SEGURIDAD EN INTERNET. PARTE II: UN CASO PRACTICO

Jesús Ibáñez Martínez
jesus@dif.um.es

Antonio Gómez Skarmeta
skarmeta@dif.um.es

Humberto Martínez Barberá
humberto@fcu.um.es

Departamento de Informática y Sistemas
Facultad de Informática
Universidad de Murcia

RESUMEN

En este trabajo presentamos los distintas etapas que se requieren para la puesta en marcha de un firewall en una organización que se quiera conectar de forma segura a Internet. Así mismo, se describirán las distintas opciones en cuanto a software y un estudio detallado de una implementación real.

Palabras Clave: Seguridad, Internet, Firewalls

INTRODUCCION

Tomando como base que se conocen los distintos problemas de seguridad que aparecen en una red conectada a Internet, vamos a mostrar un modelo de seguridad basado en firewall, indicando los pasos necesarios para su puesta en marcha. Dicho modelo se va a basar en tres pilares: robustez, portabilidad y apertura. Para garantizar la portabilidad no se utilizarán herramientas dependientes de un sistema concreto. Por otro lado, dado el ritmo al que se detectan agujeros de software y al que aparecen nuevos servicios, un sistema abierto es la mejor garantía para que nuestro sistema de seguridad perdure gran cantidad de tiempo de forma eficaz.

Lo primero que se tiene que decidir acerca de un firewall es su topología. Tomaremos como base el modelo *screened host*, debido a su versatilidad, fiabilidad y bajo coste. Hay que tener en cuenta que un sistema basado en este modelo es fácilmente portable a uno con topología *screened subnet*. La topología *screened host* consta, básicamente, de un router que realizará filtro de paquetes y

un ordenador (bastión) que se utilizará como proxy. Para preparar tanto el router como el bastión se han de seguir una serie de pasos, que se indicarán a continuación. Para el router:

- definir la lista de accesos, para redirigir el tráfico externo al bastión.
- rechazar los mensajes externos con dirección interna (spoofing).
- deshabilitar la respuesta a los mensajes *ICMP redirect*.
- quitar el soporte al acceso telnet.

El bastión será un equipo UNIX, debido a que la mayoría del software que necesitaremos se ha desarrollado sobre este sistema. Para preparar el bastión se seguirán los pasos:

- instalación segura del sistema operativo, desde los discos originales del fabricante, y posterior fijación de los bugs conocidos mediante los patches correspondientes.
- deshabilitar los servicios no requeridos, que lo único que harían es aumentar el riesgo (*NFS* y relacionados, *RPC*s, *ftpd*, *bootd*, *bootpd*, *rshd*, *rlogind*, *rexecd*, etc).
- instalar los servicios requeridos (es el caso de los servidores proxy).
- quitar los ejecutables no esenciales, para reducir el riesgo.
- configurar el sistema de logs, y montar alguna herramienta de análisis de logs (en concreto swatch permite analizar los logs en tiempo real conforme se van generando).
- montar como sólo lectura todos los sistemas de ficheros que sea posible.
- ejecutar un chequeador de integridad periódicamente (tripwire es una excelente elección; permite aplicar hasta 10 funciones hash a cada entrada de los sistemas de ficheros).
- realizar un backup del sistema completo y totalmente limpio.

SERVICIOS

Todo lo que hemos hecho hasta ahora va encaminado a conseguir que los usuarios de la red que protegemos puedan acceder a los servicios que ofrecen otras redes sobre Internet, y a ofrecer también nosotros ciertos servicios al resto de redes. Sin embargo, veremos a continuación que tanto la demanda como la oferta de cada uno de esos servicios conlleva una serie de riesgos. Veremos cuáles son éstos para cada uno de los servicios básicos de Internet, analizando las posibles soluciones, y presentando la opción que nos ha parecido en cada caso la más adecuada para nuestro modelo.

Correo SMTP

Si bien el protocolo SMTP en sí no presenta problemas de seguridad, de los servidores de SMTP no podemos decir lo mismo. El servidor de SMTP más común en Unix, Sendmail, ha sido explotado en diversas ocasiones. El problema de Sendmail se deriva de dos causas: por una lado se trata de un servidor propenso a bugs (su código consta de cientos de miles de líneas en C) y por otro corre como root (explotarlo da acceso total a la máquina). Han aparecido diversos servidores de SMTP, sustitutos del sendmail, que si bien no han sido tan explotados como éste, la evidencia sugiere que se debe a que son menos populares, no a que sean menos vulnerables.

