accesos deben producirse de forma lógica.
c. Eliminar completamente las cuentas inactivas. Para saber si existen cuentas inactivas que se han pasado por alto se puede consultar el último comando del usuario y/o comprobar los archivos de log a fin de conocer cuál fue el último rastro de actividad generado por el usuario en cuestión.

d. Facilitar el proceso de mantenimiento de las cuentas de usuario utilizando un formato de identificación de usuarios uniforme y perfectamente conocido. Una buena organización de las cuentas de usuario facilita sobremanera el análisis de los datos de los registros.
e. Los grupos de usuarios deben estar perfectamente definidos y justificados, y en ningún caso deben ser sumamente numerosos: esto facilitará enormemente al administrador la asignación de responsabilidades y la localización de errores o vulnerabilidades, e inclusos intentos de invasión local, si lo hubiere.

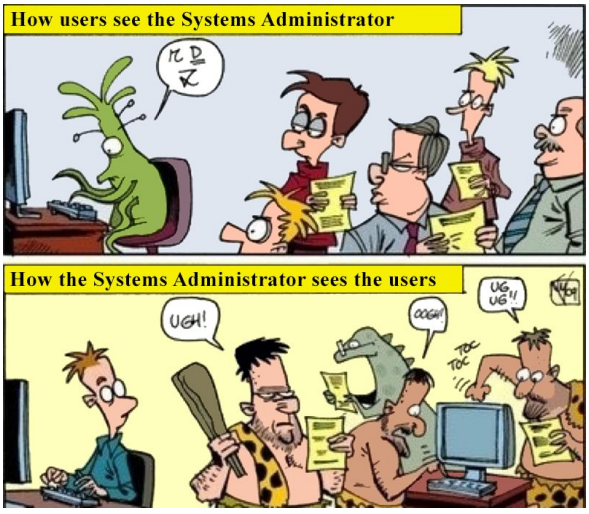


Figura 7. Ilustración: "How users see the sysadmin…"

En muchas ocasiones, se tienen en el sistema cuentas de usuario abandonadas con privilegios suficientes como para comprometer la integridad de éste. Este tipo de cuentas proporcionan un vehículo ideal para los intrusos.

Sobre la cuenta del root

Para cualquier invasor, el 'Santo Grial' del sistema objetivo es la cuenta del superusuario. Las razones son más que evidentes, y ya se han ido adelantando: la cuenta del root tiene poder para hacer y deshacer a su antojo, y control sobre la totalidad de los recursos e información almacenada en el sistema. Tiene autoridad total sobre la máquina y los usuarios, lo que también incluye a todas las máquinas y usuarios conectados a través de la red. Un administrador debe tener grabado a fuego que la cuenta de root debe reservarse para propósitos muy específicos, cuando sea estrictamente necesario. En el resto de ocasiones, el root puede y debe contar con una cuenta de usuario común. Un pequeño error cometido como administrador puede ser el germen de una catástrofe. Una regla heurística que se debe aplicar en este sentido puede ser: "cuanto menor sea el tiempo en que el administrador trabaja con todos sus privilegios, más seguro estará". Un gran poder conlleva una gran responsabilidad. El usuario administrador debe atenerse a una serie de mandamientos estrictos, si su intención es hacer bien su trabajo:

a. Cuando se trata de ejecutar un comando complejo o que puede conllevar pérdida de información sensible, éste debería ejecutarse primero de forma no destructiva, antes de tener que lidiar con las consecuencias. Esto resulta especialmente interesante en el caso de órdenes que aplican la agrupación mediante el carácter asterisco (*): por ejemplo, si vamos a ejecutar el comando rm fich*.txt, primero deberíamos ejecutar ls fich*.txt a fin de observar un listado de los archivos que van a ser borrados antes de hacer efectiva la eliminación. El uso del comando *echo* puede ser de ayuda también en esta dirección.
b. Proporcionar a los usuarios un alias predeterminado para el comando rm que solicite una confirmación ante cualquier proceso de borrado de información.
c. Usar la cuenta de root sólo para tareas específicas, siempre que sea estrictamente necesario e imposible por otros medios. Si es preciso intentar averiguar cómo ejecutar algún proceso en beneficio del sistema es mejor regresar a una Shell de usuario común hasta tener claro el procedimiento a seguir.
d. La ruta de comandos del usuario root es sumamente importante. Esta ruta (es decir,

el PATH, variables de entorno) especifica los directorios en los que la Shell buscará los programas que debe iniciar a merced de la ejecución de un comando reconocido. Es recomendable limitar la ruta de comandos para el usuario root tanto como sea posible, y nunca incluir el directorio raíz (.) en el PATH, dado que ésto habilitaría a un posible invasor para sustituir comandos del sistema por otros que escapan al control del superusuario. Asimismo, el administrador no debería tener directorios en el PATH con la escritura habilitada, ya que esto puede dar oportunidades a los atacantes para incluir binarios o realizar modificaciones en determinados directorios, permitiéndoles realizar una escalada de privilegios la próxima vez que ejecuten la orden en cuestión.

e. Evitar a toda costa el uso del conjunto de herramientas rlogin/rsh/rexec (también conocidas como "r-utilities") desde la cuenta root. Estas herramientas están sujetas a gran cantidad de ataques y entrañan grave peligro cuando son ejecutadas con permisos de administración. Bajo ningún concepto debemos crear un archivo .rhosts para el root.
f. El archivo /etc/securetty contiene una lista de los terminales desde las que el root puede loguearse. Lo más recomendable es configurar este fichero para dar acceso únicamente a las consolas virtuales locales, sin excepciones. En caso de requerir acceder de forma remota al sistema, el administrador debe ser capaz de realizar su entrada como usuario normal y una vez dentro cambiar a la cuenta del root, si así lo requiere.
g. La cautela y la deliberación son cualidades más que deseables en un administrador, pues sus acciones repercuten en la totalidad del sistema. Pensar antes de teclear es una decisión acertada; ir con pies de plomo siempre es mejor que arrepentirse de sus acciones.

En caso de necesitar que un tercero adquiera los privilegios del root -por la suerte del administrador esperemos que sea de su plena confianza- algunas herramientas pueden allanar el camino: la utilidad `sudo` permite a un usuario normal y corriente usar su propia contraseña para ejecutar un conjunto limitado de comando del usuario root. Esta utilidad también mantiene un registro de todos los éxitos e intentos fallidos de obtener los privilegios, permitiendo al administrador rastrear quién usó qué comando y con qué fin. Eso explica en gran el éxito de que goza esta herramienta. Aun así, cuenta con algunos defectos, por lo que -como sabrá ya de sobra el lector-, no conviene confiarse. Debemos restringir su uso a un conjunto controlado de tareas, como la adición de nuevos usuarios. Si no limitamos su uso, enseguida los atacantes encontrarán vías para explotar el `sudo` en contra del sistema. Un programa tan "inofensivo" como /bin/cat puede ser usado para sobreescribir archivos, una vía ideal para explotar la cuenta del root. Si la utilidad `sudo` se concibe como reemplazo del root, el sistema está en peligro.

