[VOLUMEN 22 · NOVIEMBRE 2000]
Ágora SIC Divulgación

SUMARIO

Técnicas de detección de *sniffers*

Julio César Hernández
José María Sierra
Arturo Ribagorda
Benjamín Ramos
Laboratorio de Seguridad en TIC
Departamento de Informática
UNIVERSIDAD CARLOS III DE MADRID

La detección de *sniffers* es una de las múltiples tareas, y una de las más desconocidas, que todos los administradores de seguridad tienen que realizar para garantizar que la seguridad de sus redes no se vea comprometida. Si bien hay un buen número de herramientas que facilitan esta tarea ([1],[2],[3]), es importante conocer en profundidad cómo funcionan para poder interpretar y relativizar sus resultados, ya que estos programas tienen un buen número de limitaciones y pueden ser engañados con facilidad para producir falsos positivos y falsos negativos[4]. En este trabajo, partiendo de las definiciones más básicas, se muestran algunas de las técnicas que componen el estado del arte en esta área.

Técnicas de detección de sniffers

¿QUÉ ES UN SNIFFER?

Un sniffer es un dispositivo que permite a cualquier elemento conectado a una red tener acceso al tráfico que no va destinado expresamente a él o a broadcast; tráfico, por tanto, al que no debería tener acceso.

Aunque bajo esta amplia definición cabe una gran multitud de diferentes tipos de sniffers, el más habitual contra el que un administrador de seguridad va a tener que luchar es simplemente un PC con la tarjeta de red en modo promiscuo. Esta sugerente y atractiva denominación hace referencia a una tarjeta de red que, en contra de lo que debe ser su comportamiento normal, no descarta a bajo nivel los paquetes que no van dirigidos a ella o a la dirección de broadcast y, por el contrario, pasa todos los paquetes que ve en la red a capas superiores, generalmente el kernel del SO, para su posterior procesamiento.

¿POR QUÉ SON PELIGROSOS?

Evidentemente, sin más que releer el párrafo anterior, resulta bastante obvio que un sniffer supone una amenaza grave para la seguridad no sólo de una máquina sino también de toda una red. El hecho de que permita acceder a información que circula por la red, a la que de otro modo no tendría acceso, supone un riesgo importante. Entre esta información se encuentran contraseñas, correo confidencial, números de tarjetas de crédito, registros de bases de datos, cookies con información de autenticación, etc...

Lamentablemente, una gran cantidad de tráfico confidencial viaja en claro, sin ningún tipo de cifrado, por las redes de la mayoría de las empresas. Ese es el entorno ideal para un sniffer, en que puede acceder de forma transparente a esa información, y permitir que alguien abuse de su conocimiento.

Además del riesgo que suponen en sí mismos, los sniffers son a menudo un síntoma de que la seguridad de una máquina o una red ha sido violada. Por eso es muy importante realizar búsquedas periódicas de sniffers dentro de las redes de cualquier empresa, no sólo por el daño que puedan causar, sino también porque encontrarlos es señal de que se ha producido y explotado una grave brecha y de que hay que tomar medidas inmediatas.

¿SON SIEMPRE PELIGROSOS?

Sí, aunque hay algunas excepciones. A veces, explorando una red a la busca de sniffers se detectará que hay algunos, por ejemplo, en máquinas que dependen del departamento de administración de redes. Esto puede ocurrir porque, en realidad, un sniffer no se diferencia demasiado de una herramienta de monitorización y diagnosis del tráfico de red que puede estar siendo legítimamente utilizada por personal encargado de la administración de la red.

Si bien este uso es en principio completamente normal, hay que vigilar que no conduzca al abuso, ya que ese departamento podría obtener grandes beneficios del acceso privilegiado a la información que circula por la red.

Otros dispositivos, especialmente routers y hubs, suelen producir falsos positivos que hay que tener en cuenta.

¿CÓMO SE BUSCAN?

Ni que decir tiene que hay diferentes aproximaciones al problema de cómo detectar un sniffer, y que éstas varían según se tenga acceso local a la máquina, o bien haya que descubrirlos desde alguna máquina remota. Esta última variante es, con mucho, la más compleja pero también la más usual.

