



*presenta...*

# Conferencias de Seguridad Informática

*Ponente:*

***Death Master***

# Índice de contenidos

<b>Índice de contenidos</b> .....	2
<b>Día 1 – Introducción y seguridad en la máquina local</b> .....	3
• Introducción: La importancia de la seguridad.....	3
• Conceptos previos.....	4
• Sistemas Operativos.....	4
• Breve introducción a redes y protocolos.....	6
• Algunos fallos típicos de seguridad en la máquina local.....	11
• Usuarios, contraseñas y archivos de contraseñas.....	11
• Fallos típicos en Sistemas Operativos.....	14
• Breve introducción a la seguridad en redes locales.....	17
<b>Día 2 – Seguridad en Internet</b> .....	20
• Introducción: Panorama actual en Internet.....	20
• Antivirus.....	21
• Introducción a los virus: tipos y precauciones básicas.....	21
• Uso de antivirus: cómo usarlos y cómo actuar ante un virus.....	23
• Virus e integridad en la máquina local.....	25
• Firewalls e IDS.....	26
• Conceptos básicos sobre Firewalls e IDS.....	26
• Cómo configurar un Firewall.....	28
• Alertas y logs.....	29
• Ataques e integridad en la máquina local.....	31
• Seguridad en redes de pares (p2p).....	33
<b>Día 3 – Seguridad en las comunicaciones</b> .....	36
• Introducción: La privacidad, bien escaso.....	36
• Spyware: qué es y cómo evitarlo.....	37
• Privacidad en Internet.....	40
• Uso de proxys anónimos.....	40
• Protocolos seguros y certificados de seguridad.....	42
• Criptografía y autenticación de ficheros: MD5.....	48
• Criptografía asimétrica: el sistema PGP.....	49
• Introducción a la clave asimétrica: sistema PGP/gnuPG.....	49
• Los algoritmos: RSA y DH/DSS.....	50
• Encriptación de ficheros, correo electrónico y comunicaciones.....	53
<b>Concluyendo</b> .....	55
<b>Agradecimientos</b> .....	56
<b>Distribución de este documento</b> .....	57
<b>Licencia</b> .....	58

# 1. Introducción y seguridad en la máquina local

## - Introducción: La importancia de la seguridad.

Hoy en día ya no cabe preguntarse si es necesaria la seguridad. Más bien debemos preguntarnos –cada cual en su caso- porqué es necesaria la seguridad.

El concepto de ordenador individual casi ha desaparecido. Internet está llegando cada vez más a todos los rincones de nuestra vida, desde el ordenador al teléfono móvil e incluso la televisión. Es una ventana al mundo, el concepto de aldea global ha pasado de una utopía a algo muy real. Pero no olvidemos que las ventanas tienen dos lados, y que bien se puede mirar la calle desde la apacible tranquilidad de tu casa... pero también se ve tu casa desde la calle. La gente pone cortinas, usa ventanas ahumadas... ¿por qué? Porque a nadie le gusta que le invadan su intimidad.

Creo que es un buen símil para poder extraer una idea intuitiva de la importancia de la seguridad. El crecimiento de las conexiones hace que la necesidad de seguridad sea imperiosa, urgida por la información que se transmite a través de estas conexiones. Ya no sólo leemos el correo electrónico o visitamos determinados sitios Web. Hoy día compramos por Internet, consultamos los movimientos bancarios a través de Internet, trabajamos a través de Internet... en definitiva, la cantidad de datos "sensibles" que atraviesan nuestras líneas telefónicas ha crecido y lo seguirá haciendo día a día. Y eso ya no es preocupación de empresas o de gurús de la informática, sino una preocupación muy real de cualquier usuario de a pie.

Internet es libre, y se ha convertido en el medio de comunicación más imparcial que puede existir, gracias a su concepción descentralizada. En la red de redes podemos encontrar las mejores fuentes de información, así como los más péfidos sitios con contenidos de dudosa legalidad y moralidad. Esto, unido al más temprano acceso de las nuevas generaciones a Internet cada día, crea una nueva responsabilidad social, siendo imprescindible educar a los padres sobre unos mínimos conceptos de informática, seguridad e Internet para que puedan velar por la integridad de los más jóvenes. Soy el primero que repudia la censura y cree que la toda la información debe estar ahí y la esencia de la libertad reside en poder elegir, para bien o para mal, pero también que esa decisión debe ir acompañada de una capacidad de decisión. Mientras tanto, es responsabilidad de los padres el educar a sus hijos también en el ámbito de Internet. Si no dejas que tus hijos hablen con desconocidos por la calle, ¿por qué sí en Internet?

Así pues el fin de estas pequeñas conferencias es el impartir unos conocimientos básicos sobre informática y seguridad, orientadas a que cualquiera pueda comprender y utilizar los recursos que se nos brindan sin la necesidad de poseer unos conocimientos previos, y fomentar quizá la curiosidad en vuestro interior para que esto no sea sino el principio.

Empecemos.

**Death Master**

## - Conceptos previos

Algunos se estarán preguntando porqué he dicho hace breves instantes que no eran necesarios unos conocimientos previos y el primer bloque a tratar se titula, precisamente, conocimientos previos. Justamente, los conocimientos previos necesarios para comprender las conferencias, los voy a introducir yo. Es imprescindible que conozcamos cómo funciona y se estructura a grandes rasgos una red o un sistema operativo, así como introducir algunos conceptos que luego irán apareciendo y que debemos comprender. Hablaremos de puertos, de servicios, de protocolos...

No pretendo complicar con charlas técnicas este paso previo que para la mayoría será el mayor escollo de los tres días, sino que pretendo introducir ideas intuitivas para comprender qué es cada cosa y porqué. Que nadie se asuste. ;-)

### Sistemas Operativos

Un ordenador está dividido principalmente en **dos tipos de componentes: hardware y software**, parte física y lógica respectivamente. En algunos momentos la separación no es tan clara, pues existen ciertos componentes que están a caballo entre esos dos mundos. Es el caso, por ejemplo, de la **BIOS (Basic Input-Output System)**, un software básico para que el ordenador pueda arrancar, y que se encuentra alojado en la **CMOS (Complementary Metal Oxide Semiconductor)**.

Así pues, tenemos unos componentes físicos y unos componentes lógicos, y necesitamos que interactúen entre ellos y puedan comunicarse. Esta comunicación se realiza mediante un tipo básico y especial de software que se denomina Sistema Operativo, y que permite disponer de un entorno de software que sea funcional bajo nuestro hardware. De forma intuitiva podríamos decir que es el encargado de que los "programas" se entiendan con "el ordenador".

Sistemas operativos hay muchos. **Los más conocidos y usados hoy en día son Microsoft Windows, GNU/LINUX, BSD, Unix, MacOSx...** pero principalmente hoy día podemos dividir el uso doméstico de sistemas operativos en PC (mac es otra historia) entre Windows y Unix-like (GNU/LINUX y BSD se incluyen entre los sistemas Unix-like). Debido a la orientación al usuario de a pie de esta serie de conferencias, me centraré en ejemplificar y estudiar Windows y GNU/LINUX. No voy a entrar en comparar ambos sistemas operativos porque no es el fin de estas charlas. Cada sistema tiene partidarios y detractores, e igualmente los que me conocen saben de qué pie cojeo y que soy acérrimo defensor del sistema Linux. Pero a la hora de hablar de la seguridad, hablaré exclusivamente de seguridad en cada uno, no quiero entrar a valorar nada que luego me pierdo con mis cruzadas personales :-P

Dentro del sistema **Windows**, podemos distinguir entre dos tipos: **los sistemas Windows 9x y los sistemas Windows NT**. Hace unos años, los sistemas NT estaban orientados principalmente a estaciones de trabajo en red y empresas, relegando para el usuario doméstico los sistemas 9x, mucho menos seguros y mucho menos estables que los NT. En parte era un problema de compatibilidad con determinados tipos de software, como por ejemplo el lúdico. Hoy en día esa tendencia se está invirtiendo, con los sistemas operativos **Windows 2000 y Windows XP en sus distintas versiones**, pues están basados en tecnología NT y se adaptan perfectamente al ámbito doméstico. Cada día están más generalizados, y poca gente queda hoy día usando **sistemas Windows 9x (que incluyen Windows 95, Windows 98, Windows 98 SE y Windows ME)**.

Los nuevos sistemas NT, aunque realizan un consumo de memoria y recursos de sistema bastante mayor, son también más estables y seguros (principalmente en el ámbito de máquina local) que los 9x. Aún así, muchos de los fallos de seguridad y características son heredadas y han surgido algunos fallos nuevos derivados de la nueva tecnología -sobre todo a nivel de red-, si bien su kernel o núcleo del sistema es conceptualmente distinto.

En el caso de Linux, encontramos multitud de versiones, pues se trata de un sistema operativo libre. **GNU/LINUX se compone principalmente de un kernel o núcleo llamado Linux, y un entorno de aplicaciones (GNU), entre ellos la llamada shell o intérprete de comandos**, que es la interfaz de comunicación con el sistema.

Debido a su concepción como software libre, cualquier componente es reemplazable, actualizable o modificable, lo que da lugar en la práctica a infinidad de sistemas distintos, no ya tanto con el entorno de aplicaciones, variable en cualquier sistema operativo, sino en cuanto al kernel y la shell del sistema. Esas "versiones" de Linux las denominamos distribuciones, y a su vez una misma distribución puede encontrarse en muchas versiones. **Algunas de las principales distribuciones son Debian, Red Hat, SuSe, Mandrake, Gentoo, Slackware, LinEx, Knoppix...**

Existen otros sistemas Unix-like que hay que considerar debido a la extensión que también poseen y al crecimiento que están experimentando, que son los sistemas operativos **BSD, entre ellos OpenBSD y FreeBSD** (aunque existen otros sistemas BSD propietarios). Se trata de sistemas similares a Linux, si bien ambos parten de una implementación libre de BSD diferente y que ha tenido un desarrollo paralelo en el tiempo. El entorno de aplicaciones de Linux y BSD hoy día es o puede ser el mismo, y la principal diferencia radica en el kernel del sistema. En lo referente al estudio de seguridad de los sistemas al nivel que queremos hacerlo, es indiferente hablar de uno o de otro, así que lo dicho para Linux será aplicable a sistemas BSD y a cualquier Unix en general.

Ahora llega el momento de introducir algunos conceptos sobre Sistemas Operativos que serán necesarios más adelante:

Denominamos **proceso** a un **algoritmo interpretado o compilado que está ejecutándose en un sistema y se encuentra residente en la memoria RAM**. Cualquier programa en ejecución se representa como un proceso, que puede ser auditado, analizado, matado, etc.

Denominamos **servicio** a un tipo de proceso que está a la escucha, es decir, **no está haciendo nada a no ser que sea requerido, en cuyo caso atiende convenientemente la petición**. Un servicio suele cargarse de forma permanente en memoria hasta que sea matado o el Sistema Operativo lo cierre. Un ejemplo de servicio sería un servidor FTP configurado para escuchar un puerto determinado.

Denominamos **daemon o demonio** a un tipo especial de **servicio que escucha a otros servicios o procesos**. El demonio es un "programa que escucha a otro programa". Un ejemplo claro de esto sería **INETD en sistemas Unix** (el padre de todos los demonios :-P), que se encarga de escuchar a los procesos o servicios que soliciten conexión de red, y atiende esas peticiones. Un ejemplo "de andar por casa" de demonio sería el **httpd de Apache**.

Existe un tipo especial de software encargado de comunicar el entorno de aplicaciones con el kernel o núcleo del sistema. Este software se denomina **shell o interfaz de usuario**. Aunque una shell puede ser tanto una línea de comandos como una interfaz gráfica, hoy en día se usa el término shell prácticamente como sinónimo de intérprete de comandos. ¿Y qué es un intérprete de comandos? Para que sea más sencillo de comprender, lo ilustraré de forma práctica: en Windows lo que conocemos por intérprete de comandos es "una ventana de MS-DOS" (de hecho este MS-DOS era un intérprete de comandos) que podemos abrir ejecutando `command.com` en sistemas Windows 9x o bien `cmd.exe` en sistemas Windows NT. En Linux el intérprete de comandos es la consola (si bien en Linux podemos usar distintas shells).

Así pues, podemos decir que una shell interpreta y ejecuta unas instrucciones que nosotros le hemos proporcionado por medio de una línea de comandos y sus instrucciones. Este concepto será importante más adelante cuando hablemos de **sintaxis de ejecución y argumentos** de diversos comandos y programas.

Hay otros conceptos que harán falta más adelante, pero que por sus peculiaridades y su complejidad técnica, no será necesario conocer cómo funcionan. Algunos casos podrían ser la *pila de sistema*, el *buffer de memoria*...

Aunque esto sólo suponen unas pinceladas de lo que es un Sistema Operativo, creo que con esto es suficiente para introducir el concepto, así como comprender sus funciones y estructura y conocer a grandes rasgos los diversos Sistemas Operativos existentes. En definitiva, será lo que necesitaremos comprender para poder seguir correctamente el resto de los temas, que son los que nos interesan.

## Breve introducción a redes y protocolos

La introducción a los conceptos necesarios sobre redes y protocolos es sin duda más compleja y larga, pues mi intención es introducir algunos conceptos complejos como pueden ser el modelo de red basado en capas ISO/OSI, así como introducir de una forma general lo que es el estándar de comunicación en las principales redes (Internet, LAN, WAN...): el conjunto de protocolos TCP/IP. Existen otros tipos de protocolos como NetBIOS/NetBEUI, IPX/SPX, AppleTalk y el más importante (después de TCP/IP): redes TokenRing. Esto sirva como muestra de que no todo es TCP/IP, aunque solamente el estudio en profundidad de este conjunto de protocolos llevaría meses e incluso años. Comencemos.

Todos conocemos intuitivamente el concepto de **red informática: dos o más ordenadores conectados y compartiendo una información**. A partir de aquí debemos preguntarnos cómo se comparte esa información. Sí, la información entre dos ordenadores se comparte a través de unos y ceros, pero debe ser organizada de alguna manera para que la estructura de datos de un fichero se conserve y siga teniendo sentido después de su transferencia. Así es como nace el concepto de paquete. **Un paquete de datos, como su nombre indica, es un conjunto encapsulado de información**, que no sólo contiene los datos que nosotros queremos transmitir (o parte de ellos, como ya veremos) sino también una información destinada a almacenar información sobre el propio paquete y que permite la gestión de nuestros datos. Esa información extra son más unos y ceros que indican, por ejemplo, el origen y el destino de un paquete, y se conocen por el nombre genérico de cabecera de un paquete.

Para comprender el nacimiento de Internet, hay que comprender el nacimiento de su predecesora, ARPAnet. A finales de los años sesenta, la ARPA (Advanced Research Project Agency), una agencia de investigación de Estados Unidos, quería **desarrollar una red estratégica de ordenadores importantes que no estuviera centralizada**, la cual pudiese sobrevivir a un ataque físico de cualquier tipo, se produjera en la parte de la red que se produjera. No importaba que un ordenador fuera destruido, pues la red sobrevivía. La primera idea fue que los ordenadores no podían estar cerca físicamente, y además debería existir una comunicación constante entre las partes de la red, para permitir que los datos viajaran libremente y evitar una pérdida de información en caso de la destrucción de un componente.