3.4. Integridad de la información: Sistema de Ficheros

Invertir tiempo en planificar una buena configuración del sistema de ficheros puede marcar la diferencia a la hora de salvaguardar la información de la empresa y garantizar la confidencialidad y supervivencia de los datos almacenados en él. Algunas pautas iniciales que el administrador debe seguir son las siguientes:

a. En principio, no existe razón alguna para que los directorios de los usuarios comunes contengan programas con permisos de tipo SUID/SGUID [véase el apartado sobre "Gestión de permisos especiales a ficheros y directorios" en este mismo capítulo, donde encontrará información más precisa sobre bajo qué circunstancias debe sospecharse del uso incorrecto del setuid y cómo debe restringirse su uso]. Por esta razón, se recomienda hacer uso de la opción de configuración *nosuid* en el fichero /etc/fstab para asegurarse de que no existen ficheros con el setuid activado en particiones donde la escritura está permitida para otros usuarios más allá del propio administrador. Asimismo, se recomienda aplicar las opciones nodev y noexec en los directorios /home de los usuarios convencionales, y en el directorio /var (donde se alojan, entre otros, los ficheros temporales), inhibiendo de esta manera la ejecución de programas y la interpretación de los dispositivos especiales o de bloques del sistema de ficheros.
b. Si el sistema de ficheros permite la exportación mediante NFS, el administrador debe configurar el directorio /etc/exports de la manera más restrictiva que sea posible. Huir del uso indiscriminado de 'wildcards', impedir el acceso a escritura en la raíz del sistema y realizar la exportación como 'solo-lectura' son principios muy recomendables que deben seguirse siempre que sea posible.
c. Establecer ciertas limitaciones en el uso del sistema de ficheros es una buena pauta, en lugar de proporcionar un acceso ilimitado. Para limitar el acceso a los recursos de cada usuario deben usarse lo que se conocen como módulos PAM, que se tratarán en secciones posteriores.
d. Debe garantizarse la integridad de los ficheros /var/log/wtmp y /var/log/utmp, pues contienen los registros de sesiones que han tenido lugar en el sistema. El fichero utmp nos ofrece información sobre los usuarios que están usando el sistema actualmente, mientras que wtmp registra todos los inicios y finales de sesión. El administrador debe asegurar que bajo ningún concepto estos ficheros del sistema son modificados bajo ningún concepto, y deberá acudir a ellos como medida inicial si detecta problemas de vulnerabilidad. Un análisis minucioso podría ayudar a determinar quién es el intruso y en qué momento dejó huella de su paso por su sistema. Estos ficheros deben tener asignado permisos de tipo 644, para no interferir con el funcionamiento de otros servicios del sistema que hacen uso de ellos.
e. El comando chattr debe ser un aliado del administrador, a fin de proteger los archivos del sistema de cambios involuntarios o borrados accidentales. El root puede indicar sobre los ficheros o directorios que desea proteger las órdenes que observa a continuación, y a partir de ese momento estos ficheros y directorios designados no podrán ser borrados, incluso por el root, a menos que el bit «i» ("immutable") sea retirado con la opción complementaria -i:

```
sudo chattr +i myfolder

sudo chattr +i myfile.txt
```

f. Se recomienda encarecidamente retirar el permiso de escritura global a todos los ficheros y directorios que lo tengan habilitado y en especial a aquéllos cuyo uso se desconozca o no se controle, pues son una fuente de ataques directa en caso de intrusión, más aún si se trata de archivos y directorios críticos del sistema. Para monitorizar qué ficheros tienen este tipo de permiso habilitado puede ejecutarse la orden:

```
find / -perm -2 ! -type l -ls
```

g. Hacer una exploración de los ficheros del sistema sin propietario puede ser una

buena idea como punto de partida, pues pueden ser una indicación evidente de intrusión al sistema:

```
find / \( -nouser -o -nogroup \)
-print
```

h. Eliminar todos los ficheros del tipo .rhosts del sistema debe ser una prioridad, pues permiten a un atacante acceder a toda la red desde una cuenta de usuario que haya sido explotada. Podemos consultar si tenemos ficheros de este tipo con la orden:

```
find /home -name .rhosts -print
```

i. El comando umask puede usarse para determinar el modo de creación por omisión de nuevos archivos en el sistema. Si los archivos se crean sin tener en cuenta la configuración de sus permisos, se podría estar concediendo acceso de lectura o escritura a usuarios no autorizados. Las configuraciones típicas de umask incluyen 022, 027 y 077 (que es asu vez la más restrictiva). Generalmente, la configuración se fija en /etc/profile, con aplicación sobre todos los usuarios del sistema.

Ficheros especiales: capacitación de acceso a hosts

Si el administrador lo permite, el sistema Linux puede contar con un conjunto de ficheros de configuración que habilitan el acceso a determinados equipos que proporcionan servicios de red específicos. No obstante, un mínimo fallo en los parámetros de configuración de los mismos liquida de un 'plumazo' todo el trabajo en materia de seguridad a nivel local. Históricamente, errores en estos archivos han abierto las puertas de par en par a los 'piratas' informáticos, por lo que evitar su existencia debe ser una labor prioritaria en la hoja de ruta de cualquier administrador:

a. Fichero .rhosts: capacita a un usuario para establecer un conjunto de equipos (y otros usuarios) que tienen poder para utilizar su cuenta de acceso sin necesidad de contraseña, sino a través de las famosas 'r-utilities', que ya hemos introducido en capítulos previos, a saber: rsh, rcp, … Como intuye ya el lector, una invitación a la intrusión de este calibre no debe permitirse en el sistema bajo ningún concepto. La explicación es simple: una configuración deficiente autoriza el acceso a usuarios no acreditados; o peor aún, nada impediría a un atacante con acceso a la cuenta del usuario modificar este fichero para indicar los equipos y usuarios que le son más convenientes para perpetrar un ataque. El administrador debe, por tanto, analizar el sistema en busca de ficheros de este tipo y eliminarlos y, además, deshacerse de cualquiera de las 'r-utilities' citadas.
b. Fichero /etc/hosts.equiv: es el archivo 'gemelo' del anterior, pero a nivel de máquina, que permite especificar el conjunto de usuarios, grupos y servicios que tienen poder para hacer uso de los 'comandos r'. Conviene emplear alternativas como ssh para dotar al sistema de servicios en esta línea.
c. Fichero /etc/hosts.lpd: es un tipo de fichero para proporcionar servicios de impresión que ya se encuentra prácticamente en desuso (si bien no es desaconsejable que el administrador invierta cierto tiempo en corroborar la ausencia del mismo). Este archivo ha sido históricamente un pasillo para infecciones provocadas por 'gusanos' y ataques de desbordamiento (*buffer overflows*).

Gestión de permisos especiales a ficheros y directorios: sticky-bit y setuid

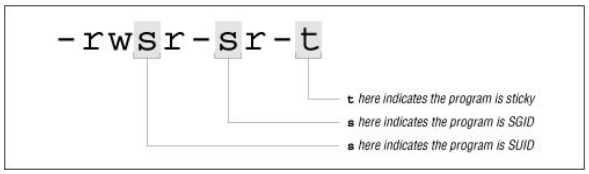


Figura 8. Comprobación de los permisos especiales en linux.

El sticky-bit es un tipo de permiso de carácter excepcional que se concede a determinados directorios para permitir a usuarios de diferentes grupos realizar procesos de escritura en la propia ruta, si bien únicamente el root o el propietario del fichero en cuestión dentro del directorio tiene potestad para borrar esta fuente de información. Un ejemplo clásico de este tipo de directorios son las carpetas temporales (como el directorio temporal /tmp). Aunque a primera vista puede parecer una característica útil, de cara a la seguridad del sistema es un arma de doble filo, por lo que resulta preferible contar con el menor número posible de directorios de este tipo y tener bajo vigilancia los cambios que en ellos se produzcan. Para conceder este permiso se empleará la orden `chmod +t dirname`, y para inhabilitarlo su opuesta `chmod -t dirname`. Conviene que el administrador realice un análisis preliminar de los ficheros que tienen el sticky-bit activo y decida si debe intervenir al respecto: para hacerlo, puede ejecutar un ls -la y comprobar si el carácter «t» aparece al final de la cola de permisos de cada directorio.