FWTK incluye una solución conceptualmente sencilla: smap/smapd. Smap es una especie de wrapper de SMTP que corre en un directorio aislado (chroot) como un usuario sin privilegios, recoge los mensajes y los salva en un directorio determinado. Smapd es un demonio que cada cierta cantidad de segundos chequea el directorio y envía el correo al MTA (sendmail u otro) para su reparto.

Nosotros instalamos smap/smapd en el bastión, y redirigimos todo el correo a través suyo. Con esto evitamos ataques directos al puerto 25. Como MTA usamos sendmail. También usamos

smap en el servidor interno, porque apenas supone carga, y proporciona un mayor grado de seguridad.

Por otro lado, para garantizar que el auténtico remitente del correo es el que aparece en la cabecera del mismo la mejor solución es el empleo de un esquema de clave pública. Con él, cualquier correo que nos llegue encriptado con la clave privada del remitente sabremos que ha sido enviado realmente por él, puesto que sólo él conoce su clave privada. Además, el esquema de clave pública nos permite el envío confidencial de correo. En efecto, si encriptamos el correo con la clave pública del destinatario, sólo él podrá leerlo.

Las dos opciones con mayor proyección de cuantas siguen este esquema son PEM (Privacy-Enhanced Electronic Mail) y PGP (Pretty Good Privacy). PEM es el estándar oficial en la pila de protocolos TCP/IP, y de él no hay aún ninguna implementación de dominio público. PGP, en cambio, es un paquete software disponible para prácticamente todos los sistemas operativos. En ambos casos (como en todos los que implican encriptación) el mayor problema es el de la gestión de claves. PEM requiere una jerarquía de autoridades certificadoras. PGP, por su parte, utiliza un modelo descentralizado de certificación. Nosotros optamos por PGP.

Telnet

El problema de Telnet es que envía toda su información sin encriptar, lo que le hace vulnerable a ataques sniffing y hijacking.

Para las conexiones salientes se puede usar tanto filtro de paquetes como proxy. Nosotros usamos tn-gw (servidor proxy incluido en FWTK). También probamos SOCKS (requiere clientes especiales), TCPR y GAU. Si elegimos tn-gw es porque por una parte es el más completo de los que

hemos probado, y por otra comparte el fichero de configuración con otras herramientas que empleamos.

Con las conexiones entrantes hay que tomar medidas de seguridad adicionales, porque las claves introducidas a nuestras máquinas podrían ser escuchadas con sniffers. Contra ésto se pueden emplear esquemas de autenticación basados en passwords no reusables. El FWTK incluye un servidor de autenticación que se integra con los servidores proxy del mismo paquete, permitiendo que estos requieran autenticación para su uso. Esto se consigue añadiendo una opción en el archivo de configuración indicándolo. El servidor de autenticación del FWTK es capaz de soportar los esquemas de autenticación SecurId, Snk y S/Key. SecurId y Snk son productos comerciales que además requieren dispositivos especiales para el cálculo del password, lo que los hace inservibles para nuestro modelo. S/Key, diseñado por Leslie Lamport y desarrollado por la empresa Bellcore, es un producto de libre distribución que funciona aplicando iterativamente un *algoritmo hash seguro*, empezando con una semilla inicial. El cálculo del password por parte del cliente no requiere ningún dispositivo especial. Se realiza mediante un pequeño programa disponible gratuitamente para prácticamente todos los sistemas y entornos.

El problema es que S/Key usa MD4, que actualmente es vulnerable. De hecho ya se han presentado diversos crackers de S/Key en Internet (Monkey es el más conocido). Sin embargo, Wietse Venema ha solucionado este problema reprogramando las librerías de S/Key para que empleen MD5 (actualmente seguro) en lugar de MD4. Nosotros compilamos las librerías de S/Key de Wietse Venema y las linkamos con el servidor de autenticación de FWTK. Con ello obtenemos un esquema de autenticación basado en passwords no reusables "seguro" (dentro de la seguridad que estos esquemas pueden ofrecer).

Pero aunque los esquemas de passwords no reusables evitan que un password escuchado pueda ser usado para entrar en nuestras máquinas, aún son posibles los ataques por hijacking. Contra éstos la única defensa es el cifrado de la conexión completa.

La opción que nos ha parecido más atractiva es ssh. Ssh permite comunicaciones encriptadas seguras entre dos hosts sobre una red insegura. Utiliza un esquema asimétrico para el intercambio de una clave privada con la que posteriormente se encriptará la conexión siguiendo un esquema simétrico (actualmente soporta IDEA, DES, 3DES, ARCFOUR, y TSS). Soporta autenticación .rhost, autenticación .rhost combinada con autenticación de host RSA, autenticación RSA desafío-respuesta, o autenticación basada en password.