En **septiembre de 1969**, se instaló el primer intercambiador de paquetes en la Universidad de California en Los Angeles (UCLA). Ese ordenador Honeywell 516, junto a otros que fueron añadiéndose constantemente, supusieron el nacimiento de ARPAnet. Durante los años setenta, ARPAnet creció a un ritmo vertiginoso, que ni los propios investigadores esperaban. Según fue creciendo la red, se fue tomando conciencia de la necesidad de un conjunto de protocolos que aseguraran el control del creciente volumen de paquetes y la gran variedad de estos. El gran problema de ARPAnet era que su protocolo (entendiendo protocolo como conjunto de reglas) hacía que la etiqueta de cada paquete fuera ligeramente diferente según el ordenador que la hubiese creado, provocando en muchas ocasiones la imposibilidad de recibir o redirigir paquetes en el tráfico, conforme la diversidad de la red se iba haciendo manifiesta. **Este protocolo, NCP (Network Control Protocol) no estaba previsto para trabajar en una red con configuraciones tan distintas como satélites, computadoras o radios.**

Así pues, **el 1 de Enero de 1983, NCP fue sustituido por el nuevo Protocolo de Control de Transporte / Protocolo de Internet, TCP/IP (Transfer Control Protocol / Internet Protocol)**. El antiguo NCP necesitaba que los paquetes tuvieran un tamaño y estructura predefinidos, pero TCP/IP podía trabajar con paquetes de todos los tipos, tamaños y redes, e incluso paquetes de sistemas informáticos con cualquier otra red, independientemente de sus peculiaridades. De ahora en adelante **denominaremos a cada uno de esos "puntos" de la red como nodo o host**. Con el nacimiento de TCP/IP, nació Internet como red, aunque el nacimiento de la Internet Web gráfica que la mayoría conocen como Internet tuvo que esperar a 1989 para que fuera propuesta por Tim Berners-Lee, un físico del CERN (Laboratorio Europeo de Física de Partículas). Pero la semilla ya estaba plantada.

Para entender cómo está estructurado el protocolo TCP/IP, hay que introducir primero el **modelo de red ISO/OSI**. Este modelo fue propuesto por la Organización de Estándares Internacionales, un conjunto multinacional de matemáticos, físicos e ingenieros que incluye organizaciones de estándares de más de 100 países. Este modelo ISO/OSI está estructurado en siete capas para organizar el hardware y el software de una red en modelos funcionales bien definidos.

No deja de ser un modelo, y de hecho **TCP/IP implementa cinco de las siete capas únicamente** (si bien en muchos manuales aparece implementado con solamente cuatro capas, englobando la capa física y la capa de enlace a datos en una única capa conocida como capa de acceso a red. Nosotros nos centraremos en el estudio de TCP/IP como modelo de cinco capas), pero por ser el modelo es útil conocerlo y entenderlo.

	Capa	Tipo de datos
7	Capa de aplicación	<i>Datos</i>
6	Capa de presentación	<i>Datos</i>
5	Capa de sesión	<i>Datos</i>
4	Capa de transporte	<i>Segmentos</i>
3	Capa de red	<i>Paquetes</i>
2	Capa de enlace de datos	<i>Tramas</i>
1	Capa física	<i>Bits</i>

Este sistema de capas está pensado de forma que cada capa se comunique con las adyacentes para transportar la información. Cuando los datos salen de la máquina emisora, atraviesan la pila de protocolos “hacia abajo”, desde la capa de aplicación hasta la capa física. En la máquina receptora este flujo es inverso, desde la capa física a la capa de aplicación. Ahora es el momento de hablar brevemente de los **principales protocolos en TCP/IP**:

**IP (Internet Protocol):** Se trata de un protocolo de **capa de red**, su misión es **encontrar la red a la que pertenece el host destino**, para ello los routers intermedios que existen entre el emisor y el receptor consultan sus tablas de rutas para encontrar el camino adecuado y entregar el paquete de datos al receptor.

**TCP (Transfer Control Protocol):** Se trata de un protocolo de **capa de transporte**, que **mueve los paquetes entre las distintas aplicaciones**. Es el principal protocolo de transporte de datos de TCP/IP.

**UDP (User Datagram Protocol):** UDP es un protocolo que **se usa de forma alternativa a TCP** en ocasiones que requieren un protocolo **más sencillo y rápido aunque menos fiable** de transporte (debido a la ausencia de control de errores que TCP posee y UDP no). También pertenece a la **capa de transporte** pero sólo transporta los paquetes de uno en uno.

**ICMP (Internet Control Message Protocol):** Protocolo de **uso principalmente informativo y de control**. Informa de errores de red y otras condiciones que requieren de atención especial del software de red. Se trata de un protocolo de **capa de red**.

Capa	Función	Protocolos
<i>Aplicación</i>	Funciones de red especializadas: transferencia de archivos, terminales virtuales...	FTP, HTTP, SMTP...
<i>Presentación</i>	Formato de datos, conversión del código de caracteres, codificación de los datos...	ZIP, AVI, JPEG...
<i>Sesión</i>	Negociación y establecimiento de una conexión con otro host.	RPC, NFS, LDAP...
<i>Transporte</i>	Envío de datos.	TCP, UDP
<i>Red</i>	Dirige los paquetes de información por las redes.	IP, ICMP, IGMP, RIP, OSPF, BGP...
<i>Enlace de datos</i>	Transferencia de unidades direccionables de frames y comprobación de errores.	SLIP, CSLIP, PPP...
<i>Física</i>	Transmisión de datos binarios a través de la red de comunicaciones.	ISO 2110, IEEE 802, IEEE 802.11...

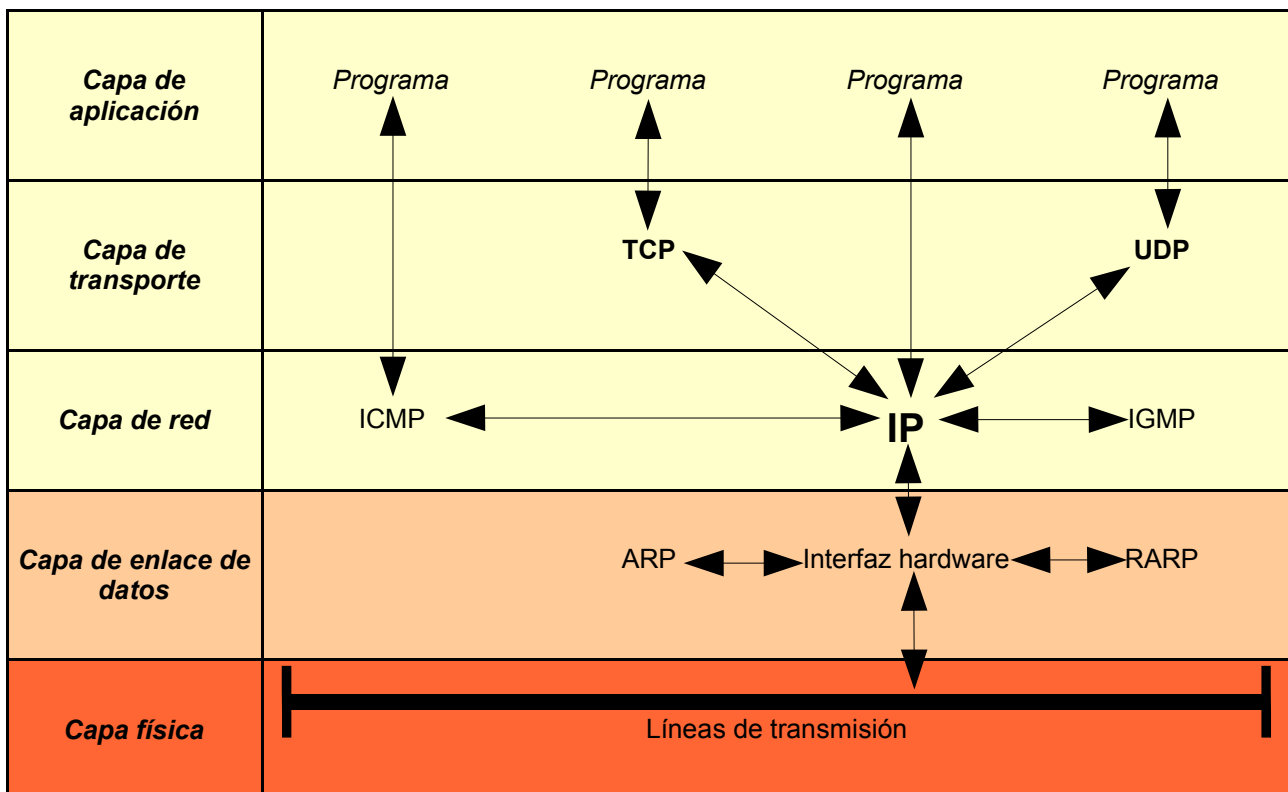
La **capa física** define la señal eléctrica para cada canal de transmisión. No es lo mismo el cable cruzado UTP que se usa en las conexiones ethernet (IEEE 802.3) que las ondas de una conexión wireless (802.11x). También se especifica el método para codificar los bits transmitidos por la red.

La **capa de enlace de datos** define cómo transmite la capa física los paquetes de la capa de red entre los distintos hosts. Define la conversión de datos en bits mediante los **protocolos que controlan la construcción e intercambio de paquetes (denominados data frames o frames simplemente)**. Se incluyen dos módulos de protocolos: **ARP (Address Resolution Protocol)** y **RARP (Reverse Address Resolution Protocol)** para crear y resolver direcciones para las transmisiones de la capa física.

La **capa de red** define cómo se envía la información recibida por la capa de transporte y cómo se dirige la red a los distintos hosts. En TCP/IP, la capa de red se denomina también capa de Internet. Es el corazón de cualquier red basada en TCP/IP, pues **contiene el módulo IP**. Los demás protocolos son **auxiliares a IP** y se usan en ocasiones especiales, como errores y multitransmisión (broadcast). En una red TCP/IP la capa de red encapsula a todos los protocolos menos al de resolución de direcciones (ARP).

La **capa de transporte** transfiere los datos entre aplicaciones y controla la transmisión a través de la **capa de red**. Utiliza TCP o UDP para construir estas transmisiones. En el caso de TCP, se usa un flujo de datos fiable con verificación del destinatario (típico ejemplo, el saludo en tres tiempos de conexiones TCP). En el caso de UDP, se usan datagramas (paquetes) para enviar y recibir datos, sin existencia de ningún tipo de verificación ni de conexión.

La **capa de aplicación** envía los datos a la capa de transporte y recibe los datos devueltos por ésta. En TCP/IP, la capa de aplicación "engloba" a las capas de aplicación, presentación y sesión del modelo ISO/OSI. Por estar en la parte superior de la pila de protocolos, se comunica directamente con el software del ordenador.



Si se ha entendido todo lo visto hasta ahora, ya hay una base sobre el conocimiento de redes, pues los rudimentos sobre la comunicación entre hosts ya están asentados. Ahora voy a introducir unos pocos conceptos más que nos serán útiles a la hora de tratar temas más adelante. Estos conceptos son más sencillos y muchos de ellos ya son de sobra conocidos de una forma natural.

En primer lugar debemos hablar de la **dirección IP**. En una red TCP/IP, **cada ordenador conectado está identificado de forma unívoca por un número de dirección**. Este número consta de 4 bytes (32 bits) y aunque existen muchas representaciones para él, la más común y a la que estamos acostumbrados es la notación en punto decimal, en la que **una IP tiene la forma xxx.xxx.xxx.xxx con 4 grupos de dígitos del 0 al 255** separados por puntos (Ej.: 212.50.60.70). Aunque existen direcciones y rangos de direcciones especiales (127.0.0.1 es localhost, loopback o máquina local, rangos como 192.168.x.x y 172.16.x.x son redes locales, x.x.x.255 son direcciones de multitransmisión o broadcast, 255.x.x.x es una máscara de subred...) en general una IP corresponde a una única máquina en Internet. Esta forma de representar y entender la IP que he explicado se denomina IPv4 y es el estándar actual, si bien cada vez crece más en uso e importancia el protocolo IPv6. No voy a entrar en detalles al respecto de este otro protocolo, basta señalar que su función es idéntica pero usa direcciones de un número mayor de bytes, con lo cual el número de posibles IP crece de forma exponencial. Es importante saber, así mismo, que IP se sitúa en la capa tercera o capa de red del modelo TCP/IP, pues esto limita su rango de trabajo y encapsulamiento de datos a esa capa. De forma general la dirección IP se presenta...:

RED		SUBRED	HOST
192	168	0	1
8 bits = 1 byte	8 bits = 1 byte	8 bits = 1 byte	8 bits = 1 byte

Esta representación de IP es solamente **un ejemplo**, pues la subred no tiene porqué ocupar 8 bits ni tiene porqué limitarse al segundo byte de la dirección IP. De hecho podemos encontrar direcciones IP de **clase A** (RED-HOST-HOST-HOST), de **clase B** (RED-RED-HOST-HOST) o de **clase C** (RED-RED-RED-HOST). **La subred se define "quitando" bits a la parte de host** de la IP, de forma que se pueden formar muchas redes de pocos nodos o pocas redes de muchos nodos a partir de una dirección de clase determinada.

En nuestro ejemplo vemos una dirección de clase B a la que hemos "quitado" 8 bits para definir la subred.

En el caso de redes conectadas a Internet a través de un encaminador, enrutador o router, la IP pública se asigna al mismo. Todos los equipos, de cara a Internet, poseen esa misma IP, si bien dentro de la red local poseen otra dirección que les identifica dentro de la misma y de cara al router. Un equipo de esa subred puede ser 192.168.0.1, y otro 192.168.0.2, pero ambos serán la misma IP de cara a un servidor externo, por ejemplo el servidor web google. Este proceso es posible gracias al **protocolo NAT (Network Address Translation)** y al **NAPT (Network Address Port Translation)**.

Igual que existe una dirección lógica para designar a un ordenador (la dirección IP), **existe así mismo una dirección física** y única que identifica, no ya a un ordenador, sino a cada dispositivo de red. Así, un PC con varios dispositivos de conexión distintos, puede tener más de una dirección física (por ejemplo, un ordenador con dos tarjetas de red). **Esta dirección se denomina dirección MAC (Media Access Control) y se graba de fábrica en el NIC (Network Interface Card)**. Está compuesta por seis bytes (48 bits), **con la forma XX-XX-XX-XX-XX-XX con 6 grupos de dígitos del 00 al FF en hexadecimal**, separados por guiones (Ej.: 00-D0-E8-55-B1-9A). Los 3 primeros bytes corresponden a la **OUI (Organization Unique ID)**, un código de identificación que es distinto para cada fabricante; mientras que los 3 últimos bytes corresponden a la **dirección física del dispositivo**, y que son asignados de forma arbitraria por los fabricantes para identificar sus productos. La dirección MAC se sitúa en la segunda capa o capa de enlace de datos del modelo TCP/IP, y unida al protocolo **ARP (Address Resolution Protocol)**, **encargado de la traducción de direcciones IP a direcciones MAC**, es la base de la comunicación de paquetes en Internet, debido al enrutamiento de paquetes por la dirección física.

Organization Unique ID (OUI)			Dirección física del dispositivo (NIC, etc)		
00 D0 E8			55 B1 9A		
8 bits = 1 byte	8 bits = 1 byte	8 bits = 1 byte	8 bits = 1 byte	8 bits = 1 byte	8 bits = 1 byte

Otro concepto básico a entender es el de **puerto lógico o puerto de comunicaciones**. Ya sabemos que las conexiones se realizan entre hosts representados por una dirección IP o MAC, pero ¿qué ocurre cuando son múltiples las conexiones? Entra en juego el concepto de puerto. En TCP/IP, cada máquina tiene un número de puertos (concretamente **disponemos de 65536**, desde el 0 al 65535) que son las distintas “ventanas” por las que se puede mirar hacia el exterior o desde las que se puede mirar el interior de nuestro host. Hablando de una forma general, los puertos pueden ser TCP o UDP según la conexión establecida, y pertenecen por consiguiente a la cuarta capa o capa de transporte del modelo TCP/IP. **Representaremos esta información de la forma IP:PUERTO** (Ej.: 127.0.0.1:139). Por norma general, el subrango de puertos 0-1024 está definido por convenio para los protocolos de nivel de aplicación más comunes (FTP, HTTP, POP3, IMAP...), dejando libre el subrango 1025-65535 para usos varios o software cliente. En la práctica podemos usar cualquier puerto para establecer cualquier tipo de conexión.