Por otro lado, el setuid es un permiso especial que se utiliza para permitir a un usuario ejecutar determinado código (programas ejecutables o guiones Shell) con los permisos de otro usuario. Si nos encontramos con programas que tienen activo el setuid del root, debemos intervenir de inmediato, pues podrían estarse utilizando como '*backdoors'*, donde un usuario corriente realiza una escalada de privilegios para ejecutar código con los poderes del superusuario, de forma transparente al propio sistema. Por lo general, conviene evitar prácticas de este tipo, por razones ya más que evidentes y eliminar el setuid siempre y cuando no se considere imprescindible su utilidad o su uso no se controle íntegramente. Ésto puede lograse a través de la orden `chmod -s` sobre el fichero en cuestión.

En un primer contacto con el sistema, el administrador debe hacer una exploración minuciosa a fin de encontrar todos los programas de tipo SUID/SGUID que existen en el sistema, valorar si su uso es conocido y responsable, y deshacerse de cualquier atisbo de duda acerca de si existen *'puertas traseras'* en el sistema de archivos. El root puede apoyarse en la utilidad 'find' para llevar a cabo esta exploración:

```
find / -type f \( -perm -04000 -o
-perm -02000 \)
```

Asimismo, la distribución Debian en la que estamos realizando las pruebas de administración nos ofrece la posibilidad de llevar un registro de los cambios en los ficheros SUID que existen en el sistema: podemos consultar el log */var/log/setuid** para obtener dicha información.

Herramientas avanzadas de monitorización del Sistema de Ficheros

Una manera de incrementar el nivel de seguridad del sistema a la hora de garantizar detección de ataques a escala local -y asimismo a nivel de red- consiste en disponer de un monitor de integridad como Tripwire, Aide u Osiris. Estos comprobadores de integridad se basan en la realización de sumas de verificación en todos los binarios críticos del sistema y archivos de configuración, a fin de detectar discrepancias y cambios con respecto de versiones anteriores de estos mismos ficheros que el administrador considera que se deben tomar como referencia. De esta manera, cualquier mínimo cambio en cualquiera de estos ficheros esenciales dejará una marca.

Este tipo de verificadores deberían convertirse en un compañero inseparable del 'sysadmin' para detectar intrusiones antes incluso de que el propio administrador las detecte. Podemos encontrar una versión gratuita inicial de Tripwire a modo de prueba, disponible directamente en el sitio web del proveedor: https://www.tripwire.com/free-downloads.

Figura 9. Tripwire como herramienta de apoyo a la administración de sistemas | Fuente: Tripwire

3.5. Política de contraseñas y encriptación

En la actualidad, los diferentes 'sabores' de Linux incorporan un nivel de seguridad adicional respecto de las contraseñas de los usuarios del sistema. En las primeras versiones, dichas contraseñas se incluían directamente el fichero /etc/passwd, perfectamente legible para cualquier usuario del sistema, provocando un agujero de seguridad astronómico si un atacante irrumpía en el mismo. Lo más común en los tiempos que corren es, en primer lugar, trasladar las claves de acceso a un fichero de 'sombras' (/etc/shadows), que debe garantizarse es exclusivamente legible por el usuario root. El fichero /etc/passwd sigue existiendo, al servicio de las aplicaciones del sistema, si bien las contraseñas han sido reemplazadas en él por un *, por lo que la información de acceso no puede ser explotada usando este fichero. Esta configuración es la empleada por defecto y también la más recomendable.

Un segundo escalón de seguridad consiste en encriptar todas y cada una de las contraseñas presentes en el fichero de shadow mediante algoritmos de cifrado como md5 -el que ofrece, por ejemplo, nuestra distribución Debian-, que resultan mucho más difíciles de romper. Podemos identificar las contraseñas cifradas en el fichero de shadow dado que son las secuencias de caracteres que vienen precedidas por el prefijo «`$id$`». Para cifrar una contraseña en un entorno Linux suele hacerse uso del comando mkpasswd, a semejanza del proceso que ya seguimos en su momento para la protección del bootloader [véase el capítulo de "Seguridad de los componentes físicos" para más detalles al respecto].

Una política de buenas prácticas que el administrador debe seguir respecto del protocolo de contraseñas consiste en obligar -decimos bien e insistimos en el término de "obligar", siempre por medios computacionales- a los usuarios del sistema a cambiar su contraseña en el período que el sysadmin estime más oportuno, impidiendo el acceso una vez haya vencido el tiempo de permuta. Se avisará a los usuarios convenientemente sobre el tiempo que tienen para realizar sus cambios. El comando `chage` -léase bien que decimos '*chage*', no «change»- puede ayudar al root en todas estas funciones de administración de claves de acceso. El empleo de una política de contraseñas con reglas preestablecidas -pero nunca siguiendo un formato fijo- (con cierto número exigido de caracteres especiales, por ejemplo) también es una práctica recomendable, a fin de evitar contraseñas simples o incluso vacías.

Asimismo, tal y como expusimos recientemente los autores en el seminario del presente curso, en el ámbito de la seguridad de acceso y contraseñas, el administrador debe 'pasar al lado oscuro' -siempre con la responsabilidad que su trabajo le demanda, por supuesto-, disfrazarse

por momentos de atacante y disponer de herramientas que sirvan para romper contraseñas de acceso débiles, siempre con la meta de actuar sobre su propio sistema y fortalecerlo. 'John The Ripper' es la herramienta por excelencia para ejecutar ataques de fuerza bruta y de diccionario sobre las contraseñas de los usuarios del sistema, y verificar si existen vulnerabilidades que un cracker real pudiese explotar. Es por ello que conviene instruir a nuestros usuarios en el uso de claves suficientemente aleatorias, que al menos no puedan ser explotadas por ataques de diccionario, y avisar a los usuarios imprudentes cuya contraseña es vulnerable por estas técnicas. No obstante, el administrador debe tener presente que herramientas de tales características se encuentran al filo de la navaja, entre la seguridad y la intrusión, por lo que hacer un uso indebido de ellas está sometido al peso de la legalidad vigente. Investigar sobre la criptografía de clave pública y clave privada puede ser un paso adicional a tener en cuenta.

3.6. Seguridad en red

Seguridad en el lado del cliente

Toda empresa que se precie emplea servicios de terceros para proporcionar los suyos propios, si bien en principio nada garantiza al sistema y a sus propios usuarios que aquello que se está utilizando y que proviene de fuentes externas a la empresa es completamente seguro, por lo que en la hoja de ruta del administrador debe estar la labor de defender al sistema del exterior y también la de proteger a sus usuarios. En ocasiones, incluso de ellos mismos. Es fundamental el uso de servicios que utilicen cifrado de contraseñas y de los datos como los vistos, así como técnicas de conexión de red cifradas, a saber: SSH y SSL/TLS. En el ámbito empresarial, conviene abandonar servicios sin herramientas de encriptación como ftp, telnet, correos inseguros, etc.

En esta línea, contar con medios de autenticación robustos, como los integrados en los propios SSH y SSL/TLS permiten al sistema verificar la identidad de los servidores de conexión y evitar peligros potenciales como los servidores falsos. Asimismo, contar con herramientas de filtrado de paquetes, como los cortafuegos (firewalls) nos permite realizar un control exhaustivo sobre los paquetes recibidos de fuentes ajenas.

Servicios de protección en el lado del cliente: ssh y stelnet

Los servicios ssh (Secure Shell) y stelnet (Secure Telecommunication Network) son suites de programas que nos permiten acceder a sistemas remotos con mecanismos de encriptación de la conexión.

En este sentido, podemos destacar también Openssh, un conjunto de programas que se utiliza como sustituto seguro de rlogin, rsh y rcp (las famosas 'r-utilities' que hemos ido mencionando en capítulos previos). Openssh emplea la clave pública para encriptar las comunicaciones entre dos anfitriones, así como para autenticar a los usuarios en los respectivos sistemas. Puede utilizarse para acceder de forma segura a un host remoto o copiar datos entre hosts, y además se previenen los ataques ejecutados por intermediarios -como el secuestro de sesión por *sniffers* de red y la falsificación de DNS-. Es de código abierto y totalmente libre bajo licencia BSD.