El objetivo que la mayoría de pruebas tratan de conseguir es que la máquina que tiene la tarjeta de red en modo promiscuo se traicione a sí misma, revelando que ha tenido acceso a información que no iba dirigida a ella y que, por tanto, tiene un sniffer. Éste es un objetivo ambicioso y complejo que, lamentablemente, puede ser imposible.

LÍMITES TEÓRICOS

A veces resulta completamente imposible detectar un sniffer. Por ejemplo, si el sniffer ha sido diseñado exclusivamente para esta tarea (generalmente dispositivos hardware), entonces no devolverá jamás un paquete, no establecerá nunca una comunicación, sino que permanecerá siempre en silencio y su detección remota será, simplemente, imposible.

La detección de este tipo de sniffers sólo puede hacerse por inspección directa de los dispositivos conectados a la red. Afortunadamente, es muy probable que muchos de los administradores de seguridad jamás se enfrenten a este tipo de dispositivos. La inmensa mayoría de los sniffers son simplemente PCs con software malicioso que cambia una tarjeta de red a modo promiscuo, que tienen una presencia activa en la red (devuelven paquetes) y, por tanto, pueden ser engañados para autoinculparse.

TÉCNICAS LOCALES DE DETECCIÓN

Aunque no se trata de una tarea trivial, ésta es, con mucho, la situación en que resulta más sencillo localizar un sniffer. Normalmente basta con revisar la lista de programas en ejecución para detectar alguna anomalía (CTRL+ALT+SUPR o ps-aux¦more).

Otro buen sitio donde mirar es en la lista de los programas que se inician automáticamente al encender el PC (archivos /etc/rc.d/rcX.d/* o .bashrc, etc... en un sistema Unix y autoexec.bat o ciertas claves del Registry en una máquina Windows) o las tareas programadas (cron, at). Es importante destacar que los ejemplos anteriores son sólo triviales y no pretenden ser una enumeración exhaustiva. Hay decenas de otras posibilidades, algunas muy ingeniosas y nada elementales.

Cualquier novedad o anomalía debe ser investigada en profundidad porque podría revelar no sólo un sniffer en funcionamiento sino también otros programas que supongan una grave amenaza (virus, troyanos, gusanos, etc...).

Pero no siempre esta simple inspección de los programas en ejecución, los preparados para ejecutarse automáticamente en el inicio o las tareas programadas, va a tener éxito. De hecho, muchos troyanos relativamente sofisticados resultan transparentes a esta búsqueda sencilla [5]. Son muchas más las técnicas de ocultación posibles, y no es difícil para un programador con cierto talento desarrollar aplicaciones que se escondan bajo otras cuyo uso es perfectamente normal y legítimo.

Afortunadamente hay otras formas de obtener información sobre el estado de la tarjeta de red.

HERRAMIENTAS DE DETECCIÓN LOCAL

En una máquina con alguno de los sistemas operativos de la familia Unix se dispone de una utilidad que resulta especialmente valiosa en la lucha contra los sniffers. Se trata de ifconfig, comando que informa del estado de todas las interfaces de red del sistema e indica si alguna de ellas se encuentra en modo promiscuo.

Es muy recomendable verificar periódicamente el estado de las interfaces con el comando ifconfig, y una buena forma de hacerlo es añadiendo esta tarea al crontab. En caso de descubrir que una de las interfaces ha cambiado a modo promiscuo, podría enviarse un mensaje de alerta y deshabilitar inmediatamente esa interfaz con ifconfig down.

La alternativa es el programa CPM [6], siglas de Check Promiscuous Mode, que realiza una tarea semejante mediante el uso de llamadas ioctl. Pueden conseguirse gratuitamente versiones para cualquier sistema Unix en multitud de lugares relacionados con la seguridad, entre otros el CERT.

Evidentemente, esta metodología de detección local de sniffers depende del buen funcionamiento del comando ifconfig o del programa CPM. Lamentablemente, esto es algo que no se puede dar por sentado. Es bastante habitual que, si no se detecta una violación de seguridad en los sistemas en una fase temprana, el atacante disponga del tiempo suficiente como para borrar completamente sus huellas y obtener un control total de todos los recursos de forma totalmente transparente.

Entre otras cosas, podría sustituirse el ejecutable /sbin/ifconfig (simplemente se cambia el código fuente, se compila y se sustituye) por otro que no informase de que la tarjeta de red está en modo promiscuo, invalidando así los métodos de detección. Lo mismo puede ocurrir con el programa CPM: su código fuente está disponible para diferentes plataformas y es un juego de niños cambiarlo para hacer que nunca, en ningún caso, informe de que la tarjeta de red está en modo promiscuo.