Un concepto también importante es el de cliente/servidor. En una conexión generalmente **un host “sirve” una determinada información, siendo el host al otro lado de la conexión el encargado de “recoger” esa información**. Intuitivamente podemos decir que el servidor es aquel que ofrece algo y un cliente es aquel que solicita algo. Por ejemplo, cuando nos conectamos con nuestro navegador web a <http://www.google.com/> nuestra máquina se conecta al puerto 80 del servidor de google (216.239.37.99) y solicita la información de su índice web (GET / HTTP/1.1), mostrando esta información en nuestra pantalla de forma transparente al usuario. Y aquí hay otro concepto importante. El puerto estándar para conexiones web es el puerto 80, pero ese es el puerto del servidor, siendo el del cliente usualmente cualquier puerto aleatorio superior al 1024. Esto suele cumplirse para casi cualquier conexión, pues si bien los puertos de servidor suelen ser fijos, los puertos de cliente suelen ser casi siempre aleatorios (más adelante sabremos porqué). Podemos comprobar fácilmente esto visitando con nuestro software navegador dos o más sitios web y consultando las conexiones existentes entre nuestro host e Internet mediante el comando netstat (invocado mediante “*netstat -an*” en sistemas Windows o bien “*netstat -tupan*” en sistemas Linux).

Sé que TCP/IP es algo muy complejo, y sé igualmente que los que estén familiarizados con las redes sabrán que esto no es más que arañar la superficie, pero mi intención no era crear un curso de TCP/IP, sino simplemente lograr una comprensión al nivel necesario para permitirnos comprender las bases de una seguridad en redes.

## **- Algunos fallos típicos de seguridad en la máquina local**

El primer bloque que vamos a tratar es el de la seguridad en la máquina local, nuestra propia máquina. Antes de comprender los riesgos que entraña una conexión, debemos comprender mejor aún los riesgos que entraña nuestra propia máquina y nuestro propio sistema operativo. En la primera parte trataremos un tema central y universal en la seguridad: Las contraseñas. Elección de contraseñas seguras, archivos de contraseñas, usuarios y sus permisos... En segundo lugar trataremos los fallos típicos en determinados protocolos que tienen los sistemas operativos, así como los errores y problemas que tiene un sistema recién instalado.

### **Usuarios, contraseñas y archivos de contraseñas**

Un problema bastante común es el desconocimiento del concepto de “usuario” y las implicaciones que tiene en la seguridad. **Cuando accedemos a un sistema, necesitamos hacer “login”**, entrar a ese sistema **con un nombre de usuario y una contraseña** (cuando no es necesaria, la contraseña se considera vacía pero eso no quiere decir que no exista).

Hay que introducir algunos conceptos, como por ejemplo el concepto de usuario como un conjunto de datos que permiten a alguien acceder a los recursos de una máquina. Se hablan también de grupos de usuarios como un conjunto de usuarios con algo en común (generalmente una política de privacidad). Por último se habla de permisos acerca de lo que puede y lo que no puede hacer un usuario, los recursos de la máquina a los que tiene acceso, la posibilidad de modificar aspectos de la misma... Se hablan generalmente de los permisos “de escritura”, “de lectura” y “de ejecución”. Estas tres definiciones son muy generales y pueden ser aplicables a casi cualquier Sistema Operativo.

Dejando aparte el sistema de usuarios de Windows 9x por su simplicidad y por el hecho de que las diferencias entre ellos son nominales simplemente, vamos a centrarnos en los tipos de cuentas que existen en nuestros sistemas.

**En los sistemas Windows NT (y por consiguiente 2000 y XP), existen tres tipos de privilegios principalmente: usuario (cuenta limitada), administrador y sistema.** Existe un tipo “especial” de usuario que es el invitado, englobado en una cuenta de usuario normal más restrictiva. La cuenta de administrador es la que puede realizar todo tipo de cambios en el sistema, instalar programas y añadir entradas al registro. La cuenta de sistema es una cuenta con privilegios “especiales” de administrador, pero aunque de cara al sistema operativo existe, no pueden asignarse dichos permisos a una cuenta de usuario (lo cual no quiere decir que no puedan usarse). Así mismo, existen ciertas cuentas predeterminadas consideradas como recursos compartidos administrativos, y que se distinguen por el símbolo del dólar al final de su nombre (Ej.: C\$, WINNT\$...)

Windows NT tiene un sistema de seguridad formado por cuatro componentes: **LSA (Local Security Authority), SAM (Security Account Manager), SRM (Security Reference Monitor) y UI (User Interface)**. No vamos a explicar todo esto en detalle, pero sí conviene tener en la cabeza el concepto de Windows NT como un sistema operativo con un modelo de seguridad basado en objetos, donde **cada objeto posee distintos atributos en la ACL (Access Control List)**, la lista de control de acceso que dice qué grupos y qué usuarios pueden acceder a qué objetos y con qué permisos. Más adelante veremos cómo algunos de estos términos aparecen.

<b>Componente</b>	<b>Descripción</b>
<i>Autoridad de Seguridad Local (LSA)</i>	También conocido como Subsistema de seguridad. Se trata del componente central de la seguridad de NT. Controla la directiva local de seguridad y la autenticación de usuarios, así como de la generación y el registro de los mensajes de auditoría
<i>Administrador de Cuentas de Seguridad (SAM)</i>	Se encarga del control de las cuentas de grupo y usuario y proporciona servicios de autenticación de usuario para la autoridad de seguridad local.
<i>Monitor de Seguridad de Referencia (SRM)</i>	Se encarga de la validación de acceso y de la auditoría para la autoridad de seguridad local. Comprueba las cuentas de usuario mientras el usuario intenta acceder a los archivos, directorios y demás. Genera mensajes de auditoría según sus decisiones. Contiene una copia del código de validación de acceso para asegurar que el Monitor de seguridad de referencia protege los recursos de forma uniforme en el sistema.
<i>Interfaz de usuario (UI)</i>	Es lo que ve el usuario y lo que se utiliza para realizar tareas administrativas.

**En sistemas Linux existe una política de privacidad mucho más estricta, con dos tipos básicos de usuarios: los usuarios y el root o superusuario.** El grupo de usuarios comprende cualquiera que no sea el root, y los permisos pueden ser muy diversos y son definidos por el root, que es el usuario con permisos totales (bueno, existen distribuciones donde esto no se cumple, pero son excepciones como Trusted Debian). Existen igualmente varias cuentas que se activan por defecto en la mayoría de sistemas Unix (makefsys, mountfsys, unmountfsys, checkfsys, lp, daemon, trouble, nuucp, uucp...) pero no son relevantes para nosotros.

En cualquier sistema operativo podemos -y debemos si queremos una mínima seguridad- definir los permisos de usuarios, en Windows podemos usar la Directiva de Seguridad Local (que puede ejecutarse mediante el comando "%SystemRoot%\system32\secpol.msc /s") y en Linux podemos definir a mano los permisos para cada fichero o carpeta, así como asignar caducidad a las cuentas, contraseñas y demás.

No es mi intención hacer un análisis detallado de los usuarios en un sistema, pero hay algo que conviene tener en cuenta siempre que miremos por la seguridad de nuestra máquina: controlar la integridad del sistema, que los permisos permanezcan como nosotros los asignamos. Igualmente hay una máxima para la seguridad: nunca trabajes habitualmente con una cuenta de administrador/root. Se debe usar por sistema la cuenta con el menor número de privilegios posible, de forma que una hipotética situación que comprometa al sistema afecte en la menor medida posible.

Cuando hablamos de contraseñas, hay ciertas normas básicas que recordar:

- 1- Nunca asignar la misma contraseña a todos nuestros pares de user/password.
- 2- Nunca elegir contraseñas obvias: el login, fechas, número de DNI, edad, nombre de mascota o pareja, etc.
- 3- Elegir una contraseña que combine mayúsculas, minúsculas, números e incluso símbolos a ser posible.
- 4- Tu contraseña ha de ser lo más larga posible.
- 5- Si existe la opción de pregunta secreta, que las respuestas no sean evidentes. Sé realista, la mayoría de la gente sabe -o puede averiguar- cómo se llama tu mascota o cuándo empezaste a salir con tu pareja. :-P
- 6- Obviamente, no comunicar tus contraseñas absolutamente a nadie. Por norma general, ningún administrador te la va a pedir, así que de entrada no te fíes y no se la des aunque te llame el presidente de tu ISP.
- 7- Es conveniente llevar un registro de contraseñas, sobre todo si sigues mi máxima número uno y todas y cada una de tus contraseñas son distintas. Pero si las apuntas todas en un cuaderno al lado del PC o las guardas en un archivo de texto plano, pierde efectividad. Para este efecto, recomiendo apuntarlas como se quiera (texto plano, base de datos, etc) y criptografiar el fichero de una forma segura (hablaremos de criptografía en la tercera conferencia).
- 8- En cuentas importantes o especialmente sensibles, cambiar con frecuencia la contraseña por una nueva y completamente distinta.
- 9- Si tu ordenador lo usan otras personas además de ti o si no es "seguro", no usar sistemas de autoguardado de contraseñas como el de Mozilla o Internet Explorer.
- 10- Cerciorarte de que tu ordenador o cualquiera en el que hagas login conserve intacta su integridad.

Estas normas son aplicables para cualquier circunstancia en la que sea necesario el uso de contraseñas, no sólo en el inicio de sesión de un sistema operativo.

Ahora es el turno de hablar de los archivos de contraseñas. Cada sistema operativo tiene su propia forma de guardar las contraseñas para su verificación y de proteger esos ficheros de ojos indeseados. Como en todo, hecha la ley hecha la trampa, y es posible comprometer un sistema por muy diversos medios gracias a los archivos de contraseñas. Vamos a ver cómo almacena cada sistema estos ficheros:

Los **sistemas Windows 9x** guardan los ficheros de contraseñas en un archivo con el **mismo nombre del usuario y extensión pwl**, en la carpeta de instalación de Windows. Por ejemplo, un usuario PEPE tendría su archivo de contraseñas en "C:\Windows\PEPE.pwl". Si alguien logra tener acceso a ese fichero (lo cual en un sistema 9x no supone ningún problema), la contraseña está comprometida, pues existen multitud de programas en la red especializados en descifrar las contraseñas de este fichero.

Para los **sistemas Windows NT** la cosa es algo más segura. Estos sistemas no guardan las contraseñas tal cual o las encriptan, sino que usan una función *hash*. Una función hash (en la tercera conferencia hablaremos más en profundidad de las funciones hash) es un algoritmo que dada una entrada, obtiene como salida una cadena de forma unidireccional e unívoca. Dos entradas no pueden producir la misma salida y a partir de la salida no se puede recomponer la entrada. **El sistema, cuando se define la contraseña, aplica el hash y la reduce a un valor**, que es comprobado cada vez que se accede mediante la aplicación de esa función hash a la contraseña. Los sistemas Windows más modernos usan SYSKEY como algoritmo de protección de contraseñas. Estos ficheros de hash están muy protegidos y no pueden ser accedidos ni por el administrador en ejecución, y se encuentran en "C:\Windows\System32\Config\SAM" y "C:\Windows\System32\Config\sam.log". Según la versión de NT (3.x, 4.x, 5.0 ó 5.1) puede existir uno de estos ficheros o los dos. Igualmente, existe un tercer fichero de contraseñas en "C:\Windows\repair\sam" que si bien no contiene todos los hashes, contiene los principales del sistema, entre ellos el administrador, y este fichero sí puede ser accedido de una forma más sencilla desde la máquina local (aunque se necesitan privilegios de administrador). Igual que en el caso anterior, la posesión o acceso a uno de estos ficheros compromete el sistema completo, pues existen multitud de programas capaces de aplicar un ataque de fuerza bruta al sistema de contraseñas de hash basado en ir probando contraseñas ("a, b, c, ... , aa, ab, ac, ...") hasta dar con la contraseña que devuelva un hash idéntico, y esa será la buena.

Para **sistemas Linux**, hoy día se usa casi exclusivamente el sistema **shadow** en lugar del más antiguo passwd. Ambos son conceptualmente idénticos y almacenan las contraseñas en un archivo cifrado, con la diferencia de que el fichero passwd (alojado en "/etc/passwd") puede ser accedido por cualquier usuario, mientras que para poder acceder al fichero shadow ("/etc/shadow" por defecto, aunque puede cambiarse) se necesita ser root. Acceder a este fichero es igualmente motivo de compromiso del sistema, pues con el archiconocido software "**John The Ripper**" (y con muchos otros) se pueden desencriptar las contraseñas de cualquier fichero passwd o shadow en un sistema Unix.

Sistema	Fichero de contraseñas	Encriptación	Acceso
Windows 9x	%SystemRoot%\USUARIO.pwl	SI	Cualquiera localmente
Windows NT	%SystemRoot%\System32\Config\SAM %SystemRoot%\System32\Config\sam.log %SystemRoot%\Repair\sam	Tipo hash	Restringido al administrador
Linux	/etc/passwd	SI	Cualquiera localmente
	/etc/shadow		Restringido al root

La obtención de esos ficheros puede darse de muchas formas distintas: un virus o troyano, un usuario local que obtiene los permisos necesarios mediante una escalada de privilegios, alguna vulnerabilidad de un servicio concreto de nuestro sistema que permita a alguien ganarse algún acceso con los privilegios necesarios, un usuario que puede arrancar con un disco de arranque o incluso otro sistema operativo y copiar el archivo de contraseñas... son muchas las posibles maneras de obtener esa información, y desde luego tantas como hay para obtenerlo, las hay para que no te lo quiten. ;-)

## Fallos típicos en Sistemas Operativos

Aunque mucha gente pensará lo contrario, todo sistema operativo tiene fallos, y más aún según terminamos de instalarlo. No me voy a ocupar de los fallos o vulnerabilidades que se solucionen con una visita al centro de actualizaciones de nuestro sistema operativo (WindowsUpdate, RedHat Network, apt-get upgrade...) con la excepción de fallos graves. Asumo y de paso recomiendo encarecidamente que lo primero que se haga al instalar un sistema (bueno, inmediatamente después de configurar un firewall decente, pero de eso hablaremos mañana) sea actualizar todos los problemas conocidos.

La mayoría de los fallos iniciales en un sistema vienen dados por puertos que el sistema deja abiertos por defecto. Nuestra prioridad es conocer esos puertos, valorar sus riesgos, y actuar en consecuencia.