Seguridad en el lado del servidor: inetd

GNU/Linux existe un fichero de configuración clave para planificar el protocolo a nivel de red en el lado del servidor, que el administrador puede hallar en la ruta /etc/inetd.conf. Se trata de un fichero del sistema que facilita la labor de gestión de los diferentes servicios y sus condiciones de arranque. Esencialmente, este fichero será utilizado por el demonio inetd, conocido con el nombre de "superservidor de red". Este demonio tiene la labor de proporcionar los servicios del propio servidor al resto de máquinas del escenario, por lo que su

configuración debe ser cautelosa. Cada línea del presente fichero de configuración indicará al demonio cómo debe comportarse cuando recibe una conexión a través de un puerto designado.

Otro aspecto a tener en cuenta de cara a los servicios de red que deben iniciarse con el arranque del sistema consiste en fijar adecuadamente el runlevel o nivel de ejecución. Se recomienda al administrador echar un vistazo al fichero /etc/inittab para determinar qué comportamiento debe tomar el sistema en cada nivel de ejecución, dejando espacio para modos de recuperación que pueden resultar útiles llegado el caso; por ejemplo, para iniciar el sistema en un modo completamente desconectado si se ha detectado una posible intrusión. Por defecto en la mayoría de distribuciones el sistema arrancará en el runlevel 5 (modo multiusuario gráfico con soporte de red), pero puede y debe configurarse si el sistema lo requiere.

Otros ficheros que proporcionan soporte adicional a la administración en la salida a red, son: /etc/services y /etc/protocols. El sysadmin debe hacer uso de ellos para consultar periódicamente qué servicios locales o de red están iniciados en el sistema y le son desconocidos, si los hay; y para obtener una lista de los protocolos en uso. Este tipo de ficheros pueden servir como una fuente de consulta para conocer en todo momento los comandos y servicios en ejecución. Asimismo, en el archivo /etc/rpc encontraremos una lista de servidores RPC y sus puertos asociados.

Si el administrador sospecha de algún servicio o cree conveniente evitar su uso, basta con comentar la línea correspondiente del fichero de configuración /etc/inetd.conf mediante el carácter #.

Herramientas avanzadas de seguridad en red

Existe un número considerable de productos de software que permiten a un administrador responsable intentar explotar su propio protocolo de seguridad a nivel de red a través del escaneo de puertos y servicios de red. SATAN, ISS, SAINT y Nessus son algunas de las herramientas más conocidas.

En este caso, el software se conecta a la máquina o máquinas objetivo del sistema a través de todos los puertos que sea posible, e intentan determinar qué servicio está funcionando en ese puerto. Basándose en esta información, se puede conocer de forma sorprendentemente sencilla si el objetivo es vulnerable en ese servidor. Nessus es un escáner de seguridad totalmente gratuito y con interfaz gráfica de usuario, que permite al administrador realizar este escaneo de conexiones, siempre bajo su propia responsabilidad: http://www.nessus.org.

Sobre los servicios de correo electrónico

Es altamente probable que nuestro servidor esté destinado a proporcionar un servicio de correo electrónico. Históricamente, los servicios de correo han sido un vector de ataque suculento para los crackers, debido al número de tareas que debe realizar y los privilegios que demanda. En este sentido, podría recomendarse la última versión de sendmail, si bien ha sido objetivo histórico de ataques. Con el objetivo central de la seguridad han surgido alternativas como qmail -estable, rápida y fiable- y postfix, de IBM. Todas ellas son herramientas de tipo MTA (Mail Transfer Agent), es decir, aplicaciones que enrutan y transmiten correo electrónico de un nodo de una red a otra utilizando protocolos como SMTP.

3.7. Seguridad de las aplicaciones

Las aplicaciones del sistema deben contar asimismo con un método de autenticación que verifique la identidad de los usuarios que hacen uso de las mismas. En esta línea, los módulos PAM son el mejor aliado del administrador para llevar un registro de los usuarios con acceso a determinadas aplicaciones. Se trata de un compendio extenso de bibliotecas compartidas que el administrador puede incorporar a los programas de usuario con estos fines, permitiendo incluso variar el método de identificación de usuarios (contraseña, token, biometría, etc) sin necesidad de modificar la aplicación en absoluto.

Esto se consigue por medio de una serie de ficheros objeto (módulos o bibliotecas) cargables dinámicamente, alojados por lo general en la ruta /lib/security. El proceso de configuración de los módulos PAM se realizará a través del directorio /etc/pam.d, donde encontramos el fichero de configuración propio de cada aplicación a la que se hayan incorporado los módulos. Aquí reside precisamente una de las ventajas fundamentales de los módulos PAM, dado que podemos tener un método de autenticación pormenorizado por aplicación. De hecho, ya existen por omisión numerosos servicios del sistema que hacen uso de módulos PAM. Algunos de ellos son: passwd, ssh o login.

Los módulos PAM proporcionan cuatro tipos de servicios: de autenticación (auth), de administración de cuentas (account), de gestión de sesiones (sesión) y de gestión de contraseñas (password). Para listar los servicios Linux que usan Linux-PAM debe escribirse la orden: `ls /etc/pam.d/.`

Los ficheros del directorio /etc/pam.d tienen una estructura elemental basada en cuatro campos, a saber:

```
Tipo-de-módulo      Flag-de-control
Ruta-del-módulo     Argumentos
```

El tipo de módulo especifica cualquiera de los servicios indicados en el párrafo anterior. En cuanto al flag de control, se utiliza para indicar si el servicio proporcionado por el módulo es necesario (required), requisito previo (requisite), suficiente (sufficient) u opcional (optional). El tercer campo fija la ubicación del módulo y el último consta de una serie de parámetros pasados al módulo en cuestión. Los argumentos que pueden pasarse al módulo dependen de éste.

3.8. Un nivel de seguridad adicional: SELinux y AppArmor

Tradicionalmente, la seguridad de los sistemas estaba supeditada a técnicas DAC (Control de Acceso Discreto), esto es, donde cada programa decide su nivel de acceso a sus propios recursos. Esto otorgaba a los usuarios potestad total sobre los recursos de su pertenencia, por lo que la seguridad del sistema dependía en buena medida de las vulnerabilidades de las propias aplicaciones. Pronto se vio que este sistema comprometía fácilmente la integridad completa del sistema, que quedaba a merced de la integridad de las propias aplicaciones. De esta manera, se implantan los mecanismos conocidos como MAC (Control de Acceso Obligatorio), donde la política de seguridad es definida por el administrador, que toma control total para decidir sobre el derecho de acceso a los recursos y los privilegios de las aplicaciones.

SELinux es un componente más dentro de la política MAC, incluido en las versiones del núcleo del sistema operativo desde su liberación en el año 2000. Dicho componente permite tener un control de grano fino sobre los permisos, como permitir añadir información a los archivos de registro, pero nunca sobreescribirlos o truncarlos, estrategias habituales en atacantes que deben borrar las huellas de su intrusión en el sistema. Permitir que determinadas aplicaciones realicen conexiones únicamente sobre puertos específicos también es una opción, poniendo el freno al acceso de troyanos y *backdoors* al denegar puertos a discreción del administrador.

En SELinux, los procesos se ejecutan en un determinado contexto y los recursos tienen asignado un tipo, fijando las acciones que se pueden llevar a cabo mediante una serie de reglas estrictas para cada contexto y tipo, así como los flujos de información que están permitidos. Asimismo, SELinux presenta dos modos de funcionamiento: *permissive* y *enforcing*. El modo 'permisivo' permite los accesos no autorizados, si bien se registran en los logs correspondientes (`var/log/messages`, principalmente). En el modo exhaustivo, cualquier intento de acceso que viole las políticas definidas se corta de inmediato.