Cambios más sutiles son también posibles, como por ejemplo sustituir el comando o programa instantes antes de su ejecución programada, para volverlo a cambiar instantes después. Pero éste es un problema general que aparece numerosas veces en el mundo de la seguridad y que tiene una solución sencilla: utilizar mecanismos de control de integridad para asegurar que los programas en ejecución son los originales y que tienen un comportamiento fiable. Para eso se dispone de las funciones resumen como MD5, SHA1 o RIPEMD, que pueden garantizar la integridad de todos los ficheros importantes del sistema.

Existe un buen número de herramientas gratuitas y altamente configurables basadas en estas primitivas criptográficas que permiten ser programadas para realizar estos estudios de integridad periódicamente ([7],[8],[9]). Son fáciles de obtener, código fuente incluido, en Internet.

Resta mencionar un par de cuestiones para finalizar este apartado: la primera es la importancia de verificar, especialmente a la hora de seleccionar una de estas herramientas de control de integridad, que ésta se autochequee, que verifique su propia integridad. Las razones de esta necesidad son evidentes. Además, los chequeos deben basarse en primitivas criptográficas y no simplemente en CRCs y comprobaciones triviales tales como el tamaño, el uid, la fecha de creación o de modificación de ficheros, etc..., porque todas estas características son fácilmente manipulables por un atacante que haya obtenido privilegios de root. Es importante tener en cuenta estas cuestiones a la hora de escoger una herramienta de control de integridad puesto que abundan las no recomendables [10].

La segunda es que, obviamente, estos controles de integridad deben ser llevados a cabo de manera regular y de forma no fácilmente predecible, porque si no podrían ser burlados con suma facilidad. Aunque estas primitivas criptográficas son excepcionalmente rápidas, el gran volumen de ficheros que normalmente tienen que procesar implica que el control de integridad puede requerir bastante tiempo. Por tanto, en muchos casos, no puede programarse de forma totalmente aleatoria porque podría interferir con otras tareas críticas en el sistema. No obstante, y dentro de los márgenes que las actividades de los sistemas permitan, debe intentarse que el momento concreto de actuación de estas herramientas de control de integridad sea impredecible.

DETECCIÓN REMOTA DESDE EL MISMO SEGMENTO DE RED

Es en este entorno donde más frecuentemente el administrador de seguridad tiene que realizar su investigación. Existe un cierto número de técnicas heurísticas que son de utilidad y que se presentan a continuación, pero hay que tener claro que estas técnicas tienen bastantes limitaciones y que no resulta en absoluto improbable que exista un sniffer en la red y que no sea detectado (falso negativo) o que máquinas o usuarios completamente inocentes sean detectados como sniffers (falsos positivos). Ninguna de estas técnicas tiene un balance óptimo entre estos dos riesgos, y, por tanto, la recomendación más sensata es, quizá, usarlas todas y conocerlas en profundidad para ser capaces de interpretar y relativizar sus resultados.

Por su ámbito de aplicación, estas técnicas se pueden dividir en dos grupos: las dependientes del sistema operativo y las que no lo son:

Dependientes del sistema operativo

Como su propio nombre indica, estas técnicas usan algún fallo o característica propia de determinados sistemas operativos (o parte de ellos, como el subsistema TCP/IP) para reconocer a una tarjeta de red en modo promiscuo. La ventaja que tienen es su excelente rendimiento cuando se explora máquinas que tienen justamente la versión del sistema operativo del que la técnica obtiene partido. La desventaja fundamental es el gran número de falsos negativos que ocasiona debido a que en muchos casos las implementaciones de la pila TCP/IP varían entre versiones del mismo sistema operativo y, por tanto, estas técnicas podrían resultar inútiles incluso con versiones posteriores del sistema operativo al que van dirigido, además de, por supuesto, con todos los demás.

A continución se exponen algunos ejemplos:

– Filtrado de paquetes en kernels Linux antiguos:

Algunos kernels linux antiguos tienen una característica curiosa que hace que sea relativamente sencillo inducirles a revelar si la tarjeta de red está en modo promiscuo o no.