Cuando hablamos de sistemas Windows, es inevitable hablar de **NetBIOS (NETwork Basic Input/Output System)**. NetBIOS nació como una API de conexión de computadoras de la mano de IBM. Se ideó un protocolo de comunicación para poder conectar varios hosts mediante la API. **Ese protocolo junto a la API de NetBIOS se denominó NetBEUI (NetBIOS Extended User Interface)**. La principal diferencia con otros tipos de redes, es que las máquinas no se denominaban con números de direcciones (como en IP o IPX) sino con nombres de máquina, lo cual para redes locales está muy bien, pero para grandes redes la cosa no es tan sencilla. Pasando el tiempo, se construyeron implementaciones de NetBEUI para trabajar sobre IPX y un poco más tarde, sobre TCP/IP, que es lo que hoy en día entendemos como NetBIOS. Por tanto, lo que hoy se denomina NetBIOS en realidad es **la implementación de NetBEUI sobre TCP/IP**. Familiarmente, se trata de el "Compartir impresoras y archivos para redes Microsoft" (junto a SMB) de los sistemas Windows y el SAMBA de los sistemas Unix. Bien, tras esta pequeña charla introductoria, vamos al meollo de la cuestión: Windows es un sistema preparado para trabajar con NetBIOS para casi cualquier tarea de red local. Así pues, un sistema Windows por defecto deja abiertos algunos de los puertos que usa este protocolo. Los puertos más comunes de NetBIOS son: netbios-ssn (139), netbios-ns (137), netbios-dgm (138); y SMB: ms-ds (445). Adicionalmente debemos hablar e introducir unas nociones de SMB (Server Message Block), un protocolo cliente/servidor de capa de presentación que permite compartir archivos e impresoras entre sistemas. En realidad se trata de un sistema que originalmente corría bajo NetBIOS y NetBEUI y que pasó a implementarse bajo TCP/IP. **En conjunto, NetBIOS y SMB forman el sistema de compartir archivos e impresoras de Windows**. Y es así mismo su gran talón de Aquiles. En adición, el protocolo RPC (Remote Procedure Call), tristemente famoso hoy día a causa del virus Lovsan o Blaster, trabaja con algunos de estos puertos, lo cual multiplica su peligro...

El clásico problema de tener estos puertos abiertos era que cualquiera con un poco de picardía podía acceder a los recursos compartidos de tu red local desde Internet. Mucha gente pensaba que si no tenía recursos compartidos (lo cual, por cierto, es imposible, porque **Windows comparte por defecto determinados recursos de sistema** de los que hemos hablado antes: aquello de C\$, WINNT\$...), no necesitaba proteger esos puertos. Quizá antes fuera así, pero hoy día han aparecido diversas vulnerabilidades asociadas a esos puertos que suponen problemas de seguridad muy graves. En un primer momento, el problema más grave era **una vulnerabilidad del protocolo SMB que permitía lanzar una petición malformada que reseteaba el ordenador víctima**, con el consiguiente problema en servidores. Este fallo está corregido con los últimos service pack de NT 4.0, 2000 y XP y no supone un problema muy grave, excepto para equipos no parcheados.

Tras esto, apareció el fallo más famoso que ha habido desde el IIS code/decode (un fallo que permitía escalada de directorios mediante la traducción UNICODE). El fallo del protocolo RPC que propició un caos considerable en Internet el verano del 2003. Básicamente **el fallo RPC consiste en una petición de llamada de procedimiento remoto malformada al puerto 135 que provoca un desbordamiento de buffer, dejando el sistema comprometido** a ejecución arbitraria de código. Rápidamente se extendieron multitud de exploits para esta vulnerabilidad, y en relativamente poco tiempo se podía con un único exploit (válido para cualquier versión de WINNT 5.x) aprovechar esta vulnerabilidad para obtener una shell remota con privilegios de sistema. Obviamente, esto supone uno de los problemas de seguridad más graves al que puede enfrentarse un sistema. Aunque las vulnerabilidades fueron parcheadas por Microsoft con relativa celeridad (y teniendo en cuenta que son unos parches que han coleado durante meses por errores en la interminable sucesión de parches), la falta de concienciación y el desconocimiento de los usuarios agravó esta situación. Como muchos habréis deducido, un firewall que cierre el acceso al puerto 135 basta para eliminar el riesgo de ser víctima de la vulnerabilidad, aún en un sistema sin parchear... pero este es uno de esos mágicos puertos que Windows deja abiertos.

Al mes de aparecer la vulnerabilidad, apareció, como era inevitable, un gusano de propagación masiva que causó estragos en la red de redes. La mayoría de los usuarios esperaban un gusano de correo, y solamente los usuarios con más conocimientos se dieron cuenta de las posibilidades que ofrecía esta vulnerabilidad para crear un gusano de red. Aunque el gusano en sí, conocido como Lovsan o Blaster, era una auténtica chapuza de la programación por muchos motivos que no vienen al caso, y en comparación con otros gusanos de red como el SQL Slammer era bastante menos efectivo; el campo de acción de la vulnerabilidad le proporcionó campo ancho para expandirse rápidamente, pues en casos como el del SQL Slammer sólo se veían afectados servidores SQL, mientras que RPC afecta a prácticamente cualquier sistema Windows NT conectado a Internet.

La actuación del virus consiste en **desbordar el buffer de la víctima y dejar a la escucha una shell de sistema en el puerto 4444**, y desde allí, mediante el protocolo de capa de aplicación TFTP (Trivial File Transfer Protocol) se transmite el propio código del virus, que se ejecuta y comienza a propagarse desde ese host. Por si el peligro del virus fuera poco, faltó tiempo para que la red se llenara de cazadores de shells buscando desesperadamente sistemas con el puerto 4444 abierto y a la escucha.

Ha pasado el tiempo y han aparecido nuevas vulnerabilidades de RCP/DCOM, entre ellas una Denegación de Servicio. Es un problema grave que aún resulta un quebradero de cabeza para mucha gente.

Adicionalmente, y por si todo lo comentado anteriormente sobre NetBIOS, SMB y RPC fuera poco, queda el tema de inicio de sesión remoto. Mediante RPC, cualquier sistema con el puerto 445 a la escucha (que Windows también deja abierto por defecto) se convierte en una **potencial víctima de inicio de sesión remoto**. Sólo es necesario un nombre de usuario y contraseña válido. La costumbre de mucha gente de dejar contraseñas vacías agrava este problema, pues existen determinados programas que automatizan las peticiones RPC y permiten un control remoto de escritorio a todos los niveles, como **Atalier Web Remote Commander**.

Deshabilitar NetBIOS bajo TCP/IP no elimina la posibilidad de su uso en sistemas NT, pues de no poderse acceder a los puertos 137, 138 ó 139; se gestionará la conexión mediante SMB (445). La solución es o bien **no compartir absolutamente nada de cara a Internet**, o bien **configurar los filtros adecuados** en el router o firewall (mañana veremos un ejemplo de esta configuración en un firewall).

En el caso de querer usar NetBIOS en Internet, debemos prestar especial atención a los **inicios de sesión nulos** (posibilidad de conexión sin necesidad de nombre de usuario o contraseña), puesto que una vez iniciada una sesión nula, es posible averiguar las contraseñas de cualquier otro usuario.

Así mismo, y debido a la instalación de software, Windows ejecuta en el inicio una gran cantidad de programas que pueden quedar potencialmente a la escucha de determinados puertos, disminuyendo el margen de seguridad en nuestra máquina. Lo mejor para estos casos es eliminar del inicio estos procesos molestos, desde el registro (*regedit.exe*) o desde el configurador de inicio de Windows (*msconfig.exe*).

Todos estos motivos son suficientes para darse cuenta de la importancia de los firewall (mañana se hablará de ellos) y conocer nuestro sistema en profundidad. Un sistema operativo recién instalado no es seguro ni muchísimo menos.

Ahora le toca el turno a **Linux y sus puertos abiertos por defecto**. La mayoría de estos puertos por defecto vienen dados por los servicios y daemons que se arrancan con el sistema operativo. Por ello yo recomiendo dejar los arranques lo más limpios posibles, e iniciar a mano los daemons o servicios que se necesiten cuando se requieren (por ejemplo, no iniciar SAMBA si no necesito acceder a ficheros de otro host en una red). Para limpiar el arranque, primero debemos conocer el runlevel de arranque de nuestro sistema, lo cual podemos consultar fácilmente en el fichero "*/etc/inittab*". Una vez conocido, accedemos al directorio "*/etc/rc.d/rcX.d/*" donde X representa nuestro runlevel, y allí eliminamos los scripts de arranque que no sean necesarios. En caso de necesitar arrancar algún daemon o servicio una vez iniciado el sistema, podemos ejecutarlo mediante la sintaxis "*/etc/rc.d/init.d/servicio start*".

El único gran fallo de Linux en cuanto a puertos abiertos por defecto es **el servidor de las X-windows**. Este servicio escucha por defecto el puerto 6000 y no es posible eliminar este servicio del inicio si queremos poder usar un interfaz gráfica, pero podemos solucionarlo de dos formas.

La primera es mediante la edición del fichero de configuración del servidor de las X añadiendo el flag “*--nolisten-tcp*”. La segunda es una manera más sencilla, mediante el uso de la propia herramienta de filtrado/firewall que incorpora el kernel de Linux: IPTables. Mediante comandos de consola, **con IPTables podemos asignar sencillas reglas de filtrado de datos en un firewall de nivel de red**. Concretamente, para filtrar el puerto 6000 tanto TCP como UDP, usaremos los comandos:

```
iptables -A INPUT -p tcp --dport 6000 -j DROP
iptables -A INPUT -p udp --dport 6000 -j DROP
```

Con esto los puertos pasarán a estado silent. Para evitar tener que ejecutar estos comandos cada vez que iniciemos el sistema, podemos incluir en el arranque del sistema (la carpeta de nuestro runlevel) un sencillo script con nuestro “firewall” que guardaremos con el nombre “*S99firewall.sh*” o cualquier otro nombre de script de arranque válido. Incluiremos en el fichero con cualquier editor de texto plano como vi...:

```
# Esto es mi firewall
iptables -A INPUT -p tcp --dport 6000 -j DROP
iptables -A INPUT -p udp --dport 6000 -j DROP
```

Y eso serviría para proteger nuestro servidor de las X de miradas indeseadas.

Otro problema en todos los sistemas operativos es el tema de los **paquetes ICMP que no queremos que nos lleguen**. Hoy día la mayoría de las posibles técnicas dañinas con paquetes ICMP están limitados, como por ejemplo el **ICMP redirect**, **ICMP router solicit**, **ICMP info req**, **ICMP addr mask** o **ICMP timestamp**. Excepto el redirect, cualquier sistema Windows hoy en día bloquea todos estos paquetes entrantes, y en Linux las últimas versiones del kernel se protegen de este tipo de ataques y otros similares. Para deshabilitar del sistema operativo Windows la recepción de paquetes ICMP redirect, debemos poner a 0 el valor de registro “*HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Parameters\EnableICMPRedirect*”. En Linux no es necesario pues el kernel lo deshabilita por defecto.

Pero **la función más común de los paquetes ICMP es el envío de pings** y su respuesta (pongs :-P). Un ping consiste en un paquete o conjunto de paquetes con una información de determinado tamaño encapsulada (normalmente suelen ser 32 ó 40 bytes) que se envían a un host, este los devuelve, y nos indica que el host está disponible así como su tiempo de respuesta. Bien, para complicar las cosas a los posibles curiosos para con nuestro sistema, la situación ideal es que NO respondamos a los pings que se nos hagan, de forma que el host que nos envíe el ping obtenga un “Tiempo de espera agotado” pues es posible incluso resetear ciertas conexiones dial-up mediante paquetes ping malformados.

Para bloquearlo en Windows, lo mejor es usar cualquier Firewall de nivel de red/transporte (la mayoría trabaja en red, transporte y aplicación) y **bloquear los paquetes con protocolo ICMP que lleguen a nuestro puerto EchoRequest (8)**. Con ello bastaría, aunque por supuesto necesitamos el firewall...

En Linux podemos realizar esta acción mediante un comando de consola, que podemos añadir al script de nuestro firewall realizado antes, de forma que quedaría...:

```
# Esto es mi firewall
iptables -A INPUT -p tcp --dport 6000 -j DROP
iptables -A INPUT -p udp --dport 6000 -j DROP
echo 1 >/proc/sys/net/ipv4/icmp_echo_ignore_all
```

Esta línea (*echo 1 >/proc/sys/net/ipv4/icmp\_echo\_ignore\_all*) pondría a “true” el valor de la variable de ignorar los ICMP echo para IPv4. Para poder aceptar los pings de nuevo, bastaría con usar el comando “*echo 0 >/proc/sys/net/ipv4/icmp\_echo\_ignore\_all*”. ¿Fácil, verdad? ;-)

## **- Breve introducción a la seguridad en redes locales**

Para esta pequeña introducción a seguridad en redes locales **asumiremos que hablamos de redes ethernet (IEEE 802.3)**, por tratarse éstas de los tipos de redes locales más comunes en el ámbito doméstico. Se trata de redes bastante rápidas, que resultan más funcionales en conexiones de corta distancia (la distancia máxima sin repetir la señal son 99 metros) y con un bajo número de equipos (a mayor número de equipos por segmento de red, aumentan las colisiones de paquetes).

Antes de nada, es conveniente distinguir los tres tipos de dispositivos de conexión que existen para estas redes: **repetidores/hubs, puertos/switches y routers o enrutadores**. También pueden conectarse dos equipos directamente de una tarjeta ethernet a otra, pero a efectos de red sería lo mismo que una conexión por hub. Es importante entender las diferencias entre los dispositivos:

Un **repetidor lo único que hace es regenerar una señal** (excepto en el caso de los hubs pasivos, que únicamente transmiten la señal, pero es un caso poco habitual) para que pueda extenderse más allá del límite de 99 metros del cable de par cruzado UTP. Un hub es un repetidor multipuerto, capaz de gestionar varias conexiones a la vez. **Tanto los repetidores como los hubs son dispositivos de capa 1** o capa física del modelo TCP/IP. Un hub toma una señal entrante y la reproduce por todos y cada uno de sus puertos, de forma que cada paquete viaja a todos los hosts, y es el host el encargado de decidir si se lo queda o no, mediante el análisis de la dirección MAC. Esto ya nos presenta un primer problema, pues alguien que ponga su tarjeta de red en modo promiscuo y un **sniffer** (software o hardware capaz de capturar y analizar paquetes de tráfico de red), se quedará todos los paquetes que le lleguen, independientemente de que el destino fuera él o no, con el consiguiente problema de seguridad que respresenta. Igualmente supone un problema de eficiencia en la red, por el tema de colisión de paquetes, así como la gran cantidad de paquetes de datos redundantes y la consiguiente pérdida de ancho de banda.

En parte para solucionar este problema se ideó el dispositivo que denominamos puente. Un switch es un puente multipuerto. Su cometido es **analizar el tráfico de forma que pueda ser redireccionado** al segmento de red que corresponda, de forma que un paquete destinado a un host de la subred A no pase por la subred B. Esto se realiza analizando el tráfico mediante las direcciones físicas (capa 2) y conmutando el tráfico según la necesidad. Es útil puesto que, en principio, elimina el riesgo de capturar alegremente el tráfico (ya veremos más adelante que no es así), además de **reducir el broadcast** y aumentar el aprovechamiento del ancho de banda. **Tanto los puentes como los switches son dispositivos de capa 2** o capa de enlace de datos del modelo TCP/IP.

Y por fin llegamos al más complejo dispositivo de red de este tipo: el enrutador o router. **Se trata de un dispositivo de capa 3** o capa de red del modelo TCP/IP, si bien un router es capaz de operar en capa 2 y capa 1 igualmente. El router **trabaja principalmente con direcciones lógicas (capa 3) y también con direcciones físicas (capa 2)**. Su cometido principal es analizar el tráfico entrante para **decidir cuál es el mejor camino para comunicar dos hosts y conmutar los paquetes** a la salida más adecuada. También elimina el broadcast de la red. Así mismo un router puede conectar distintos tipos de redes entre sí, sin importar su tecnología.