Asimismo, la herramienta permite definir dos modos de política de seguridad: *targeted*, donde la mayor parte de los procesos operan sin limitaciones y se confina a los servicios y demonios del sistema bajo contextos de seguridad específicos; y *strict*, donde absolutamente todo proceso está supeditado a su propio contexto de seguridad particular y cada acción se controla en relación a las políticas definidas.

No obstante, a pesar de ser un servicio increíblemente potente, la definición de reglas en materia de política de seguridad puede complicarse hasta llegar a extremos donde se haga insostenible, e incluso definir una política demasiado restrictiva podría provocar el funcionamiento deficiente de determinados servicios. Con todo, nunca deberá considerarse a SELinux un sustituto de las políticas de seguridad definidas en la presente documentación, sino un complemento adicional.

Justificando precisamente lo complicado que resulta el manejo de SELinux en sistemas avanzados a largo plazo, muchos administradores recomiendan migrar a herramientas similares como AppArmor, que permite asociar a cada programa un perfil de seguridad que restrinja su ámbito de acción y capacidades de acceso, según lo requiera. La diferencia con SELinux reside en la manera de aplicar las políticas, donde el perfil se asigna a la ruta de instalación del programa a ejecutar, a diferencia de SELinux, donde las reglas se aplican con etiquetas a los propios ficheros.

4. Detección y protección frente a intrusiones

4.1. Detectando intrusos

La detección y prevención de intrusiones a nuestro sistema es el siguiente paso que debemos realizar una vez hemos establecido nuestra seguridad de un modo más o menos correcto. Para ello emplearemos los sistemas IDS (sistemas de detección de intrusos, usando nuestros recursos o explorando nuestro sistema con el fin de encontrar fallos de seguridad). Dichos sistemas crean procedimientos para escuchar y generar alertas en caso detectar alguna situación sospechosa (un posible accidente de seguridad). Existen sistemas basados tanto en información local (para obtener información de registros del sistema, detectar cambios en los ficheros del sistema o simplemente de la configuración de servicios típicos) como basados en red (para evitar comportamientos anómalos como suplantaciones, situaciones de tráfico anómalo, interfaces de red en modo promiscuo...). Algunas herramientas de este tipo serían *Logcheck*, *Tripwire* (y su versión libre *Aide*) y *Snort*.

4.2. Medidas de protección

Protección mediante filtrado:

TCP Wrapper.

Los *TCP Wrapper* son un software intermediario entre peticiones del usuario de un servicio y los daemon de los servidores que lo ofrecen. Muchas distribuciones ya vienen con los *wrapper* activados, siendo los niveles de acceso configurables. Además, se pueden combinar con *inetd* o *xinetd* (protegiendo los servicios que ofrecen). En resumen, el *wrapper* reemplaza el *daemon* del servicio por otro que actúa como intermediario (*tcpd* que se encuentra normalmente en */usr/bin/tcpd*). Tras recibir una petición, verifica el usuario y el origen de la petición para ver si la configuración del wrapper del servicio lo hace o no utilizable. Por otro lado facilita la generación de registros e informar por correo de posibles intentos de acceso, para ejecutar a continuación el daemon correspondiente al servicio. El sistema de los *wrapper* es controlado desde el fichero */etc/hosts.deny*, lugar en el que especificamos qué servicios negamos y a quién se los negamos y el fichero */etc/hosts.allow,* donde especificamos el servicio que utilizaremos y quiénes pueden acceder. Por otra parte destacamos en este apartado los comandos *tcpdhck* (que comprueba que las configuraciones de los ficheros *hosts* sean las adecuadas) y *tcpdmatch* (nos dice cómo se comportará un sistema tras darle un servicio y un cliente).

Firewall o cortafuegos.

Sistema o conjunto de sistemas empleados para reforzar las políticas de control de acceso entre diferentes redes. Por regla general, o bien lo empleamos para proteger una máquina concreta que está conectada a Internet, o bien protegemos nuestra red interna a través de una o varias máquinas. De entre las diferentes posibles soluciones, la mejor considerada es usar un ordenador que disponga de más de una tarjeta de red para aislar de este modo las diferentes redes conectadas. De esta manera el cortafuegos decide qué paquetes pasan, cuales no y a que red. Si lo combinamos con un *router*, podemos enlazar paquetes de diferentes redes. Podemos usar una única tarjeta de red que conecta una máquina individual directamente a Internet, cuando el objetivo sea protegerla de ataques o tráfico indeseado. Por otro lado, con dos tarjetas de red, podremos dedicar una al tráfico con Internet y otra al de nuestra red interna para eliminar el tráfico del que no somos receptores, así como controlar el que va a Internet. En general, el cortafuegos permite al usuario definir unas políticas de acceso a través del control de los puertos *TCP* y *UDP* que están autorizados para entrada o salida. Podemos también mencionar la existencia de la técnica *NAT*, que oculta de Internet las direcciones IP que se usan para la red privada, manteniendo eso sí, el acceso desde las máquinas. Concretamente, el *NAT masquerading*, permite que uno o varios dispositivos de la red, sean vistos como una única dirección IP desde fuera (un ejemplo sería el enrutador ADSL que ofrece una combinación de esta técnica con cortafuegos).

Paquetes para la gestión de cortafuegos en las distribuciones: Podemos destacar en este punto algunas herramientas de configuración (teniendo en consideración que generalmente no ofrecen las mismas prestaciones que la configuración manual de *iptables*, de la que hablaremos en el siguiente punto):

a. *Lookkit*: Permite al usuario decidir si quiere un nivel de seguridad alto, medio o bajo. Una vez seleccionado nos muestra los servicios afectados para que podamos dejar pasar o no a los diferentes servicios. Se emplea principalmente en la distribución *Fedora/Red Hat* pero es posible instalarlo en *Debian* o una versión gráfica (*gnome-lokki*t).

b. *Bastille [Proa]*: Se trata de un programa didáctico que nos muestra paso a paso varias recomendaciones de seguridad y la configuración del cortafuegos. Se emplea en *Fedora* y *Debian*.
c. *fwbuilder*: En este caso estamos ante una herramienta de construcción de reglas de cortafuegos de un modo gráfico. Se emplea en múltiples sistemas operativos como *Fedora*, *Debian* o incluso *MacOs*, empleando diferentes tipos de cortafuegos.

Pese a contar con cortafuegos correctamente configurados, pueden ser burlados por determinados ataques, es más, las necesidades actuales de conectividad pueden hacer necesaria la creación del software que permita el paso a través de los cortafuegos. Podemos señalar como ejemplo los códigos móviles por web como son *ActiveX*, *Java* o *JavaScript*, que al cruzar los cortafuegos, dificultan la protección de los sistemas en caso de que sean vulnerables frente a agujeros descubiertos. Como conclusión, los cortafuegos son una buena protección frente a amenazas de seguridad, pero como todo, tiene vulnerabilidades frente a ataques. En seguridad nunca debemos esperar que una solución nos proteja absolutamente y debemos ser capaces de examinar los posibles problemas y detectarlos a tiempo con políticas de prevención.

Netfilter: iptables.