En condiciones normales, los paquetes son aceptados o rechazados a nivel hardware por la tarjeta de red según la MAC address de destino que aparezca en el frame Ethernet. Sólo si esa MAC address es la de la propia máquina o la de broadcast, el paquete es aceptado (copiado) y procesado (se pasa al kernel); en caso contrario, se rechaza (se ignora).

Pero, cuando la tarjeta de red está en modo promiscuo, las cosas cambian drásticamente. Todos los paquetes son aceptados y procesados, por lo que todos pasan directamente al kernel del sistema operativo. Es ahora éste el que debe encargarse de filtrarlos y decidir cuáles van a ser procesados por otras aplicaciones y cuáles van a ser descartados.

Pues bien, ese filtro a nivel de sistema operativo puede realizarse de varias formas. La más adecuada, por coherencia

con la acción a nivel físico, es volver a inspeccionar la MAC de destino, aunque también se puede hacer a nivel de IP. El frame encapsula un paquete IP con la dirección IP de destino (la de la maquina o la broadcast de la red) y esta dirección IP puede ser utilizada para decidir si el paquete va dirigido a la máquina o no porque, en condiciones normales, coincide con la IP correspondiente a la MAC address. Se puede sacar partido de este hecho de la siguiente forma:

Para cada PC del segmento de red que se desee analizar se crea un paquete con una MAC address de destino que no exista en el segmento, y que encapsule un paquete del tipo ICMP echo reply (ping) con dirección destino la IP de la máquina a analizar y origen del PC. Cualquier máquina de la subred con la tarjeta en modo no promiscuo rechazará directamente un paquete que tiene como MAC de destino una que no es la suya ni la de broadcast, y, por tanto, no procesará el paquete. Cualquiera con la tarjeta en modo promiscuo pasará el paquete al kernel y, si se trata de uno de los antiguos kernels linux afectados, éste analizará el paquete exclusivamente según los datos del paquete IP que encapsule. Desde ese punto de vista, el paquete es un ping completamente normal y, como tal, es contestado por la máquina que tiene el sniffer, revelando así su estado.

Este peculiar modo de filtrado de paquetes, tan útil para localizar sniffers, también ha sido durante algún tiempo una característica del sistema operativo NetBSD.

– Filtrado de paquetes broadcast en algunos drivers de plataformas Windows:

La idea es la misma que en el ejemplo anterior, utilizar un fallo o característica de filtrado de paquetes para distinguir cuándo una tarjeta de red está en modo promiscuo.

La característica a considerar en este caso es cómo el driver del sistema operativo decide cuándo un paquete ethernet va dirigido a la dirección de broadcast ff:ff:ff:ff:ff:ff. Esta parece una tarea relativamente sencilla: cabe esperar que simplemente verifique si los 6 octetos tienen el valor 0xff, sin más. Pero no es este el caso: cuando la tarjeta de red está en modo no promiscuo, sí se verifican los 6 octetos, mientras que cuando ésta se encuentra en estado promiscuo sólo se verifica el primero de ellos.

El porqué de este extraño comportamiento no es evidente a primera vista, aunque podría tratar de justificarse como un intento de mejorar la eficiencia del proceso de paquetes justo cuando (al estar en modo promiscuo) se espera una avalancha de ellos. En cualquier caso, facilita mucho la detección de sniffers, el procedimiento a seguir no puede ser mas simple:

Si se crea un paquete dirigido a la MAC address ff:00:00:00:00:00, cualquier tarjeta en modo no promiscuo lo va a rechazar automáticamente porque esa dirección no coincide con la suya ni con la de broadcast. Sin embargo, una tarjeta en modo promiscuo con un sistema operativo Windows que use el driver afectado confundirá ese paquete con uno legítimo dirigido a broadcast y lo procesará adecuadamente. El resto es bien conocido: se puede encapsular, por ejemplo, un paquete ICMP echo reply en el frame ethernet y así conseguir que la máquina con el sniffer se traicione y devuelva un ping de respuesta informando de su estado.

Por supuesto, estos métodos pueden ser burlados con relativa facilidad, pero en muchos casos resultan útiles.