Además ssh puede crear circuitos transparentes encriptados entre dos puertos, ya sean de máquinas diferentes o de la misma máquina, que pueden ser aprovechados por otros protocolos. Así, soporta la redirección de conexiones X sobre un canal encriptado.

El servidor de ssh puede compilarse en prácticamente cualquier sistema Unix. Actualmente, además de para Unix, existen dos clientes para Windows, uno de ellos de libre distribución.

FTP

FTP usa dos conexiones separadas: una para comandos y resultados entre el cliente y el servidor (canal de comandos), y la otra para los ficheros y listados de directorio transferidos (canal de datos). En el extremo del servidor el canal de comandos usa el puerto 21, y el canal de datos usa normalmente el 20. Para iniciar una sesión FTP, un cliente coge 2 puertos TCP para él por encima del 1023. Usa el primero para abrir la conexión del canal de comandos al servidor y entonces usa el comando PORT para decirle al servidor el número del segundo puerto que el cliente quiere usar para el canal de datos. El servidor abre entonces la conexión del canal de datos. Esta apertura (por parte

del servidor al cliente) complica las cosas para definir reglas de filtro de paquetes que aseguren que las conexiones son iniciadas desde dentro, porque servidores FTP externos intentarán iniciar conexiones de datos a clientes internos. Es más, esas conexiones irán a puertos en un rango inseguro.

La mayoría de servidores y muchos clientes FTP soportan un modo alternativo que permite al cliente abrir los dos canales, el de comandos y el de datos. Este modo se conoce como modo pasivo o modo PASV. Con este modo, se pueden evitar los problemas en el filtro, porque todas las conexiones se abrirán desde dentro por el cliente. No todos los clientes y servidores soportan el modo pasivo.

Con lo visto, tenemos dos posibilidades para permitir que nuestros clientes accedan a servidores externos de FTP a través de un cortafuegos: usar clientes en modo pasivo y definir las reglas correspondientes en el filtro de paquetes; usar clientes en modo normal a través de un servidor proxy de FTP instalado en el bastión. Nosotros hemos instalado ftp-gw, el servidor proxy para ftp incluído en el FWTK. Para ello hay que permitir conexiones TCP desde un puerto 20 externo hacia puertos sobre el 1023 en el bastión. Esto no debe ser un problema porque en esos puertos no vamos a tener nada en principio.

Además, para los casos en los que necesitemos acceder a servidores internos no anónimos desde las redes externas, lo haremos redirigiendo el canal de comandos por un canal encriptado previamente con ssh, de modo que el password viajará encriptado por la red.

Por otra parte, para correr un servidor de FTP anónimo interno, se puede usar un servidor "seguro". En particular se puede emplear el ftpd incluido en el FWTK, o el wuarchive, más robusto para cargas altas. Además, para los casos en los que necesitemos acceder a servidores internos no anónimos desde las redes externas, lo haremos redirigiendo el canal de comandos por un canal encriptado previamente con ssh, de modo que el password viajará encriptado por la red.

Worl Wide Web

Para el acceso de nuestros clientes a servidores se puede usar tanto filtro de paquetes como servidores proxy. Hemos probado los dos servidores proxy más usados: el httpd del CERN y el http-gw incluido en el FWTK. La ventaja del http-gw es la simplicidad de su código. Sin embargo, aunque el código del servidor del CERN es mucho mayor, y por tanto más propenso a bugs, lo cierto es que hasta el momento no se le ha detectado ninguno. El servidor de CERN aporta además la posibilidad de realizar caché de las páginas consultadas, lo que mejora el rendimiento, por eso lo hemos elegido.

Dada la debilidad de los servidores de Web, si se desea correr uno, no se debe correr en el cortafuegos, se debe usar un bastión dedicado. Se debe ejecutar sin permisos (nobody por ejemplo). Se deben deshabilitar todos los servicios que no se usen en ese bastión. Se debe de llevar cuidado con los permisos. Se recomienda correr el servidor en un entorno protegido (chroot). Se deben evitar los modos en los que alguien podría dar de alta un programa en el sistema (por mail o ftp, por ejemplo), y ejecutarlo a través del servidor HTTP. Se deben controlar cuidadosamente los cgi a los que el servidor puede acceder, evitando que puedan hacer algo más que aquello para lo que se han escrito.