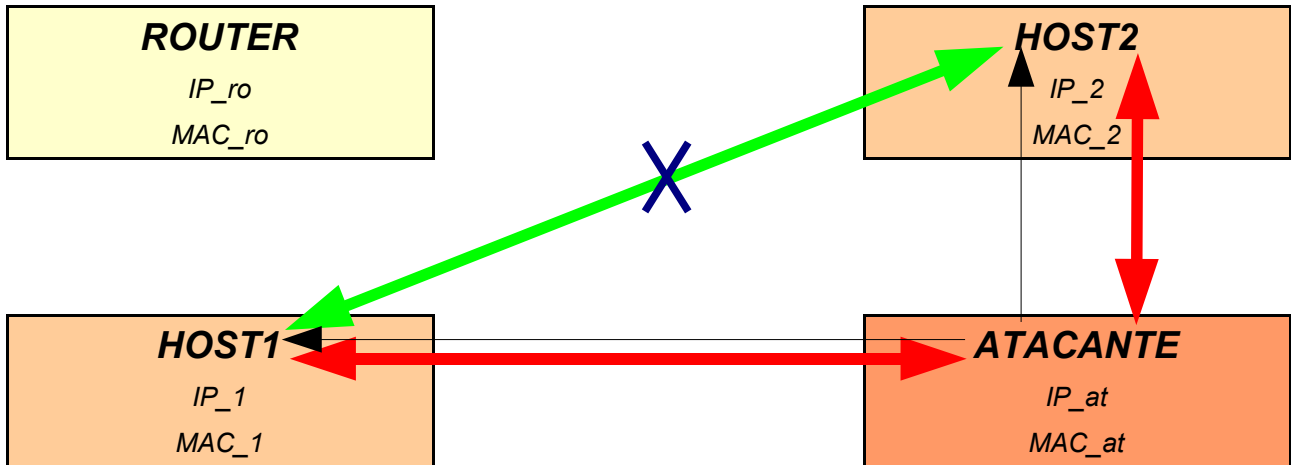
También es importante hablar de las **tarjetas de red**, dispositivos de capa 2 (enlace a datos) que **utilizan MAC**, aunque también trabajan en capa 1 (capa física) al convertir bits en señales mediante el **transceptor**.

Y ahora unas preguntas que nos estaremos haciendo todos... ¿Cómo relaciona un router la dirección lógica y la física? Bien, pues vamos a explicar y entender el protocolo **ARP (Address Resolution Protocol)**.

Cuando un host necesita saber la MAC de otro host, lo que hace es enviar un paquete **ARP request** a todos los equipos de la subred (broadcast) con la IP cuya MAC asociada quiere conocer. El equipo cuya IP coincide, envía al primer host un paquete **ARP reply** con su propia MAC. Obviamente, este trabajo no se realiza cada vez, pues supondría un malgasto de recursos de red. Así pues, **cada host posee una tabla ARP donde se guardan esas equivalencias IP-MAC**, que puede ser consultada mediante invocación desde shell con el comando "*arp -a*" (tanto en Linux como en Windows). Recomendando encarecidamente leer la ayuda del comando arp para saber más sobre sus posibilidades y sus argumentos ("*man arp*" en Linux y "*arp -help*" en Windows). Esta tabla se construye generalmente de forma dinámica, almacenando en caché las relaciones IP-MAC que obtiene el sistema operativo según las necesita.

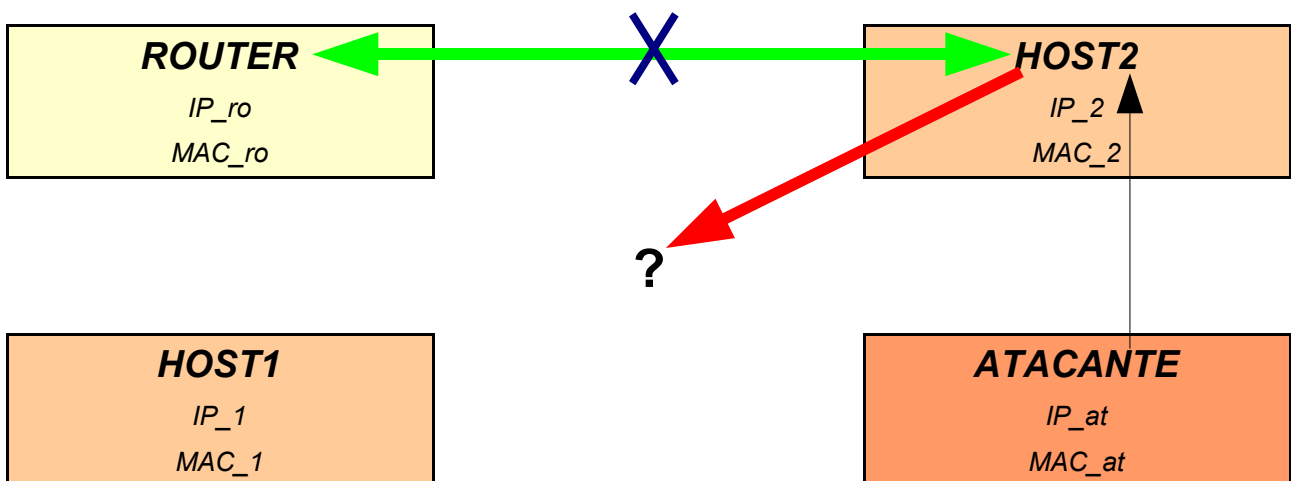
Esta operación supone una de las mayores vulnerabilidades en redes locales, que es explotada con mucha frecuencia. Un atacante malintencionado puede enviar paquetes ARP reply falsos de manera que altere la tabla de caché ARP del host al que dirige el paquete. Esta técnica se conoce como **envenenamiento de caché ARP o ARP Spoofing**, y vamos a entender qué peligros entraña y cómo evitarlos. Los ataques más comunes derivados del ARP Spoofing son dos tipos: los ataques **“man-in-the-middle”** y los ataques **DoS (Denial Of Service, Denegación de Servicio)**.

Ataque “man-in-the-middle”:



La flecha verde representa lo que sería el tráfico normal de datos en una comunicación HOST1-HOST2. Pero ATACANTE manda un paquete ARP reply falso (flecha negra) a HOST1 donde dice que IP<sub>2</sub> corresponde a MAC<sub>at</sub>, y después manda otro a HOST2 diciendo que IP<sub>1</sub> corresponde a MAC<sub>at</sub>. De esta forma, HOST1 y HOST2 creerán que están hablando entre ellos respectivamente, y dirigirán sus paquetes en consecuencia a la MAC que tienen almacenada en sus tabla ARP, que es MAC<sub>at</sub>. ATACANTE, por su parte, se encarga de redirigir los paquetes de un host a otro y, de paso, quedarse los y analizarlos todos con un sniffer, capturar sesiones de algún protocolo de aplicación (lo que se conoce como **Hijacking**), suplantar la IP de alguien (IP Spoofing), o cualquier otra trastada, que de ser nosotros HOST1 o HOST2, no nos interesaría que ocurriera. En el caso de que esta suplantación se realizara a un router y no a un host, ATACANTE obtendría todo el tráfico que pasa por la red. Este proceso es mucho más complejo que esto, pero el concepto está claro, ¿no?

Ataque DoS:



En el caso de un DoS, HOST2 quiere conectarse a Internet a través del router. ATACANTE manda un paquete ARP reply falso a HOST2 diciendo que IP\_ro se corresponde con MAC\_? (cualquier MAC que NO exista en la red). Con ello, logra que HOST2 dirija sus paquetes a una MAC que no existe, y por consiguiente, pierda la conexión a Internet por no poder comunicarse con el router. Esto es lo que se conoce como Denegación de Servicio.

Bien, entendido en qué consiste MiTM y DoS, hay que aclarar que según el sistema operativo, las tablas ARP se "vacían" al cabo de un tiempo, por lo cual el ataque perdería su efectividad, al renovarse la tabla con las MAC auténticas. Aún así, un atacante normalmente bombardea de forma periódica la red con paquetes ARP reply falsos, de forma que se asegure que las tablas ARP permanezcan envenenadas. En sistemas Linux, la tabla se renueva pasado un tiempo, al contrario que en los sistemas Windows, donde la tabla permanece intacta hasta que se reinicie.

¿Cómo evitamos este molesto problema en una red local? Bien, antes dije que las tablas se generan normalmente de forma dinámica... pues hay que generar unas tablas a mano y hacerlas estáticas.

**Una tabla de caché ARP estática supone introducir a mano las equivalencias IP-MAC** de toda nuestra subred, de forma que no puedan ser eliminadas ni modificadas (bueno, en realidad en ciertos sistemas Windows es posible envenenar un caché ARP aún con entradas estáticas. En sistemas Linux es absolutamente imposible). La gran ventaja es evitar este tipo de ataques de ARP Spoofing, pues dejan de ser operativos los paquetes ARP reply (en realidad pueden recibirse, pero sólo para crear entradas nuevas, no para modificar las existentes). Algunas desventajas se hacen manifiestas en redes grandes, pues es necesario configurar a mano las tablas ARP de todos y cada uno de los hosts, incluyendo el router. Cuantos más hosts, más configuraciones y más entradas en cada tabla. Para introducir entradas estáticas en la tabla de caché ARP se usa la sintaxis "*arp -s IP MAC*" (tanto en sistemas Linux como Windows).

En grandes redes, para evitar mantener estas tablas a mano, podemos usar herramientas específicas para **detectar un flujo demasiado alto de paquetes ARP reply, así como los "ACK storm"**. Igualmente, ciertos programas y firewalls detectan los cambios en las tablas ARP y avisan al administrador de la red mediante un correo... Aún así lo más efectivo sigue siendo una tabla ARP estática.

Existen otros problemas en redes locales derivados de la arquitectura TCP/IP como los DoS por tormentas de ICMP. Básicamente, un atacante envía un paquete ICMP echo request malformado a una dirección de broadcast, de forma que el paquete se distribuye y todos los hosts de la red responden el ping a la IP indicada, que obviamente no es la del atacante, sino la de alguien a quien no quiera mucho y que ha falseado en el paquete ICMP echo request. Este tipo de ataques puede generar un tráfico enorme y hacer caer al host atacado, **provocando un DoS especial que se conoce como smurfing**. También hay otras técnicas con paquetes ICMP como **ICMP sweep** (barridos de pings a rangos de direcciones), **ICMP redirect** o **ICMP addr mask**.

Existen así mismo algunos fallos que pueden ser explotados maliciosamente en otros protocolos de TCP/IP, pero como en el caso de ICMP, la mayoría se resuelven con un control en la máquina local (estos problemas de ICMP se pueden resolver fácilmente mediante algunas sencillas reglas en el firewall), o mejor aún, imponiendo determinados filtros en los routers de la red (la mayoría de los routers hoy día los incluyen de fábrica).

Aunque existen muchos problemas más, sólo se pretendía introducir brevemente los conceptos de redes locales y tipos de dispositivos de enrutamiento, así como tratar algunos problemas comunes (con especial atención al envenenamiento de caché ARP).

## 2. Seguridad en Internet

### - Introducción: Panorama actual en Internet

Internet se ha convertido en el símbolo del progreso tecnológico del ser humano, e igualmente en símbolo de su decadencia. Hoy en día Internet es tierra de nadie y tierra libre al mismo tiempo. Para lo bueno y para lo malo.

Desde que el 2 de Noviembre de 1988 naciera el primer virus de Internet (el llamado "Gusano de Internet de 1988") ha llovido mucho, no cabe duda. Antes las redes se diseñaban para que fueran funcionales, nadie pensaba en la seguridad, porque nadie tenía un motivo para violarla. Hoy, ya sea por dinero, por fama o por el simple ego, hay mucha gente dispuesta a violarla. Así pues, han evolucionado las redes. El concepto de virus y antivirus nos es muy familiar a todos los Internautas, e incluso en los últimos tiempos hemos asistido por televisión a detenciones espectaculares de creadores de virus, piratas informáticos, crackers... Y también hemos visto cómo la televisión ha prostituido el término "hacker" que antes era sinónimo de honor y ahora es sinónimo de delincuencia.

Aún así, hay muchas personas que aún se preocupan por la seguridad en Internet, expertos en seguridad, hackers, usuarios... una gran comunidad que se informa e intenta informar a los que le rodean. El conocimiento, como siempre digo, es donde reside la verdadera libertad, y en Internet si queremos ser libres, tenemos que poseer ese ansia de conocimiento, tenemos que saber y conocer cómo funcionan las cosas, de qué manera podemos protegernos. Hay gente que me dice "¿Para qué quiero un firewall si no tengo nada de interés en mi ordenador, sólo mis correos personales y las fotos de mi último viaje a Benidorm?". La respuesta es sencilla: son TUS correos y son TUS fotos. Es tu derecho protegerlos, pero así mismo es tu deber.

Si habéis visto la película "Piratas de Silicon Valley" o habéis leído sobre los inicios de la informática doméstica, habréis visto una muestra de esta lucha, de cómo empezó esto. Las grandes empresas del momento le preguntaban a Steve Wozniak para qué querría una persona cualquiera un Apple en su casa. Resultaba irrisorio pensar en un oficinista con un ordenador en su casa. Hoy en día trabajamos, jugamos, compramos, hablamos... todo desde casa, gracias a ese aparato. El espíritu de la libertad tecnológica ya no vive en Apple, la blanca y negra bandera pirata, símbolo de rebelión y libertad, no ondea en sus oficinas centrales desde hace mucho tiempo, pero ese espíritu aún vive, y ese afán de acercar a todos lo que las grandes corporaciones creen sólo suyo sigue vivo en Internet, alimentado por la comunidad hacker. ARPAnet evolucionó en Internet, el NCP dio lugar a TCP/IP, los viejos Apple dieron lugar a los x86, los módem de 28.8 Kbps dieron lugar a la fibra óptica y el wireless... y nosotros hemos de evolucionar de forma pareja con la tecnología. La seguridad en Internet es tan necesaria como la propia Internet, y es responsabilidad de nosotros mantener vivo el espíritu hacker en base a la seguridad.

Y vamos a aprender a protegernos.

**Death Master**

## - Antivirus

Hoy en día, por desgracia, no hay nadie que no haya oído hablar de los virus informáticos. El mayor problema quizá sea la falta de información o la gran cantidad de información incorrecta que circula dentro y fuera de la red. Prueba clara de esto que digo son los denominados "**hoaxes**", que son bulos -generalmente relacionados con los virus- que circulan en forma de cadenas de correo electrónico (¿Soy el único que odia a muerte las cadenas :-m ?). Supongo que todos hemos recibido alguna vez (muchas más de las que desearíamos) el típico correo informando de un nuevo virus que solo con abrir el correo puede provocar que se te caigan los libros de las estanterías o algo por el estilo. Otra fuente importante de errores es la infame caja tonta. He llegado a oír ciertas cosas que le hacen a uno pensar en qué creará la gente "no informatizada" de lo que es Internet. De hecho nunca olvidaré cuando, hace unos años, mi abuela me llamó alarmada para pedirme que tuviera cuidado con el ordenador porque acababa de ver en televisión una alerta del virus "**ILoveU**" y claro, la pobre mujer estaba muy preocupada no fuera a ser que yo cogiera algo... :-p

### **Introducción a los virus: tipos y precauciones**

Antes de nada, vamos a ver algunas definiciones de virus: "*pedazo o secuencia de código ejecutable que se caracteriza por ser capaz de clonarse a sí mismo*"; "*programa que hace una copia de sí mismo sin la aprobación del usuario*"; "*código o script capaz de autoreplicarse sin el consentimiento del usuario*". Parece que en algo estamos de acuerdo: un virus se replica y actúa sin el consentimiento del usuario del ordenador. Con esta definición englobamos muchos tipos de virus, y conviene echar un vistazo a cada tipo para conocer sus peculiaridades y peligros. Conviene recordar que en esta sección, nos vamos a referir únicamente a virus y antivirus para sistemas Windows, pues en Linux apenas existen virus y por la propia concepción del sistema operativo, es innecesario el uso de antivirus (aunque existen). Por la propia evolución de los virus, pareja a la informática, hay muchas clasificaciones clásicas de manual que están obsoletas, por ejemplo la que divide los virus en las clases **caballo de troya, polimorfos, sigilosos, lentos, retro-virus, multipartitos, blindados, de compañía, voraces y gusanos**. Esta clasificación está sacada de un libro de seguridad informática de hace seis años (1998). Como podéis imaginaros, esta clasificación se ha quedado obsoleta, hay nuevos tipos, algunos de los mencionados están en desuso... en definitiva, que voy a usar otra clasificación, pero es importante comprender que no es nada estricto y cerrado, y podéis encontrar muchas otras clasificaciones, además de que ésta se quedará obsoleta igualmente...