No dependientes del sistema operativo

En general son menos fiables y menos concluyentes. Suelen basarse en suposiciones sobre el comportamiento de determinados sniffers, que pueden no darse en casos concretos, convirtiendo alguna de estas técnicas en completamente inútiles en determinados ambientes. Otras son más generales, pero poco resolutivas, porque no clasifican sino que simplemente dan indicios que en muchos casos no son suficientes. No suelen proporcionar muchos falsos positivos, aunque pueden ser burladas y utilizadas para inculpar a terceras personas. Tampoco falsos negativos, aunque la ultima generación de sniffers ya incorpora técnicas de evasión bastante sofisticadas que evita su detección.

Véanse algunos ejemplos:

– Tests DNS:

Los tests DNS basan su eficacia en la suposición de que el sniffer, para resultar mas útil a su usuario, transforma las direcciones IP que ve en la red en sus correspondientes nombres, generalmente mucho mas informativos. Obviamente, nombres como firewall.empresa.com y bbdd.empresa.com son mucho más informativos que sus correspondientes IPs, y es natural que un atacante prefiera disponer de ellos y que, por tanto, utilice un sniffer que le facilite esta información.

Pues bien, si ese es el caso, hay una técnica muy sencilla que puede facilitar el descubrimiento de sniffers dentro del segmento de red. La idea es engañar al sniffer con tráfico falso que contenga una IP nueva, induciéndole así a realizar una petición DNS para resolver esa IP, revelando de esta forma que ha tenido acceso a tráfico que no va destinado a él. Como siempre, el objetivo fundamental es convertir una máquina que hasta el momento puede haber sido completamente pasiva en un elemento activo.

Por tanto, hay que crear un paquete IP que vaya destinado a alguna máquina ficticia (que no exista en el segmento de red y que no haya sido contactada por ninguna de las máquinas sospechosas), por ejemplo a la IP 1.1.1.1. Cualquiera de las máquinas del segmento de red que no estén en modo promiscuo, simplemente ignorarán el paquete porque no va dirigido a ellas. Sin embargo, un sniffer activo que use esta característica de resolución DNS comprobará en su caché local que no tiene una entrada para la IP 1.1.1.1, e inmediatamente hará una petición DNS para resolver esta IP. Aunque hay muchas otras posibilidades, un método de detección de estas peticiones DNS es, por ejemplo, instalar en la máquina desde la que se realiza la búsqueda un sniffer que analice todo el tráfico del segmento de red a la búsqueda de peticiones de resolución de la IP ficticia. La máquina que haya originado esa petición es la que tiene instalado el sniffer.

Con el método citado han de tenerse algunas precauciones porque, aunque no se ha hecho hasta ahora, un sniffer podría ser diseñado para hacer consultas DNS e implicar a otra máquina de su segmento de red. De esta forma, con esta técnica basada en tests DNS sería imposible de descubrir y, lo que es peor, podría implicar a un tercero completamente inocente.

¿Cómo? Muy sencillo, basta con generar los paquetes de petición de resolución de DNS, haciendo IP spoofing de otra máquina en el mismo segmento de red. Los paquetes serían respondidos por el servidor DNS y descartados por la víctima, pero, al estar el sniffer en el mismo segmento de red que ésta, podría acceder sin dificultad a su contenido e informar así a su usuario evitando ser detectado.

Y, por supuesto, para que esta técnica funcione es imprescindible que el sniffer haga peticiones de resolución de nombres de dominio. En otro caso, aunque interesante, resulta completamente inútil.

– Tests de latencia:

La idea fundamental detrás de estos tests es la más general que se puede aplicar en la búsqueda de sniffers. De hecho, es la única suficientemente general como para poder ser aplicada fuera del restringido ámbito del propio segmento de red.

Como se ha comentado antes, en condiciones normales los paquetes son aceptados o rechazados a nivel hardware por la tarjeta de red según la MAC address de destino. Pero, cuando la tarjeta de red esta en modo promiscuo, las cosas cambian radicalmente. Todos los paquetes son aceptados y procesados, por lo que todos pasan directamente al kernel del sistema operativo. Es ahora éste el que debe encargarse de filtrarlos y decidir cuáles van a ser procesados por otras aplicaciones y cuáles van a ser descartados.

En suma, los paquetes son filtrados en niveles superiores al nivel hardware/físico al que los filtra la tarjeta de red cuando ésta no se encuentra en modo promiscuo. La consecuencia de este cambio es un incremento drástico del tiempo de procesamiento por paquete, en varios órdenes de magnitud en la mayoría de los casos.