El Netfilter consiste en un subsistema de filtrado, proporcionado por el núcleo de Linux. Una de las interfaces de filtrado que soporta es *iptables*, que es de hecho, la más utilizada. La interfaz de este comando permite llevar a cabo las tareas de configuración de las reglas que conciernen al sistema de filtrado. Para arrancar se usa */etc/init.d/iptables start* (a no ser que estuviese ya configurado previamente en el *runlevel*). El parámetro `-L` del comando `iptables`, permite listar las reglas activas en el momento para cada una de las cadenas o *chains*. En caso de no haberse llevado a cabo una configuración previa, por defecto se aceptan todos los paquetes de las cadenas *INPUT*, *OUTPUT* y *FORWARD*. El nivel superior del sistema de *iptables* son las tablas (cada una con diferentes cadenas y cada cadena con diferentes reglas). Existen tres tablas:

a. *Filter*: Normas de filtrado.
b. *NAT*: Traslación de direcciones en el interior de un sistema que utiliza NAT.
c. *Mangled*: Opciones de control y gestión de paquetes.

Podemos mencionar algunos aspectos de configuración de la tabla Filter:

a. `iptables -A chain -j target`: *chain* puede ser cualquiera de las tres cadenas mencionadas anteriormente, *target* es el destino que se le asignará al paquete correspondiente con la regla. Con la opción `-A` añadimos la regla a las que ya existen. La opción menos `-j` por otra parte indica qué hacemos con los paquetes (*accept*, *reject* o *drop* - este último significa perderlos simplemente y a diferencia de *reject* no se emite ningún tipo de respuesta). Otro target posible es log que se usa para enviar el paquete al sistema de registros, en este caso suelen aparecer dos reglas, una con *log* y otra con *accept*, *reject* o *drop*.
b. `iptables -I chain position -s source -d destination -j target`: La opción `-I` se puede utilizar para indicar una posición en la que se coloque la regla en la cadena. La opción `-s` indica el origen de los paquetes y -d el destino.
c. La opción `-D` nos permite borrar una regla en una posición de la cadena, o directamente la regla exacta:

```
iptables -D chain position
iptables -D chain -s source -j
      target
```

d. La opción `-P` permite definir una política por defecto para los paquetes. Unos ejemplos de esta orden serían:
   - `iptables -A INPUT -s 20.0.0.0/8 -d 192.168.1.2 -j ACCEPT`: Aceptamos los paquetes procedentes de

algún elemento de la red 20.0.0.0 con destino a 192.168.1.2

- `iptables -D OUTPUT 2`: Borra la regla en la segunda posición de la cadena *OUTPUT*.
- `iptables -L`: Muestra las reglas activas para cada cadena.

En lo que respecta a nombres de protocolos y puertos el sistema *iptables*, emplea la información de los ficheros */etc/services* y */etc/protocols*. La configuración de *iptables* normalmente se lleva a cabo a través de llamadas consecutivas al comando *iptables* con las reglas. Las reglas se pueden hacer permanentes, por ejemplo en *Fedora* para que queden guardadas en */etc/sysconfig/iptables*, se puede usar el comando */etc/init.d/iptables save.* En *Debian* si existe el script mencionado previamente puede hacerse con el comando */etc/init.d/iptables save rules-name* (debe existir el directorio */var/lib/iptables* que es donde se encuentra el fichero *rules-name*). Si no existe soporte directo para poder guardar o recuperar las reglas podemos hacer uso de los comandos *iptables-save* e *iptables-restore* para redirigirlas a ficheros. El comando */etc/init.d/iptables load* permite cargar las reglas (en *Debian* es necesario indicar el nombre del fichero de reglas o usar *iptables-restore*). Hay que señalar que *Debian* por defecto soporta los ficheros *active* para reglas normales e *inactive* para las que quedan tras la desactivación del servicio.

5. Herramientas de análisis y seguridad adicionales

Debemos tener en cuenta ante todo que muchas herramientas de seguridad pueden utilizarse para realizar ataques y por lo tanto debemos probarlas sobre máquinas de nuestra red local o privada, de otro modo, podría tener consecuencias y se nos podrían pedir responsabilidades. Algunas de estas posibles herramientas que han podido ser mencionadas en puntos anteriores son:

a. *TripWire*: Herramienta que mantiene una base de datos de sumas de comprobación para los ficheros importantes del sistema. Puede servir como *IDS* preventivo. Básicamente permite tomar una foto del sistema y poder comparar cualquier modificación que ocurra en él para asegurar que no se haya visto comprometido. Es importante elegir un subconjunto importante de archivos del sistema o que sean directamente una posible fuente de ataque.
b. *Nmap*: Como ya hemos mencionado previamente en este documento se trata de una herramienta de escaneo de puertos para redes, permitiendo diferentes tipos de escaneos con base en las restricciones del sistema. Puede utilizar distintos paquetes de *TCP* y *UDP* para realizar comprobaciones sobre las conexiones.
c. *Wireshark*: Se trata de un analizador de protocolos que actúa como un husmeador y tiene como objetivo principal capturar el tráfico de la red. Ofrece la posibilidad de ver detalles de los paquetes individuales, agruparlos y hasta reconstruir el tráfico de una sesión completa de un protocolo de tipo *TCP*.
d. *Snort*: Sistema *IDS* para llevar a cabo un análisis del tráfico en tiempo real, guardando registros de los mensajes. Concretamente, analiza el tráfico de red de un modo que le permite detectar patrones de un ataque. El sistema usa reglas para anotar la situación - *log* - de modo que puede producir un aviso - *alert* - o descartar la información - *drop*.
e. *Nessus*: Detector de vulnerabilidades conocidas que, gracias a pruebas de diferentes técnicas de intrusión, es capaz de asesorar acerca de cómo se puede mejorar la seguridad para aquellas que son detectadas. Se basa en una arquitectura cliente/servidor, el cliente muestra los resultados y el servidor lleva a cabo las diferentes comprobaciones sobre las máquinas. Su

capacidad llega hasta incluso poder analizar una red entera.

Lista adicional de órdenes a tener en cuenta:

Por último, dentro de este punto es conveniente destacar algunos comandos útiles para configurar mejor la seguridad de nuestro *Debian*. Por ejemplo si ejecutamos como *root* el comando `nmap -sTU -0 localhost` podemos detectar los servicios abiertos (para la obtención de servicios activos a través de la búsqueda de puertos activos a la escucha podemos usar `netstat -lut`). Si aplicamos *nmap* con el nombre *DNS* o *IP* de la máquina podríamos observar lo que se ve desde el exterior. Desde */etc/inted.conf* se nos permite desactivar los servicios, y, para que se vuelva a leer la configuración que hemos cambiado hay que reiniciar *inetd* con */etc/inet.d/inetd restart*. Para mirar a que se dedica un puerto concreto podemos consultar */etc/services*. Por otro lado el comando *netstat* presenta estadísticas del sistema de red, incluyendo paquetes enviados y recibidos además de conexiones abiertas y quién las usa entre otras. Estos comandos pueden ser de gran importancia a la hora de otorgar una capa de seguridad a nuestro sistema.

6. Principios de buenas prácticas

6.1. Monitorización de los registros

Una costumbre de primer orden que todo administrador debe incorporar a su *'receta'* de buenas prácticas es el análisis de los registros -ficheros de log- del sistema. Una revisión periódica de estos archivos permite al sysadmin tener una idea global y precisa del estado general de los equipos y las conexiones, los cambios que han tenido lugar, detectar sucesos de interés para la continuidad del sistema y, llegado el caso, definir si es necesario actuar frente a intrusiones o accesos no autorizados, si el atacante no ha sido lo suficientemente 'precavido' como para borrar las huellas a su paso. La mayor parte de los logs se encuentran en el directorio /var/log.