**Código ActiveX malicioso:** Se trata de objetos incrustados en páginas web que se ejecutan de forma automática al visitar el sitio. Normalmente se asocian a vulnerabilidades del software navegador, usualmente Internet Explorer (por su uso extendido y por su enorme catálogo de vulnerabilidades).

**Código Java malicioso:** Es similar al caso anterior, solo que se trata de objetos programados en lenguaje Java. Hoy día, con el sector de telefonía móvil capaz de interpretar Java en alza, es un peligro latente que tarde o temprano comenzará a dejarse notar.

**Virus de macro:** Similar a los casos anteriores, si bien el objeto se encuentra incrustado en algún tipo de documento perteneciente a un software capaz de interpretar scripts de macro. Típicamente es el caso de Microsoft Office y sus virus de macro.

**Virus del sector de arranque:** Hoy en día son casi una reliquia de museo, aunque hace unos años suponían el mayor porcentaje de los programas maliciosos. Se trata de virus que infectan el sector de arranque o la tabla de particiones del disco duro. Se eliminan con bastante facilidad.

**Virus de fichero de acción directa:** Se trata del tipo más famoso de virus. Es aquel que ejecuta archivos .exe o .com y se replica desde la memoria a todos los programas que se ejecuten, sobrescribiendo parte del código original. Generalmente tienen algún objetivo adicional, como formatear el disco duro, mostrar un mensaje en una fecha prefijada...

**Virus de script:** Se trata de los virus más extendidos hoy en día junto a los gusanos. Hoy día, gracias principalmente a los lenguajes VisualBasicScript y JavaScript, cualquier usuario con conocimientos bajos en programación puede generar un virus en minutos. Además, debido al Scripting Host de Windows, se pueden ejecutar con doble click, como si fueran un archivo ejecutable.

**Gusano:** El otro gran tipo de virus de hoy día. Se trata de un programa que utilizan copias completas de sí mismo para infectar a otras máquinas y propagarse, dejando una copia de su código o parte de él en la otra máquina y empezando de nuevo el proceso desde ella. Hasta hace poco eran casi exclusivos los gusanos de correo, si bien hoy en día, debido a virus como SQLslammer o Lovesan, los gusanos de red se han hecho tristemente famosos.

**Troyano:** Éste es un tipo especial de virus. Un troyano es una herramienta de control remoto que alguien maneja (obviamente no es el dueño de la máquina infectada). Se compone de dos módulos: cliente y servidor. El servidor es el módulo que infecta, y generalmente suele ir escondido en archivos ejecutables o en correo electrónico, siendo típico el uso de ingeniería social para lograr su ejecución. El cliente es el módulo de control mediante el cual un atacante puede obtener control de la máquina. Los troyanos no se reproducen ni infectan otros programas, pero debido a sus peculiaridades, se consideran igualmente virus.

Esta sería la clasificación general de virus que considero más correcta dado el panorama actual.

Hay algunos conceptos que conviene definir respecto al tema de virus. En primer lugar, antes he hablado de ingeniería social. **Por ingeniería social entendemos el uso de técnicas psicológicas para convencer a un usuario** de que realice alguna acción que no debería o querría realizar en circunstancias normales. Claros ejemplos sería obtener contraseñas con alguna artimaña, obtener respuestas de preguntas secretas mediante conversaciones aparentemente casuales, lograr que un usuario ejecute algún programa que no debería... no tiene nada que ver con los virus pero conviene conocer el concepto, pues en el tema de troyanos es el pilar principal.

Otro concepto importante a comprender es el concepto de payload. El término payload viene del inglés *carga*, y simboliza la "carga peligrosa de un virus". A efectos prácticos, **los payloads son las características y efectos dañinos de un virus**. Por ejemplo, un payload podría ser "Formatear el disco duro", "Mostrar por pantalla un mensaje el día X"...

Conviene conocer la nomenclatura de los virus. Un virus se denomina de la siguiente manera: **TIPO/Nombre.versión**. Por ejemplo, W32/Mimail.J indica que se trata de un gusano -en este caso de correo- (W32), cuyo nombre es Mimail y que éste en concreto se trata de la versión J (alfabéticamente). Los nombres suelen proceder de algún payload o del código del virus. A veces, distintas compañías de seguridad denominan a los virus de distinta forma, como en el caso del gusano Blaster o Lovsan. Veamos ahora cómo se clasifica un virus de forma detallada. Toda esta información ha sido extraída del, para mí, mejor portal antivirus de habla hispana: Video Soft Antivirus (<http://www.vsantivirus.com>) con sede en Maldonado, Uruguay.

### **W32/Mimail.J. Asunto: "IMPORTANT"**

<http://www.vsantivirus.com/mimail-j.htm>

**Nombre:** W32/Mimail.J

**Tipo:** Gusano de Internet, troyano robador de información

**Alias:** Mimail.J, W32.Mimail.J@mm, W32/Mimail.J.worm, W32/Mimail.J, W32/Mimail.J-mm, WORM\_MIMAIL.J

**Fecha:** 17/nov/03

**Plataforma:** Windows 32-bit

**Tamaño:** 13,856 bytes

\* Herramienta para quitar el W32/Mimail de un sistema infectado

\* Sugerencias para administradores (bloqueo antispam)

Variante de W32/Mimail.I, que se propaga masivamente por correo electrónico, en un mensaje anunciando la expiración de la cuenta del usuario en Paypal, una conocida plataforma de pago vía Internet. Al ejecutarse el adjunto, el gusano intenta robar la información de la tarjeta de crédito del usuario.

[...]

Observamos un título, el nombre, tipo de virus, alias, fecha, plataforma, tamaño, herramientas de desinfección y una extensa información detallada sobre desinfección manual, payloads y demás datos de interés (que he recortado por su enorme extensión).

Las precauciones básicas para evitar infecciones por parte de virus son por lo general las que nuestro propio sentido común nos dicta. Vamos a ver en primer lugar las “**Pautas generales para mantenerse alejado de los virus**” de VSAntivirus y después hablaremos en términos coloquiales...

*1) Use regularmente un programa antivirus (nosotros siempre recomendamos no confiar en uno solo, pero usar más de uno no significa que debamos tenerlos a todos instalados, simplemente ejecutamos esos antivirus en su opción de escaneo, sobre la carpeta que contenga los archivos a revisar). Y por supuesto, de nada vale usar algún antivirus si no lo mantenemos actualizado con los upgrades, updates o add-ons correspondientes. Actualmente, las actualizaciones son diarias (AVP, Panda y otros) o al menos semanales. No existen los "virus demasiados nuevos y sin antídotos", la reacción de las casas de antivirus es inmediata en todos los casos. Pero mejor pregúntese, si la suya también lo es a la hora de actualizarse.*

*2) No abrir ningún mensaje ni archivo recibido a través del correo electrónico de fuentes desconocidas o muy poco conocidas. En el caso de personas conocidas, se deben igualmente tomar las precauciones correspondientes. Asegurarse con esa persona del envío ("Melissa" y otros pueden ser enviados por conocidos que ignoran estar mandando el virus en sus mensajes), y nunca ejecutarlos, sino guardarlos en una carpeta temporal y pasarle a esa carpeta dos o tres antivirus actualizados antes de tomar la opción de ejecutarlos (.EXE) o abrirlos (.DOC, .RTF, etc.). Pero ante cualquier duda, simplemente se debe optar por borrar el mensaje (y archivos adjuntos). Como se dice vulgarmente, "la confianza mata al hombre", en este caso al PC.*

*3) Estar informado de cómo operan los virus, y de las novedades sobre estos, alertas y anuncios críticos, en sitios como el nuestro.*

*4) No bajar nada de sitios Web de los que no tenga referencias de seriedad, o que no sean medianamente conocidos. Y si se bajan archivos, proceder como los archivos adjuntos. Copiarlos a una carpeta y revisarlos con dos o tres antivirus actualizados antes de optar por ejecutarlos o abrirlos.*

A grandes rasgos se trata simplemente de sentido común: no es normal que tu tío del pueblo te mande un email redactado en inglés (cuando él no sabe hablarlo) donde te dice que te manda un vídeo de Britney Spears en un archivo .exe. ;-)

Es importante usar el antivirus, y desconfiar por sistema de todo aquello que resulte sospechoso, tener claro que nadie da duros a pesetas, y tener siempre los ojos abiertos.

## Uso de Antivirus: cómo usarlos y cómo actuar ante un virus

Hoy día en sistemas Windows es imprescindible contar con un buen software antivirus. Los principales representantes de software antivirus pertenecen a las compañías más famosas: **Symantec** (<http://www.symantec.com/>), **McAfee** (<http://www.mcafee.com/>), **Panda** (<http://www.pandasoftware.com/>)... si bien también hay nuevos antivirus muy capaces, como **Kaspersky Antivirus** (<http://www.kaspersky.com/>), **NOD32** (<http://www.nod32-es.com/>), **F-prot Antivirus** (<http://www.f-prot.com/>), **Dr Web Antivirus** (<http://www.drweb.sk/>)... Y hay, en último lugar, gran cantidad de software antivirus gratis. No hablamos de versiones shareware, como Panda, sino de versiones completas (a veces reducidas) totalmente gratuitas. Tenemos **AVG Free Antivirus** ([http://www.grisoft.com/us/us\\_dwnl\\_free.php](http://www.grisoft.com/us/us_dwnl_free.php)), **TrendMicro Antivirus** (<http://www.trendmicro.com/en/products/desktop/housecall/more/download.htm>), **Bit Defender Antivirus** ([http://www.bitdefender.com/bd/site/products.php?p\\_id=24](http://www.bitdefender.com/bd/site/products.php?p_id=24)), y otros tantos.

Hay que pensar en el **software antivirus** como un desembolso totalmente necesario según están las cosas hoy en día. ¿Cuál elegir de los citados anteriormente? Cualquiera sirve. ¿Qué buscamos en un antivirus? Lo primero que debemos mirar es que posea un soporte poderoso, con **actualizaciones diarias de su base de datos**. Con un ritmo de aparición de virus de 3 a 10 virus diarios (los que estén apuntados a la lista de correo de VSAntivirus saben de qué hablo), es imprescindible tener un respaldo de actualizaciones. Así mismo, es importante un **soporte técnico** al que podamos remitir ficheros en caso e infección por un virus desconocido. También es importante que este software **consume el menor número de recursos** de procesador y memoria posibles en modo de protección permanente. Es interesante que el antivirus **analice en tiempo real los emails** cuando nuestro cliente de correo recoge los nuevos mensajes por POP3 o IMAP, y sobre todo cuando se envían por SMTP, evitando en gran medida problemas con gusanos de correo.

Hoy día es común la aparición de “mini-antivirus” específicos para ciertos tipos de virus. Cuando algún virus concreto tiene una incidencia o peligrosidad alta, las compañías antivirus crean pequeños programas especializados en su detección, eliminación, y si es necesario, el parcheo de vulnerabilidades asociadas. Estos programas son gratuitos para todos los Internautas, y este verano hemos tenido ejemplos muy notorios de este tipo de programas para virus como Lovsan o Sobig.

Al principio hablamos de troyanos, y pusimos de manifiesto las grandes diferencias con el resto de los virus, no obstante de ser considerados programas dañinos. Aunque cualquier antivirus del mercado reconoce en sus bases de datos los troyanos, existen programas específicos encargados de detección, eliminación y protección contra troyanos y backdoors (del término inglés “puerta trasera”). Este software se denomina **antitroyanos**. El más conocido es **The Cleaner** (<http://www.moosoft.com/products/cleaner/>). En mi opinión es el mejor, pues no sólo protege al sistema de troyanos mediante protección permanente, sino que incluye una utilidad llamada TCmonitor que se encarga de escuchar el registro de Windows y lanzar una alerta cada vez que se modifique algo (en instalaciones es normal, pero si te bajas un flash y suena la alarma, algo no va bien). Igualmente existen otros programas antitroyanos muy conocidos, como **Anti-Trojan** (<http://www.anti-trojan.net/>) o **Trojan Remover** (<http://www.simplysup.com/tremover/>). En general el funcionamiento de los antitroyanos es muy similar al de los antivirus.

Es hora de saber **cómo funciona un antivirus** y cómo debemos usarlo, pues cada tipo de usuarios tiene unas necesidades distintas, y de forma diferente deben cubrirse. Lo primero a tener en cuenta es **la protección permanente**, que con distintos nombres, todos los antivirus incluyen. Esta protección genera un servicio del antivirus que está permanentemente escuchando la memoria y los datos que entran y salen de nuestro ordenador, con el fin de detectar al instante cualquier evento potencialmente peligroso. Es una opción que recomiendo encarecidamente a todas aquellas personas no muy familiarizadas con la seguridad informática, despistados, niños pequeños que usan el ordenador... y en general cualquier usuario que pueda suponer un peligro en este sentido. Los usuarios más avanzados y con más control sobre su máquina preferirán desactivar esta protección para comprobar a mano los archivos que consideren sospechosos, con el consiguiente ahorro de recursos del sistema.

Respecto a las opciones de escaneo, prácticamente todos los antivirus comparten las mismas opciones: comprobación de todo el sistema, de la memoria, por unidades, por carpetas y por archivos. Es bueno realizar periódicamente revisiones completas del sistema, o incluso programarlas. También es importante revisar una unidad cuando sospechamos que puede contener algún tipo de programa dañino (Típicamente disquetes y CD's prestados). Casi todos los antivirus integran así mismo opciones de menú contextual, de manera que para escanear un fichero sólo es necesario realizar click en el botón secundario del ratón y elegir la opción adecuada.

Cuando un antivirus detecta un virus, normalmente pregunta al usuario sobre el curso de acciones a tomar (si es posible configurarlo, recomendando elegir esta opción). Normalmente encontraremos las opciones: **Eliminar, Cuarentena, Enviar, Ignorar**. Cuando el fichero no es importante o no sabemos de dónde ha salido, lo mejor es eliminarlo directamente (aún así la mayoría de los antivirus generan copias de seguridad de forma automática, que también conviene borrar). Cuarentena es una buena opción para ficheros “sospechosos” que hemos obtenido por nosotros mismos pero cuyo contenido no es claro. Los antivirus **hoy día son capaces de reconocer virus que no se encuentran en su base de datos en base al comportamiento** de estos, pero a veces fallan. Yo me he encontrado casos donde un software perfectamente normal y legal era detectado como “posible virus”, y borrarlo no era la mejor opción. Cuando se trata de archivos muy extraños, dudosos o con información importante, es mejor enviarlo a la compañía antivirus contratada para que lo analicen profesionales. Ignorar es útil cuando queremos que el antivirus no realice ninguna acción.

Por último hemos de hablar del **log o registro**. Cualquier software de este tipo mantiene un registro donde quedan reflejadas las acciones realizadas y los eventos importantes, tales como escaneos y sus resultados, alertas de virus, acciones realizadas por el usuario o por el software en caso de haber sido programado de forma automática...