¿Cómo aprovechar esta idea en el desarrollo de tests de detección de sniffers? Un buen procedimiento es realizar pruebas de latencia (mediante el envío de pings y la obtención de estadísticas de tiempos de respuesta medios) de cada una de las máquinas cuando la red se encuentre poco saturada, repitiendo estos tests mientras se satura la red con tráfico basura. El tiempo de respuesta de cada una de las máquinas, por supuesto, empeorará significativamente, pero el de una máquina con una tarjeta de

red en modo promiscuo será varios órdenes de magnitud superior al del resto. Así es como, idealmente, se detecta la presencia de un sniffer.

Pero, evidentemente, pueden encontrarse dificultades en caso de hallarse frente a un atacante lo suficientemente astuto. ¿Cómo puede evitar ser detectado por este tipo de técnica? Hay, entre otras, dos posibilidades principales.

Fijémonos en que, de forma implícita, para que esta idea funcione se necesita que sólo una pequeña parte de las máquinas esté en modo promiscuo. Desde luego que éste es el caso en la mayoría de las situaciones, entre otros motivos porque no tiene demasiado sentido instalar varios sniffers en el mismo segmento de red, ya que todos tendrían acceso al mismo tráfico. Sin embargo, es importante tener presente que, en el caso extremo en que todas las máquinas de un determinado segmento tengan instalado un sniffer, esta técnica no va a permitir localizarlos y, lo que es peor, induciría a pensar que no hay ninguno. Si el segmento tiene un número relativamente pequeño de máquinas (de 2 a 5) podría suceder que el atacante instale sniffers en todas las máquinas para burlar este test de latencia, que es considerado por muchos como el mejor de los que se conocen.

Hay otra técnica ingeniosa para conseguir burlar esta prueba, o al menos para convertir sus resultados en poco concluyentes. Como siempre, la idea es muy sencilla: el atacante puede sospechar que acaba de comenzar una búsqueda de sniffers si detecta que el tráfico en la red ha aumentado drásticamente (ha comenzado la fase de inundación de la red) respecto al trafico habitual (podría estar realizando estadísticas y tener un umbral de alarma). En ese caso, simplemente cambia la tarjeta de red a modo no promiscuo durante unos segundos, para pasar el test con normalidad y luego cambia de nuevo a modo promiscuo.

Esta técnica de evasión ya ha sido implementada en un sniffer gratuito, así que no se puede descartar que comience a utilizarse masivamente por parte de un buen número de atacantes en breve plazo.

HERRAMIENTAS DE DETECCIÓN REMOTA DESDE EL MISMO SEGMENTO DE RED

La mejor herramienta de detección de sniffers en la actualidad es AntiSniff, de L0pht. Se trata de un programa comercial con una versión de evaluación de 15 días que implementa todos los tests citados anteriormente, junto a algunas variaciones muy interesantes.

Sin embargo, al poco de anunciarse su aparición se desarrolló un sniffer gratuito y con código fuente disponible llamado Anti-AntiSniff que no es detectado por ninguno de los test que AntiSniff realiza. AntiSniff está disponible para Windows NT/2000 y hay una versión de prueba gratuita y con código fuente disponible en desarrollo para diferentes versiones de Unix que, eso sí, está muy por detrás de la versión de Windows NT/2000 en cuanto a interfaz gráfica.

Le sigue, en cuanto a utilidad y no precisamente de cerca, el proyecto Sentinel, que tiene la ventaja de ser un proyecto público y abierto que permite acceder al código fuente del programa. En la actualidad es sólo capaz de realizar unos pocos tests, muchos menos que AntiSniff, pero tiene un interesante futuro por delante que hace recomendable su seguimiento.

Antiguos programas de detección de sniffers que en su día fueron pioneros, como NEPED, están ahora totalmente superados porque tanto Sentinel como AntiSniff implementan las técnicas que les hicieron tan útiles en el pasado.

CONCLUSIONES

La lucha contra los sniffers es un tema de permanente actualidad: cada vez se desarrollan técnicas más sofisticadas para su detección y, casi al instante, se crean nuevos modelos de sniffers que las burlan. Se trata de una lucha desequilibrada en contra del que los busca que, además, tiene que convivir con el hecho de que su tarea es, a veces, imposible.