La mayoría de los servicios se apoyan en las funcionalidades proporcionadas por otro demonio del sistema, Syslogd, para llevar a cabo el registro de sus actividades en el Sistema Operativo. Este Daemon se inicia automáticamente con el arranque del propio S.O. y es el encargado de realizar y guardar informes sobre el funcionamiento general de la máquina. Su labor central gira en torno a la recepción de mensajes procedentes de diferentes partes del sistema perfectamente identificadas (núcleo, programas, servicios, etc), y conducirlos a diferentes localizaciones, según se defina en el fichero de configuración /etc/syslog.conf, donde se especifica todo este conjunto de reglas de direccionamiento. Estas reglas se estructuran por niveles de mensajes, según su grado de importancia: debug, info, err, notice, warning, crit, alert y emerg, de menor a mayor. Por norma general, la mayor parte de los mensajes generados se reenvían al fichero /var/log/messages, pero puede configurarse para cambiar la ubicación de destino.

Otra labor en el trabajo de un administrador consiste en llevar un control del tamaño de los ficheros de log, pues recogen continuamente nueva información, por lo que su tamaño aumenta por momentos. En Debian, este control se realiza por medio de otro Daemon, "logrotated", que realiza copias periódicas de los logs y las comprime. De no hacerlo, el almacenamiento del sistema podría colapsar de un momento a otro. Su funcionalidad puede configurarse en el fichero /etc/logrotate.conf.

De entre todos los registros a los que el administrador debe prestar especial atención -y también proteger de miradas curiosas- cabe destacar los siguientes:

a. /var/log/messages: contiene la mayor parte de los mensajes generados por todo aquello que reside en el sistema (programas, servicios, núcleo, etc), información del

arranque, y más información procedente de diversas fuentes. Es el fichero utilizado, como se ha visto, por el demonio Syslogd. Revisar este fichero en caso de intrusión en las fechas donde se crea que se ha producido el ataque puede incluso ayudar a descifrar el modus-operandi seguido por el invasor.

b. /var/log/utmp, /var/log/wtmp, /var/log/btmp: ya hemos hablado de ellos con anterioridad. Contienen información acerca de las sesiones que han tenido y tienen lugar en el momento actual del sistema. El primero de ellos revelará información sobre todos los usuarios que se encuentran en el sistema y es el utilizado por el comando 'who' para desempeñar su labor. Por su parte, wtmp registra la fecha y fuente de todas las sesiones abiertas en el sistema y el fin de las mismas, y permite realizar un análisis pormenorizado por usuarios, siendo el origen de la funcionalidad del comando last. Una variante de este comando es lastb, que hace uso del log /var/log/btmp y registra todos los intentos de acceso fallidos.
c. /var/log/dmesg: una vez que el sistema arranca, se encarga de registrar todos los errores detectados por el kernel a nivel de hardware. Puede ser útil de cara a una inspección del sistema en la capa física.
d. /var/log/auth.log: todos los eventos relacionados con la autenticación se registran en este fichero. Resulta útil para revisar las autorizaciones de acceso concedidas a los usuarios. Asimismo, permite desentrañar ataques de fuerza bruta realizados contra el sistema.

Resulta obvio, llegado este punto, pero conviene recordar al administrador que debe tomar en consideración que liberar la escritura del directorio /var/log supone un riesgo de seguridad enorme, y que sólo debería ser escrito por el root y, como es lógico, los demonios del sistema asociados y conocidos.

6.2. Realización periódica de copias de seguridad

La segunda regla de oro para el mantenimiento de un sistema y, en especial, de cara a asegurar su supervivencia, es la realización de copias de seguridad, tanto en cuanto del estado general de la máquina como de la información sensible, de forma periódica y fijando unas pautas y momentos que causen el mínimo impacto en los usuarios y en el propio sistema, realizándose de forma transparente a su funcionamiento normal. Algunos puntos a tener en cuenta en este asunto son los siguientes:

1. Realización de copias de seguridad completas del estado de la máquina: el administrador decidirá el cómo y el cuándo, si bien se recomienda realizar copias de respaldo periódicas del estado de la máquina en unidades físicas protegidas contra escritura y fuera de línea.
2. Elaboración de un calendario de realización de copias adecuado: el administrador debe tener previsto cuándo se realizarán estas copias, ya sean completas o incrementales, de tal manera que el sistema de respaldo sea manejable y mantenible. Una buena '*receta'* puede consistir en realizar una copia incremental cada día de la semana de lunes a viernes, y realizar una copia completa cada quincena, en el viernes correspondiente, para mantener un equilibrio entre tiempo invertido en el guardado y el tiempo que habría de invertir en una posible restauración. Tenga en cuenta que las copias incrementales son más ligeras de crear, pero más complejas de restaurar, pues deben recuperarse todas ellas desde la última completa; caso opuesto es el de las copias integrales, que se demoran más en su proceso de creación, pero su restauración se limita a recuperar una única copia. Puede valorarse la necesidad de realizar una copia completa fuera de orden si se han realizado

cambios en el sistema que se consideran críticos.

3. Realización de pruebas sobre las 'backup': conviene que el administrador realice cuando estime oportuno una batería de test necesarios para conocer si las copias de seguridad que está almacenando se comportan como deben: pruebas de restablecimiento de ficheros concretos o carga de copias antiguas en sistemas de prueba pueden ser algunos ejemplos.

6.3. 'Diccionario' de buenas prácticas:

Sirva el siguiente apartado como una breve guía acerca de los pasos más elementales que el administrador debe seguir -y en lo posible, recordar- en todo momento de su relación con el sistema de la empresa. Esta serie de buenas prácticas le ayudarán a mantener la seguridad efectiva e integral del propio sistema -o, por lo menos, a poner algún candado más que abrir a sus invasores- y le proporcionarán una metodología de actuación para enfrentarse a nuevas vulnerabilidades que pudieran aparecer en él:

a. Cifre las comunicaciones con el servidor: toda la información transmitida a través de la red es susceptible de ser leída. Emplee servicios como rsync o sftp para la transferencia segura de ficheros entre máquinas.
b. Evite el uso de ftp, telnet y las 'r-utilities': el uso de estos complementos no impide a un atacante capturar credenciales de acceso de forma sencilla. Otros servicios como OpenSSH o SFTP ayudan a paliar el compendio de problemas de seguridad derivados del uso de estas herramientas.
c. Cuanto menos software, menos vulnerabilidad: conviene llevar un registro de los programas instalados en el sistema y deshacerse de aquellos que hayan dejado de tener utilidad.
d. Distribuya sus servicios en diferentes servidores, si es posible: ver comprometida una máquina donde se aloja un solo servicio siempre es preferible a ver comprometido un servidor central completo.
e. Mantenga actualizado el kernel y el software: conviene, no obstante, tener un equilibrio, utilizando siempre las versiones más estables y actualizadas posibles.
f. Emplee servicios de seguridad adicionales, como SELinux o, en su defecto, AppArmor.
g. Elabore una política de contraseñas suficientemente segura, evitando las contraseñas vacías o integradas por palabras de diccionario comunes, que puedan romperse por medio de programas de fuerza bruta como 'John The Ripper', que también debe ejecutar contra su propio sistema de forma responsable para detectar vulnerabilidades. Definir períodos de actualización de contraseñas a sus usuarios y evitar la repetición de contraseñas antiguas son buenos consejos a seguir en esta dirección.
h. Bloquee las cuentas de los usuarios que hayan recibido un número suficiente de intentos de login erróneos.
i. Conserve las contraseñas de los usuarios en un lugar seguro; se recomienda encarecidamente el uso del fichero de sombras que encontrará en la ruta /etc/shadow, únicamente legible para el propio usuario root. Cifre las contraseñas de sus usuarios con protocolos de encriptación que proporcionen un nivel de seguridad aceptable, como md5.
j. Asegúrese que ningún usuario del sistema, a excepción del priopio root, tenga asignado el UID 0, que le otorgaría permisos para hacer y deshacer a discreción sobre el sistema.
k. El administrador nunca debería loguearse directamente como root, sino con una cuenta de usuario normal y ascender a administrador únicamente cuando sea imprescindible para el bienestar del sistema.