## Virus e integridad en la máquina local

Antes de nada, hay que definir eso de "Integridad" de una máquina. **Entendemos por integridad de un sistema la inalteración de los recursos del mismo:** ficheros, configuraciones, elementos de seguridad, información contenida en el registro... Cualquier acción que pueda comprometer un sistema, como borrar o modificar un fichero importante del sistema operativo, eliminar o introducir entradas en el registro sin consentimiento del usuario, cambio de configuraciones, eliminación o alteración de los objetos de seguridad, o crear la posibilidad de que cualquier persona de forma remota ejecute alguna de las acciones anteriores... se considera una alteración en la integridad del sistema.

A partir de esta definición, podemos entender que prácticamente cualquier virus compromete la integridad local. Normalmente cuando detectamos un virus y queremos saber en qué medida ha resultado dañada la integridad del sistema, debemos acudir inmediatamente a algún buscador (yo recomiendo San Google) para buscar información relativa al virus, y poder **consultar una detallada lista de sus payloads**. Normalmente podremos corregir a mano los cambios efectuados, si el antivirus no lo hace de forma automática y no existe una herramienta de desinfección específica. Los peores casos de compromiso de integridad del sistema se encuentran quizá en los rootkits. **Entendemos por rootkit una herramienta o conjunto de herramientas diseñada específicamente para romper por completo la integridad del sistema**, obteniendo cuentas de administrador ilícitas, creando puertas traseras, cambiando funcionalidades del sistema operativo para que las acciones no se detecten... Por suerte suelen ser detectados y bloqueados en ejecución, pero mucho cuidado cuando no se tiene activada la protección permanente.

## **- Firewalls e IDS**

Como ya hemos dicho, prácticamente todo el mundo hoy día está familiarizado con el concepto antivirus. Pero por desgracia no es así con los **firewalls o cortafuegos** y con los **IDS (Intrusion Detection System)**. Y otro gran sector que “ha oído hablar” de ellos piensan que son programas para “proteger de los virus” o “proteger de los hackers” o algo similar. Para protegernos de los virus están los antivirus, y de los auténticos hackers no hay necesidad de defenderse. ;-)

Pero sí hay necesidad de defendernos de muchas otras cosas, y este tipo de software nos ayuda a estar al tanto de lo que pasa en nuestro PC, y ser nosotros mismos los que llevemos la seguridad del sistema, con control total sobre el mismo.

### **Conceptos básicos sobre Firewalls e IDS**

Antes de nada, debemos dejar claros ciertos conceptos, como definiciones y tipos de firewalls e IDS; y también hablaremos brevemente de la utilidad y necesidad de usar este tipo de software. Este tema podría alargarse días y días, pues hay mucho que decir sobre él, así que trataré de resumir todo, orientándolo a una seguridad práctica a nivel usuario.

**Un firewall o cortafuegos es un filtro (ya sea software, hardware o una combinación de estos) que analiza y filtra el tráfico** entre hosts, segmentos de red o redes. Este control de información no es únicamente desde fuera hacia dentro como habitualmente se piensa, es bidireccional. Tan malo es que entre determinada información por un puerto que no debería estar abierto, como que un programa malicioso instalado en nuestro sistema (por ejemplo un troyano) mande paquetes con información sensible hacia fuera. Esta definición es muy general, lo sé, y dentro de ella pueden abarcarse desde la mayoría de los routers o enrutadores (ya entenderemos esto cuando hablemos de los tipos de firewalls) hasta los grandes firewalls con hardware especializado y miles de euros de coste, pasando por los típicos firewalls basados en software que se instalan bajo un sistema operativo.

Hay muchas clasificaciones de firewalls, dividiéndolos bajo criterios muy complejos. No quiero entrar a un nivel de detalle tan profundo, por lo que dividiremos los firewalls según el nivel al que se realice el filtrado (aquí es útil haber entendido bien la introducción a TCP/IP de ayer).

En primer lugar encontramos los **Paquet Filtering Firewalls o Firewalls de Filtrado de Paquetes**. Estos firewalls se basan en un **filtrado de paquetes según las cabeceras de los mismos**. No tienen el concepto de sesión, y cada paquete individual es analizado, estudiado, y aceptado o denegado según su naturaleza. **Trabaja principalmente en la capa de red**, aunque bajo determinadas circunstancias también lo hace en la de transporte del modelo TCP/IP, y aunque en un principio eran los firewalls más populares, hoy en día están en desuso. Este filtrado tan básico no contemplaba conceptos como la autenticación, por lo que **es vulnerable a ataques de Denegación de Servicio y Spoofing**. Como desventaja añadida, son extremadamente **complicados de administrar en redes complejas** y cuanto mayor sea el tráfico de la red, más es entorpecido por el firewall, produciéndose mayor número de colisiones de paquetes. La mayoría de los **pequeños dispositivos de red y routers pueden realizar la función de firewall de filtrado de paquetes**.

El segundo tipo de firewalls que podemos distinguir son los **Stateful Application Inspection firewalls o Firewalls de Inspección de Estado de Aplicación**. Se conocen familiarmente como firewalls a nivel de red. Estos firewalls **trabajan en multitud de capas del modelo TCP/IP** (enlace de datos, red, transporte y parte de la capa de aplicación, concretamente la capa de sesión), con el consiguiente aumento en la complejidad de los mismos. Estos firewalls **reconocen el tipo de tráfico y filtran según las necesidades específicas de cada protocolo**, mediante un módulo de control de sesiones, conexiones y su contexto que trabaja en las capas de red y enlace a datos. Controlan el inicio de sesiones, filtrando el tráfico por protocolo y puerto, y reconocen el fin de las mismas. Sus principales ventajas son el **alto rendimiento** derivado de la operabilidad a nivel de sesión en lugar de aplicación, así como el poco consumo de ancho de banda que supone. Por contra, no revisa la trama del paquete y **una vez establecida la conexión, puede entrar a través de ella todo tipo de tráfico** sin que el firewall lo analice.

El último tipo que vamos a distinguir es el **Application Gateway Firewall o Firewall de Puerta de Enlace de Aplicación**. Son los que coloquialmente se conocen como firewalls a nivel de aplicación. Estos firewalls **trabajan principalmente en todos los niveles de la capa de aplicación**, si bien también trabajan en las capas de transporte, de red y de enlace a datos. Basan su trabajo en **reconocer y filtrar el tráfico que es solicitado por aplicaciones (como un navegador web) o protocolos (FTP, HTTP...)**, así como **reconocer las sesiones y la autenticación de usuarios**. Suponen los firewalls de más alto nivel, pero también son los **más lentos en cuanto a consumo de recursos del sistema**, y pueden provocar problemas de **incompatibilidad con ciertos protocolos** con los que tengan imposibilidad de trabajar.

Hay que tener en cuenta que esta división sería en casos ideales, puesto que en la práctica cualquier firewall doméstico (no hablamos de los firewalls con hardware específico como un PIX de CISCO) son **híbridos entre el firewall de red y firewalls de aplicación**. Estos firewalls domésticos reconocen el uso de red de determinadas aplicaciones, así como el establecimiento de sesiones, uso de determinados protocolos independientes de la aplicación (ICMP, IGMP...), etc. Son quizá el tipo más completo de firewall.

Ejemplos de firewalls populares para sistemas Windows serían: **Kerio Personal Firewall, Agnitum Outpost Firewall, Sygate Personal Firewall, McAfee Desktop Firewall, Norton Firewall, BlackICE PC Protection, Freedom Firewall, Zone Alarm...** Para Linux lo más común es el uso de la **herramienta de filtrado de Netfilter conocida como IPTables**, aunque también existen firewalls (basados o no en IPTables) de aspecto y e interfaz de uso similar a los de Windows, como **KMyFirewall, Firewall Builder, Shorewall, Guarddog, Firestarter...** La gran mayoría de ellos se basan en iptables en menor o mayor medida, desde los generadores automatizados de scripts hasta las completas interfaces gráficas.

Es hora de hablar de **IDS (Intrusion Detection System)**. Un IDS es un **sistema de alerta en tiempo real** que, basándose normalmente en una base de datos, **es capaz de identificar y reconocer algún patrón definido como potencialmente peligroso**. Estos eventos no sólo se refieren a intentos de conexión, sino que son capaces de reconocer cualquier evento dentro del host o red (por ejemplo si tenemos un servidor de telnet en nuestro sistema y queremos permitir el uso de comandos como vi, man, less, ls, cd... pero no queremos permitir que se pueda usar gcc o gdb). Podemos decir que un IDS es al tráfico de red lo que un antivirus al análisis de los archivos almacenados en disco.

Mientras un firewall solamente filtra y genera registros (logs), un IDS vigila y cuando reconoce un patrón, actúa según haya sido programado (no siempre filtrando), además de generar alertas en tiempo real por pantalla, por correo electrónico... Muchos se estarán preguntando si estos dos conceptos no son compatibles... y efectivamente lo son. **Cada vez es más común que los firewalls incluyan funcionalidades de IDS**, como ya hacen **Agnitum Outpost Firewall, Kerio Personal Firewall, BlackICE PC Protection...** y bastantes otros. Aún así, las características IDS que incluye un firewall son bastante limitadas, siendo preferible el uso de software IDS especializado, como **SNORT, CIDS (Cerberus Intrusion Detection System), MIDAS (Monitoring Intrusion Detection Administration System)...**

La combinación más usual es: **un buen IDS** (Snort suele ser la opción elegida) con un sistema de **base de datos** (normalmente MySQL), **generación web con estadísticas** (PHP), herramienta de **control por consola** (ACID, Analysis Console for Intrusion Databases) y **servidor web** (Apache). El IDS supone el núcleo central, la base de datos controla tanto la programación del IDS como el registro de logs, PHP genera webs que Apache sirve y que podemos consultar en tiempo real mediante cualquier navegador. ACID ayuda a la programación y orquestación de todos los elementos. La única desventaja es que la instalación y correcta configuración de **Snort + MySQL + PHP + ACID + Apache** es bastante tediosa y compleja.

Ahora vamos a ver los tipos de IDS que hay atendiendo al tipo de sistema al que están orientados. La clasificación de IDS es descendente, pues en el orden en que los voy a citar, cada tipo es englobado por los siguientes, y a su vez éstos son particularidades de sus precedentes. Cualquier IDS, debido a su naturaleza programable, puede funcionar como cualquier tipo de IDS, aunque existen excepciones.

En primer lugar, distinguimos los **HIDS (Host Intrusion Detection System)**, que son los que conocemos como IDS simplemente, pues están **orientados a un único host**, se instala y actúa en una sola máquina. Son los IDS que todos conocemos y los más comunes. Los HIDS, por contra de los otros tipos, consumen bastante pocos recursos de sistema.

En segundo lugar, un tipo particular de IDS es el **NIDS (Network Intrusion Deteccion System)**, orientados a la monitorización de redes completas y VPNs (Virtual Private Networks). Se instala en una máquina pero actúa en un grupo de ellas. Normalmente además del uso de una base de datos, se apoyan en la búsqueda de patrones en los paquetes que pasan por la red (similar al firewall de filtrado de paquetes).

Por último, un tipo especial de NIDS son los **DIDS (Distributed Intrusion Detection System)**, que son básicamente un tipo de NIDS estructurados de forma distribuida, esto es, con diversos módulos que operan desde distintas máquinas. Se instala en diversas máquinas con un control centralizado. Esto nos otorga la poderosa ventaja de que el sistema DIDS no cae en caso de ataque a una de la máquinas que lo sustentan, si bien un potencial atacante que conozca el sistema procurará lanzar un **DDoS (Distributed Denial Of Service)**, un tipo especial y distribuido de denegación de servicio. Tanto los NIDS como los DIDS, especialmente los últimos, realizan un consumo de recursos de sistema bastante importante.

## Cómo configurar un Firewall

Lo primero, debería ser leer el manual correspondiente al software que hayamos elegido para instalar en nuestro sistema. Después, deberíamos hacer un pequeño estudio acerca del uso que va a tener la máquina donde vamos a instalar el firewall. No es lo mismo configurar un equipo doméstico que un servidor web o un host bastión (se denomina **host bastión** a un ordenador independiente, interpuesto entre la máquina a proteger y la red, de forma que filtre todo el tráfico; Sería convertir un PC en un firewall dedicado :-P). Una vez realizado este estudio, hay que preguntarse cómo preferimos que se definan las reglas: permitir todo por defecto y definir a mano las excepciones o bien denegar todo por defecto y definir a mano las excepciones. Yo recomiendo esta segunda opción. Y por último queda llevar todo esto a la práctica.

Es importante tener en cuenta las **"direcciones de confianza"**. Cuando definimos reglas, podemos hacer que se apliquen a todas las direcciones IP, a una dirección concreta, a un rango de direcciones, a una subred (mediante una IP y una máscara de subred), etc. Estas direcciones de confianza son un grupo especial de direcciones, definidas por el usuario para aplicar reglas de forma excepcional, por ejemplo, permitir netBIOS en red local pero denegarlo de cara a la conexión a Internet. En la mayoría de los firewalls existe la opción de **"Trusted Networks"** o algo similar, donde podremos añadir direcciones, rangos o subredes a este grupo. En IPTables con Linux, podemos añadirlos al archivo `/etc/hosts.allow`, al igual que los denegados se pueden añadir a `/etc/hosts.deny`, donde recomiendo añadir un `"ALL: ALL"` por seguridad.

Aunque un firewall deniegue por defecto toda conexión, siempre prefiero especificar reglas a mano para poder controlar mejor el acceso de paquetes al sistema. Estas otras reglas las añado normalmente por el método de aceptar el tipo genérico de conexión y definir las excepciones denegadas. Veamos algunos ejemplos de configuración:

Para los paquetes ICMP, definimos en primer lugar una regla que acepte la entrada y salida de cualquier paquete ICMP bajo cualquier puerto y dirección, y después definimos reglas para denegar la entrada de paquetes ICMP de los siguientes tipos locales: *AddrMaskReq (17)*, *InfoReq (15)*, *Redirect (5)*, *RouterSolicit (10)*, *TimestampReq (13)*, *EchoRequest (8)*.

Para poder usar netBIOS en una red local (sistemas Windows) y denegar su acceso al exterior, definimos en primer lugar las reglas que permitirán su uso en una red local mediante cuatro grupos de reglas:

- **Aceptar entrada y salida a direcciones de confianza** de paquetes UDP bajo los servicios remotos y locales de datagrama *netbios-dgm (138)*, de nombre de servicio *netbios-ns (137)* y de sesión *netbios-ssn (139)*.
- **Aceptar entrada y salida a direcciones de confianza** de paquetes TCP bajo los servicios remotos y locales de SMB *ms-ds (445)*, de nombre de servicio *netbios-ns (137)* y de sesión *netbios-ssn (139)*.
- **Denegar entrada y salida a cualquier dirección** de paquetes TCP bajo los servicios remotos y locales de SMB *ms-ds (445)*, de nombre de servicio *netbios-ns (137)* y de sesión *netbios-ssn (139)*.
- **Denegar entrada y salida a cualquier dirección** de paquetes UDP bajo los servicios remotos y locales de datagrama *netbios-dgm (138)*, de nombre de servicio *netbios-ns (137)* y de sesión *netbios-ssn (139)*.

Y ahora vamos a intentar entender cómo funciona este conjunto de reglas. Las dos últimas suponen la denegación absoluta de netBIOS entrante y saliente, TCP y UDP a cualquier dirección, mientras que las dos primeras definen unas excepciones para esos mismos protocolos en el caso de tratarse de direcciones de nuestra lista de confianza. Así, nadie podrá acceder desde el exterior a nuestro dominio netbios y ni tan siquiera si logran enviar una petición serviría de algo, pues esta no podría salir a través de nuestro firewall a otro sitio que no fuera una dirección de confianza. Una regla importante que podría incluirse en este grupo, y que protege nuestro sistema de cualquier ataque RPC aunque no estemos parcheados sería: **Denegar entrada a cualquier dirección** de paquetes **TCP** al servicio local **135** y aplicación *Generic Host Process for Win32 Services* (*svchost.exe*).