No obstante, es fundamental que cualquier administrador de seguridad conozca todos los programas que pueden ayudarle a detectar sniffers en sus redes, junto a las técnicas de evasión que un atacante puede utilizar para luchar contra ellos ([11],[12]), para comprender cuál es el valor exacto de los resultados que puede esperar de sus herramientas.

Es posible que en un futuro próximo esta lucha se desequilibre en favor del administrador de sistemas, cuando el cifrado del tráfico de red se extienda y se convierta en práctica habitual en la mayoría de las empresas. La introducción generalizada de VPNs , favorecida por el nuevo marco de IPv6, que previsiblemente se producirá en los próximos años, también contribuirá a dificultar, quizá a imposibilitar, la tarea del atacante.

En la actualidad hay algunas soluciones sencillas que pueden ser aplicadas para minimizar el impacto de un sniffer, fundamentalmente la segmentación de la red bien sea fisicamente o por medio de hubs inteligentes que impiden que ningún tráfico no destinado a ellas llegue a la tarjeta de red.

Evidentemente, estas dos recomendaciones finales no pueden aplicarse siempre por motivos bien de arquitectura de red bien económicos, así que, en la inmensa mayoría de los casos el administrador de seguridad se verá obligado a seguir las citadas propuestas e implantar un programa de inspección periódica capaz de detectar sniffers, desactivar tarjetas y disparar alarmas. ■

Julio César Hernández
José María Sierra
Arturo Ribagorda
Benjamin Ramos
Laboratorio de Seguridad en TIC
Departamento de Informática
UNIVERSIDAD CARLOS III DE MADRID
{jcesar,sierra,arturo,benja1}@inf.uc3m.es

REFERENCIAS

[1] La última versión de **AntiSniff** puede conseguirse junto a todas sus especificaciones técnicas en www.l0pht.com/antisniff/

[2] La *web* del proyecto **SENTINEL**, con información sobre el proyecto y el código fuente del programa, es ww.subterrain.net/projects/sentinel

[3] **NEPED** es un interesante programa que durante mucho tiempo fue una de las mejores herramientas. Ahora ha sido superado por AntiSniff y Sentinel, pero puede obtenerse en ftp://apostols.org/AposTools/snapshots/neped/neped.c

[4] El *sniffer* más elaborado es el **Anti-AntiSniff** obra de Mike Perry y que puede conseguirse junto a otras herramientas en hackpalace.com/hacking/unix/sniffers. Se considera imprescindible para demostrar de forma efectiva las limitaciones a estas técnicas.

[5] **A Methodology for studying untrusted software**. *LOGIN* June 2000 by Hernández, J.C., Sierra, J.M., Ribagorda, A., Ramos, B.

[6] **CPM** puede obtenerse en ftp://coast.cs.purdue.edu/pub/tools/unix/cpm/

[7] **FCheck** es una herramienta Perl de control de integridad que permite escoger la primitiva criptográfica a utilizar. Se recomienda MD5 o SHA-1 como las únicas alternativas razonables. Puede conseguirse en http://sites.netscape.net/fcheck/fcheck.html

[8] **Claymore** es un conjunto de *scripts* Perl que utiliza MD5 como primitiva criptográfica para realizar el checksum. Puede obtenerse en linux.rice.edu/magic/claymore

[9] **L6**, sucesor del conocido **L5**, es una buena alternativa para el control de integridad de los ficheros. Como los anteriores, ha sido desarrollado en Perl para convertirlo en una herramienta multiplataforma. Puede utilizar MD5 y/o SHA-1. Hay administradores de sistemas que usan ambas simultáneamente, algo completamente absurdo para cualquiera que tenga unos conocimientos siquiera discretos de criptografía. Puede obtenerse en www.pgci.ca/

[10] **ViperDB** es un conjunto de *scripts* en Perl que hacen el control de integridad más sencillo y configurable. Una mala alternativa a Tripwire, mucho menos recomendable, porque no utiliza primitivas criptográficas de ningún tipo y no se autochequea. Puede obtenerse en ww.resentment.org, aunque se recomienda su uso.

[11] **Detecting sniffers on your network** es un interesante artículo que puede leerse en: www.securiteam.com/unixfocus/Detecting_sniffers_on_your_network.html

[12] **Basic Packet-Sniffer Construction from the Ground Up** es un interesante artículo de Chad Renfro que puede conseguirse en: packetstorm.securify.com/sniffers/Sniffer_construction.txt