Recuerde: *"Superman no puede llevar siempre la capa"*.

l. Asegure físicamente su servidor, mediante las configuraciones oportunas de la BIOS y el Bootloader..
m. Deshabilite los servicios y demonios desconocidos e innecesarios. Apóyese para ello en la utilidad '*systemctl'*.
n. Monitorice los puertos de escucha. Utilizadas convenientemente, herramientas como *nmap* pueden ser útiles en este sentido.
o. Evite el uso de entornos gráficos (como el entorno de ventanas X) si es preciso en sus servidores. En general, los servidores no demandan este tipo de entornos, y deshabilitarlos incrementa el rendimiento y la seguridad del sistema.
p. Configure convenientemente su cortafuegos por medio de Iptables. Un cortafuegos ayuda en la misión de filtrado de paquetes y tráfico de red.
q. Configure convenientemente las particiones. Incluya en el fichero /etc/fstab las opciones noexec, nodev y nosuid para abolir permisos innecesarios en particiones que no los requieren.
r. Fije cuotas a sus usuarios en el acceso a los recursos del sistema.
s. Realice una exploración de los ficheros que tienen el bit 's' levantado, especialmente si ese bit otorga a determinados usuarios ejecutar binarios con los privilegios del usuario root, ya sea a nivel de usuario (SUID) o de grupo (SGID).
t. Evite en la medida de lo posible el uso de ficheros con escritura global, así como los ficheros sin propietario.
u. Emplee sistemas de autenticación robustos en sus aplicaciones, por medio de técnicas certificadas como los módulos PAM. El uso de OpenLDAP es asimismo sumamente recomendable en la comunicación entre clientes y servidores.
v. Revise periódicamente la información almacenada en los logs del sistema. Acuda a la ruta */var/log* para lo propio.
w. En caso de necesidad de procesos de login remoto, emplee sistemas certificados como OpenSSH.
x. Emplee, en la medida de las posibilidades, herramientas avanzadas de monitorización del sistema de ficheros. Tripwire es la más recomendable.
y. Proteja la supervivencia de su sistema frente a cualquier tipo de invasión o pérdida de datos realizando copias de seguridad periódicas y bien organizadas.
z. Instruya a los usuarios del sistema en la aplicación de prácticas saludables en el manejo del material de oficina, tanto software como hardware: el primer paso para garantizar la seguridad del sistema es la precaución de quienes lo hacen funcionar.

7. El sistema ha sido comprometido: política de intervención

Recuperando una de las ideas iniciales de este ensayo, conviene recordar que el único sistema plenamente seguro es aquel que se "oculta del mundo y se encierra bajo llave". Por lo tanto, un sistema, por muy bien que se defina y ejecute su política, siempre estará expuesto a riesgos de seguridad nuevos y conocidos. Tan importante es definir un protocolo que nos diga cómo debemos prevenir la 'enfermedad' como establecer una serie de pautas de intervención que nos digan cómo podemos erradicarla, una vez que enemigo está dentro de nuestro sistema.

Si el atentado de seguridad se produce a nivel físico de la manipulación de las máquinas o incluso robo de material de oficina, lo más recomendable es iniciar una investigación de la mano de las autoridades locales. En caso de que el ataque se produzca a nivel de la red y del software, existen una serie de medidas que el

administrador debe tomar ipso-facto para interceder sobre las acciones del atacante:

En primer lugar, si el ataque se produce a escala local, esto es, un usuario de la propia empresa está intentando comprometer su propia seguridad, lo más recomendable es actuar por medios presenciales (a través de reuniones o por vía telefónica), a fin de encontrar el origen de la amenaza y valorar si es procedente el despido del propio trabajador. A partir de aquí, la política de actuación dependerá de las circunstancias de la intrusión, intenciones de la persona y de la propia reglamentación de la empresa en particular.

En caso de detectar señales de ataque a través de la red, lo primero que debe hacerse es detectar el medio de transmisión implicado y cortar esa vía de comunicación: si la conexión es por módem, desconecte el modem; si llega a través de Ethernet, desconecte el cable Ethernet, etc. Como siguiente medida a realizar, y especialmente si no dispone de medios para desconectar sus equipos de la red, es la aplicación de reglas de filtrado que corten la conexión de los atacantes (tcp_wrappers, ipfwadm, …). Finalmente, se debe bloquear la cuenta del usuario infectado o sospechoso, haciendo nuevamente especial hincapié en la ausencia de servicios como .rhosts, el acceso al FTP, y ficheros con el bit 's' habilitado, fuente común de puertas traseras (*backdoors*).

Una vez completado este protocolo de actuación, el administrador debe monitorizar por completo el sistema (revisión de los *logs* y empleo de las herramientas de seguridad avanzadas que se han ido comentando), comprobando si existen nuevos intentos de acceso desde otras cuentas e incluso ubicaciones de red diferentes. Si, con suerte, hemos conseguido sacar al atacante del sistema, el trabajo no ha concluido. Es el momento de cerrar la brecha de seguridad para que el ataque no vuelva a suceder. El sistema se ha recuperado de la agresión, es el momento de fortalecerse. Lo primero es tratar de determinar en lo posible qué medios utilizó el atacante en su proceso de invasión. Como ya se ha dicho en numerosas ocasiones, revisar los *logs* del sistema en busca de huellas del ataque es una medida recomendable. De la misma manera, debemos comprobar qué archivos o utilidades han sido infectados, para lo cual ha de comprobarse la suma de verificación de los diferentes servicios, procesos y ficheros mediante las utilidades oportunas (ya presentadas en los capítulos previos). Si los cambios provocados en el sistema se consideran suficientes o críticos, debemos restaurar el sistema de forma íntegra mediante las copias de seguridad de que se disponga en el momento del ataque. Se recomienda evitar la restauración de archivos desde el propio sistema comprometido, a fin de evitar la acción de *troyanos* no detectados que puedan infectar de nuevo los ficheros recuperados. El administrador también debe preguntarse cuánto tiempo puede llevar el sistema infectado (detectar una infección no implica necesariamente que el sistema haya sido comprometido por primera vez en ese momento) y actuar en consecuencia. Este factor determinará, por ejemplo, la copia de seguridad que habremos de utilizar para realizar la restauración del sistema, pues quizá dispongamos incluso de copias infectadas.

El paso final consistirá en iniciar una investigación de la mano de las autoridades competentes. Es preciso indicar que estas investigaciones solo se emprenderán si existen motivos justificados para ello, por lo que el administrador deberá facilitar a estas autoridades toda información que considere relevante sobre el ataque recibido, generalmente almacenada en los propios *logs* del anfitrión infectado. Con todo, Debian cuenta además con una página de referencia que el administrador debe consultar en su lucha contra las vulnerabilidades de su sistema: http://www.debian.org/security/. Puede considerarse la 'Biblia' en cuanto a la seguridad de la distribución se refiere, por lo que debe ser

una fuente de consulta permanente, sobre todo en materia de actualizaciones, bugs detectados, parches de seguridad publicados, etc

8. Bibliografía y referencias

[1] "Linux Security HOWTO" | Kevin Fenzi, Dave Wreski (22 de enero de 2004).

[2] "Administración de Seguridad" | Josep Jorba Esteve [UOC].

[3] 40 Linux Server Hardening Security Tips [2019 Edition] - nixCraft
https://www.cyberciti.biz/tips/linux-security.html

[4] Debian Handbook
https://debian-handbook.info/browse/stable/

[5] Introduction to Linux security principles - PenguinTutor
http://www.penguintutor.com/linux/introduction-linux-security

[6] «Los siete virus informáticos más dañinos de la historia» - La Vanguardia
https://www.lavanguardia.com/tecnologia/20170521/422734535859/virus-informaticos-mas-daninos-historia.html