Desde el punto de vista del cliente, la ejecución de código Java y JavaScript es potencialmente peligrosa. En particular, Java, pese a haber sido diseñado como un lenguaje que propiciase la escritura de código seguro, las implementaciones desarrolladas hasta la fecha presentan bugs que propician la creación de herramientas de ataque. Por ejemplo, las applets (aplicaciones en Java) supuestamente sólo son capaces de hablar al servidor del que son origen, sin embargo en la práctica hay un bug que les permite hablar a cualquiera.

X11

En X11, el terminal del usuario es un servidor. Un intruso con acceso a un servidor X11 puede ser capaz de obtener la memoria de pantalla, leer pulsaciones de tecla, y generar pulsaciones de teclas. Un atacante, desde cualquier lugar de Internet, puede probar servidores X11. Si están desprotegidos, podrá realizar la conexión, en principio sin notificación al usuario.

Hay diversos mecanismos de protección en X11, aunque no son tan útiles como uno podría esperar. El primer nivel es una autenticación basada en la dirección del host, y por tanto susceptible de ser atacada mediante spoofing. Un segundo mecanismo usa la llamada magic cookie. La aplicación y el servidor, comparten una cadena secreta; los procesos sin esta cadena no se pueden conectar al servidor. Pero darle la cadena al servidor de una forma secreta es difícil. Un tercer mecanismo de seguridad X11 usa un esquema criptográfico. Este podría ser bastante seguro, pero sufre el mismo problema de distribución de clave que el mecanismo de autenticación magic cookie. Existe una variante Kerberos, pero aún no ampliamente disponible.

En cualquier caso, como normal general, no se debe permitir a clientes de la red externa que se puedan conectar a nuestros servidores X11, y si se tiene que hacer, debe usarse un servidor proxy de X11 en el bastión. Nosotros instalamos x-gw (el servidor proxy de X11 incluido en el FWTK) en el bastión. Cuando un intruso intente conectarse a un servidor X11 interno, x-gw mostrará al usuario del servidor una ventana preguntándole si acepta esa conexión. Además, para los casos en los que necesitemos acceder desde el exterior a máquinas internas a través de X11, lo haremos redirigiendo la conexión a través de un canal encriptado con ssh.

Finger

El servicio finger puede utilizarse para obtener información sobre usuarios que tienen una cuenta dentro del sistema consultado. El problema es que ofrece demasiada información útil a potenciales atacantes: información personal (útil para adivinar passwords), cuándo se usó la cuenta por última vez (las cuentas nunca o poco usadas son más proclives a tener malos passwords), desde dónde se conectó por última vez (y por tanto un destino para un ataque indirecto).

La salida más útil de finger de cara al exterior es el mapeo entre el nombre y la dirección de correo, de modo que una solución razonable sería ofrecer sólo esa información. De hecho ya existen diversos servidores finger que se comportan así. La solución por la que nosotros hemos optado es tan sencilla como segura. Simplemente modificamos la entrada de finger en el achivo inetd.conf para que visualice el contenido de un archivo, que sólo da información genérica sobre las direcciones de correo y un teléfono de contacto para posibles dudas.

Con esto resolvemos el problema de finger, y además obligamos a los administradores a seguir una política homogénea en la asignación de cuentas de correo. Además, con esta solución da lo mismo en que máquina tenga el usuario la cuenta, porque desde el exterior todas las consultas finger se resuelven en el cortafuegos de un modo global.

Usenet news

NNTP es un protocolo relativamente simple. Realmente no se conocen ataques que hayan usado NNTP en sí. Para correr un servidor de news interno lo mejor es ponerlo en la red interna y dejar fluir el tráfico entre éste y la máquina que le sirve. Esto se puede conseguir con una apropiada configuración en el filtro de paquetes, o bien usando un proxy genérico en el cortafuegos.

Nosotros optamos por emplear plug-gw (el proxy genérico incluido en el FWTK) en el bastión para permitir el tráfico entre el servidor de news interno y el servidor proveedor. Plug-gw se configura de forma que cuando el servidor externo de news (el proveedor) se conecte al puerto de NNTP del bastión, éste se conectará automáticamente al servicio NNTP del servidor interno, y cuando el servidor interno se conecte al puerto de NNTP del bastión, éste se conectará automáticamente al servicio NNTP del servidor externo.

EXPLOTACION

Ahora que ya está correctamente instalado y configurado el firewall, hemos de tener en cuenta dos puntos importantes: cómo mantener el firewall y qué hacer en caso de incidencias. Con un firewall, el concepto de seguridad pasiva (configurar y usar) es poco aceptable, ya que si se colara algún intruso podríamos no detectarlo. Para ello tendremos que tener un plan de seguridad activa (mantenimiento), de forma que se examinen diversos patrones para detectar intentos de ataque, e incluso éxitos. Swatch nos permite realizar todo esto. Es capaz tanto de analizar los logs mientras estos se generan como de realizar un análisis más exhaustivo del fichero de logs, teniendo en cuenta el contexto. El FWTK dispone además de una serie de scripts para generar estadísticas de los logs. Es muy importante mantenerse al día a base de la información que aparece en las listas de correo electrónico, así como en los boletines del CERT (Computer Emergency Response Team) y del CIAC (Computer Incident Advisory Capability). En caso de que se detecte un ataque contra nuestra red, se ha de actuar sin prisa y sin pánico, compreendiendo lo que se hace y documentándose. Lo primero que se ha de hacer es notificar el indicente al CERT-CC o a algún otro equipo de respuesta a incidentes, de forma que nos ayuden a trazar el ataque y a diagnosticar el modo en que comprometieron nuestras máquinas. Además se realizará un backup de los equipos implicados para poder realizar un estudio posterior.

CONCLUSIONES

Se ha presentado un sistema flexible, seguro y abierto, que se encuentra actualmente en explotación en la red de la Comunidad Autónoma de la Región de Murcia desde hace varios meses. Además se ha podido conseguir todo esto con software de dominio público.

REFERENCIAS

- Chapman, D.Brent and Zwicky, Elizabeth D. "*Building Internet Firewalls*". O'Reilly & Associates, Inc. Septiembre, 1995.

- Cheswick, William R. and Bellovin, Steven M. "*Firewalls and Internet Security. Repelling the Wily Hacker*". Addison-Weslwy Profesional Computing Series. 1994.

- Cruz, Francisco. "*Seguridad en redes y sistemas*", Boletín de RedIRIS, nº 35. Madrid. Abril 1.996.

- Curry, David A. "*Unix System Security. A Guide for Users and System Administrators*". Addison-Weslwy Profesional Computing Series. 1992.

- Farmer, Dan; Venema, Wietse. "*Improving the Security of Your Site by Breaking Into it*". 1993.

- Hansen, Stephen E.; Atkins, E. Todd. "*Automated System Monitoring an Notification with Swatch*", LISA, Monterrey, CA. November 1.993.

- Joncheray, Laurent. "*A Simple Active Attack Against TCP*". Merit Network, Inc. Abril 1995.

- Kim, Gene H.; Spafford, Eugene H.; "*Experiences with Tripwire: Using Integrity Checkers for Intrusion Detection*", Purdue Technical Report, COAST Laboratory, Department of Computer Sciences, Purdua University, West Lafayette, Febrero 1.994.

- Kim, Gene H.; Spafford, Eugene H.; "*The Design and Implementation of Tripwire: A File System Integrity Checker*", COAST Laboratory, Department of Computer Sciences, Purdue University, West Lafayette, Febrero 1.994.

- Liu, Cricket; Peek, Jerry; Jones, Russ; Buus, Bryan; Nye, Adrian. "*Managing Internet Information Services*". O'Reilly & Associates, Inc. Diciembre, 1994.

- Martínez, Rubén. "*Seguridad en redes*". Centro de Comunicaciones CSIC RedIRIS. Madrid 1996.

- Martínez, Rubén. "*Servicios de Distribución y Certificación de Claves PGP*", Boletín de RedIRIS, nº 35. Madrid. 1.995.

- Morris, Robert T. "*A Weakness in the 4.2BSD Unix TCP/IP Software*". AT&T Bell Laboratories. Murray Hill, New Jersey 07974. Febrero 1985.

- Newman, David; Melson, Brent. "*Can Firewalls Take the Heat?*". Data Communications. Noviembre 1995.

- Portillo, Eloy; López, Lourdes. "*Seguridad en redes telemáticas. Parte I: la problemática de la seguridad*", Boletín de RedIRIS, nº 31. Madrid. Abril 1.995.

- Portillo, Eloy; López, Lourdes. "*Seguridad en redes telemáticas. Parte II: entornos seguros*", Boletín de RedIRIS, nº 32. Madrid. Julio 1.995.

- Ranum, M.J. "*Thinking about Firewalls*", Trusted Information Systems, Inc.

- Siyan, Karanjit; Hare, Chris. "*Internet y Seguridad en Redes*". Prentice Hall. 1995.

- Stallings, William. "*Network and Internetwork Security. Principles and Practice*". Prentice Hall International Editions. 1995.

Las herramientas mencionadas en este artículo, así como otras muchas que se han testeado para llegar a decidirnos por éstas, pueden ser accedidas desde la página:

<http://www.cs.purdue.edu/coast/coast.html>