No voy a definir las reglas para todos los servicios existentes en un ordenador, está claro, pero más o menos creo que se comprende la **idea de generalidad y excepción**, y de esta manera debemos definir las reglas para cualquier aplicación de red. Es importante que en las aplicaciones en las que sea posible, **definamos un rango de puertos** para permitir al firewall controlar de forma más eficiente el acceso a la red de esa aplicación.

En Linux, IPtables nos permite definir reglas mucho más complejas, mediante aplicación condicional de políticas de filtrado de datos. La pega es que la complejidad, al tratarse de comandos es mucho mayor, por lo que o bien se usa una interfaz gráfica que trabaje sobre IPtables, como las nombradas anteriormente, o bien se aplican las reglas generales de cualquier firewall con la sintaxis de IPtables, que no voy a explicar porque sería muy largo y complejo. Podemos ver un pequeño script de bash de ejemplo para entender cómo se estructuran:

```
#!/bin/bash
$IPT=/sbin/iptables

$IPT -A INPUT -p tcp -m tcp --dport 22 --syn -j ACCEPT
$IPT -A INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 ACCEPT
$IPT -A INPUT -i lo -j ACCEPT
$IPT -A INPUT -p tcp -m tcp --syn -j REJECT
$IPT -A INPUT -p udp -m udp -j REJECT
```

Esto es solamente un ejemplo, pero nos sirve para ver la sintaxis de creación de reglas. Para más información, "*man iptables*".

Es importante así mismo **activar los logs para el firewall**, o si se puede, activarlos solamente para determinadas reglas de las que queramos hacer un seguimiento (programas que hayamos denegado de forma explícita, pings entrantes, escaneos de netbios, SMB o RPC a nuestro sistema...). Os sorprenderíais al ver la cantidad de acceso a red que realizan programas que no tendrían porqué. ]:-D

## Alertas y logs

Ya sabemos qué es y cómo trabaja un firewall, un IDS y un sistema híbrido. Ahora llega el momento de saber cómo obtener información valiosa de ellos, además de protección.

**Las alertas son avisos**, generalmente en tiempo real, que nuestro firewall o IDS nos manda (por pantalla, por correo electrónico...) acerca de un evento determinado. Esta funcionalidad es muy común hoy en día en prácticamente cualquier firewall de entorno Windows. En Linux debemos usar algún tipo de IDS o implementación especial de firewall basado en IPtables, pues las propias IPtables no tienen sistema de alertas.

**Los logs son registros**, generalmente en texto plano o html, que genera el firewall o IDS con los eventos importantes o destacados, tales como intrusiones, ataques, y cualquier cosa que le hayamos configurado para que genere un registro.

¿Cuál es la forma de presentarse una alerta en el sistema? Pueden ser muchas, aunque como todo en estos casos, por norma general son bastante parecidas. Para probarlo voy a visitar un sitio web que realiza un escaneo de puertos online a tu PC (<http://www.auditmypc.com/>). Después de aceptar las normas y leer la alerta sobre la IP que va a figurar en el firewall en caso de tenerlo activado, acepto. Obviamente el firewall está bien configurado y el escaneo no detecta nada abierto, pero el firewall salta con una alarma por pantalla que informa de ciertos detalles acerca del evento. Vamos a verla y a entenderla:

*Intrusion Type: Port Scan*  
*Intruding IP: <IP de la web>*  
*Time Of Event: <Fecha>*

*Ethernet Header Information:*  
*Source Addr: <MAC de origen>*  
*Dest Addr: <MAC de destino>*

*IP Header Information:*  
*Source Addr: <IP de la web>*  
*Dest Addr: <Mi IP>*  
*Protocol: TCP*  
*TTL: 109*

*TCP Header Information:*  
*Source Port: 20*  
*Dest Port: 886*

*Packet Data:*  
*[HEX] [ASCII]*

He copiado solamente algunos de los campos que genera la alerta y no siempre de forma literal, pero sí manteniendo una estructura para comprender cómo se genera la misma. Ahora vamos a ver qué significa cada uno:

- **Intrusion Type: Port Scan** - Nos indica el tipo de intrusión detectada (*Port Scan*, *SYN Flood*, *ACK storm...*) y da una pequeña descripción del tipo de ataque.
- **Intruding IP: <IP de la web>** - IP de origen del ataque, en este caso la de la web que realiza el escaneo.
- **Time Of Event: <Fecha>** - Aquí se registra la fecha de detección del evento.
- **Source Addr: <MAC de origen>** - MAC de origen del escaneo. Debido a cómo está estructurada la arquitectura TCP/IP en Internet, las cabeceras MAC son cambiadas por cada enrutador por el que pasa el paquete, por lo que la MAC de origen que figura en mi registro es en realidad la del último enrutador de mi ISP por el que pasó el paquete antes de llegar. Puedo saberlo además porque el OUID de la MAC es el mismo, es decir, los 6 primeros dígitos hexadecimales son idénticos para el enrutador de mi ISP y mi módem (suministrado por el ISP).
- **Dest Addr: <MAC de destino>** - MAC de destino, en este caso la del dispositivo desde el que se originó la petición web y por tanto el que controla la IP que generó esa petición. En este caso no es ninguna de mis tarjetas de red, sino el módem que me proporciona conexión a Internet.
- **Source Addr: <IP de la web>** - Ídem que *Intruding IP*.
- **Dest Addr: <Mi IP>** - IP de destino del ataque. En casos en los que el ordenador tenga varias IP's (no olvidemos que IP es una dirección lógica), la IP de destino nos indica el dispositivo de red usado. No es lo mismo que sea la IP pública (como en este caso) que la IP de una red local, pues ayuda a saber el origen del ataque.
- **Protocol: TCP** - Protocolo que usa el paquete que generó la alerta. En este caso TCP.
- **TTL: 109** - Para no complicar en exceso este dato, diremos que el **TTL (Time To Live)** es una medida que indica a los enrutadores el tiempo de vida que le queda al paquete en la red. Es como una cuenta atrás antes de la destrucción por tiempo de espera del paquete.
- **Source Port: 20** - Puerto de origen del paquete generador de la alerta.
- **Dest Port: 886** - Puerto de destino del paquete generador de la alerta.
- **Packet Data:** - Aquí podemos ver, bien en Hexadecimal o bien en ASCII, el contenido del paquete propiamente dicho. Hace falta un conocimiento de redes muy profundo para comprender o generar paquetes de esta forma, pero generalmente, podemos extraer información del propio paquete.

Por último queda considerar las opciones sobre el **curso de acciones a tomar** que ofrece el firewall. Ofrece bloquear (bien por un tiempo o bien permanentemente) el host de origen del ataque, o bien no bloquearlo. **Aunque no bloqueemos el host, esto no quiere decir que un escaneo de puertos detecte algo, pues las reglas seguirán aplicándose.** Bloquear el host supone una denegación total de cualquier tipo de conexión con esa IP, aunque sea por un programa que tiene permiso en la política del firewall. Es algo **parecido a las direcciones de confianza pero al revés.** Más de una vez no he podido enviar archivos a un amigo y luego he caído en la cuenta de que por alguna pruebecilla que hubiéramos hecho algún día, estaba en mi lista de hosts denegados. :-P

Para entender los logs, voy a generar adrede una entrada en el log de mi firewall. Por defecto mi firewall detecta, deniega y registra los pings entrantes. Pues me voy a hacer ping a mi mismo, pero para evitar que el firewall detecte que me estoy haciendo ping desde mi propia IP y lo ignore, voy a generar un paquete ping con una IP de origen falsa -133.133.133.133- (Igual que se puede ver el contenido en hexadecimal, se puede generar para crear paquetes "especiales"). Ahora vamos a ver el log del firewall y concretamente la entrada correspondiente a ese ping.

```
dd/mm/aaa hh:mm:ss
Traffic
133.133.133.133
Blocked Incoming ICMP EchoRequest
Source 133.133.133.133
Destination <Mi IP>
```

Ahora vamos a analizar esa entrada en el registro (que aunque en el log se genera una línea seguida para cada entrada, he preferido separar por campos).

- **dd/mm/aaa hh:mm:ss** - Hora de generación del registro. Para evitar herir sensibilidades mejor que no veáis a qué horas escribo estas cosas... :-)
- **Traffic** - Tipo de evento (Traffic, Intrusion...) que generó la entrada del registro.
- **133.133.133.133 & Source 133.133.133.133** - IP que generó la entrada del registro.
- **Blocked Incoming ICMP EchoRequest** - Mensaje del generador de registros, en este caso informa del tipo de paquete filtrado.
- **Destination <Mi IP>** - IP de destino, la mía en este caso y que también he censurado.

Ahora ya sabemos interpretar los campos que aparecen en las alertas y los registros de nuestro firewall. Es muy importante ser consciente de que aunque un firewall puede ser instalado y olvidado, dejando que se encargue de todo el filtrado, **es mejor prestar atención a la valiosa información que puede facilitarnos.** Teniendo una IP de origen, podemos consultar las bases de datos WHOIS y averiguar cuál es la empresa que está detrás de esa dirección, pues aunque una IP sea dinámica, todo el rango está registrado por un ISP. En caso de tratarse de ataques graves o reiterados, esos datos pueden ayudarnos a ponernos en contacto con el ISP y que tome medidas en el asunto.

## Ataques e integridad en la máquina local

Ya conocemos el concepto de integridad en la máquina local. En el caso de los virus, la medida en la que la integridad de nuestra máquina queda comprometida es más concreta, tenemos datos y podemos acotar la magnitud de los daños. Un virus tiene unos payloads más o menos fijos que pueden guiarnos para saber cómo actuar. En el caso genérico de **ataques detectados por un firewall o un IDS, es muchísimo más complicado.**

Un firewall puede detectar multitud de eventos como ataques potenciales: un ping denegado, un intento de infección de un gusano de red, escaneo de puertos, aplicación que intenta acceder a recursos de red que le hemos denegado, ataque de denegación de servicio... Como vemos la variedad es enorme, y no podemos equiparar el peligro de un simple ping al de un gusano de red intentando acceder a nuestro sistema, o un atacante intentando dejarnos sin conexión. **El firewall puede darnos datos** a través de las alertas y de los logs, como IP's, protocolos, aplicaciones... pero no valora la peligrosidad de las mismas, son todo eventos de igual magnitud para él. **Somos nosotros los encargados de valorar la peligrosidad** de una alerta y actuar en consecuencia.

Muchas de las alertas que se configuran en un firewall (al menos yo mismo) son poco más que anecdóticas o estadísticas: registrar los pings que te hacen, cuántas veces intenta acceder sin tu permiso el Reproductor Multimedia de Windows a Internet, cuántas personas se aburren lo suficiente como para escanear la red en busca de equipos con recursos compartidos... **Esas alertas hay que saber valorarlas en su justa medida**, sin alarmarse cada vez que veamos una entrada. Un escaneo de puertos reiterado a nuestra máquina desde una misma IP sí es un motivo de alarma mayor, y es en esos casos en los que toman realmente importancia los datos, y cuando debemos comprobar que la integridad de nuestra máquina sigue intacta.

También hay que tener en cuenta que **un firewall no es infalible**. Los programas tienen fallos y vulnerabilidades, y el software específico de protección no es menos. De vez en cuando se descubren vulnerabilidades para firewalls e IDS. Es conveniente estar al tanto (la mejor forma es apuntarse a las listas de correo específicas de <http://www.securityfocus.com/>) y **parchar o actualizar de vez en cuando** el software. Es así mismo importante darse cuenta que cuando un software está muy extendido, la gente se esfuerza más en encontrarle fallos: Zone Alarm es un buen firewall, pero aún así cualquiera de los otros citados anteriormente es una mejor opción, pues Zone Alarm dispone de una versión gratuita que está muy extendida en Internet y para la que han aparecido en los últimos meses varias vulnerabilidades, que dicho sea de paso, los programadores de ZoneLabs no parchean con demasiada celeridad. Así mismo, hay fallos que pueden dejar "agujeros" que expongan nuestro sistema a ataques. Lo mejor para luchar contra todos estos problemas es **estar informado**. Hay una opción más a la hora de elegir firewalls: el **Software Libre**, que no sólo IPtables es código abierto cuando hablamos de firewalls. En contra de lo que muchas corporaciones nos quieren hacer creer, **el software de código abierto no es menos seguro que el software propietario**, pues conocer el código de un programa bien hecho no ayuda a desmantelarlo. Una buena opción en este campo es **Portus** (<http://www.opensourcefirewall.com/>), poderoso firewall que ha acumulado varios premios de seguridad y cuya calidad es equiparable a cualquier otro de código propietario.

Así pues, podemos concluir diciendo que hay que vigilar tanto la actividad del firewall con sus alertas y logs, como estar al tanto en noticias de seguridad y publicaciones de parches y actualizaciones. Es importante proteger nuestros ordenadores, y esa seguridad hoy por hoy no tiene precio.

## **- Seguridad en redes de pares (p2p)**

Las redes de pares, el *peer-to-peer*. Creo que habrá poca gente hoy en día que no haya oído hablar, para bien o para mal, de las redes p2p. Pero, ¿qué es una red p2p?

Las redes de pares, también conocidas como **redes peer-to-peer (p2p) son enormes redes de intercambio distribuido** sustentadas sobre la arquitectura de Internet. En su esencia las conexiones que tienen lugar son conexiones punto a punto, pero éstas están organizadas según una estructura que conforma esa red. Se trata de **redes distribuidas, auto-estructuradas y con una gran escalabilidad**. Se distinguen en general dos tipos de redes de pares: Las **redes p2p híbridas** tienen un conjunto de hosts conectados y **unos pocos hosts que actúan como servidores** y facilitan la conexión entre los demás. Redes como la red *eDonkey* o la red *Overnet* se encuentran entre las redes p2p híbridas. El otro tipo son las **redes p2p puras**, en las que **todos y cada uno de los hosts actúa como cliente o servidor** según las necesidades, como por ejemplo en la red *Freenet*.

**Decimos que una red p2p es distribuida** puesto que se sustentan en un gran grupo de hosts (**a un host de una red p2p se le denomina *peer***), y la funcionalidad de la red es independiente a los fallos técnicos que puedan acaecer en un segmento de la misma.

**Decimos que una red p2p es auto-estructurada** puesto que una arquitectura p2p genera su propia organización, bien sea a nivel de host, donde cada host que adquiere permiso para convertirse en nodo de la red, **adquiere inmediatamente igualdad con respecto a todos los demás nodos**, o a nivel de red, pues una gran red p2p puede ser segmentada a su vez en cualquier número de subredes virtuales y un nodo puede funcionar independientemente de la disponibilidad de los demás peers, lo que proporciona una capacidad de conectividad variable.

**Decimos que una red p2p es escalable** porque puede ser organizada de cualquier forma, con **máxima segmentación o unidad total**, además de ser posible **aumentar su tamaño tanto como se quiera**, puesto que la arquitectura no entorpece el crecimiento en ningún momento.

Las redes p2p no son algo nuevo, aunque en su concepción actual puede decirse que nacieron con Napster, que supuso una vuelta de hoja en Internet. **Napster** cerró (la cerraron) y aunque ahora pretende reabrirse con otro concepto de red y conservando el nombre, sentó unas bases que darían lugar a las actuales redes p2p, apoyadas en otros pilares, como **Audiogalaxy**. Hoy en día las principales redes que podemos encontrar son:

**La red FastTrack**. Heredera directa de Napster, esta **red p2p híbrida** tiene su más conocido representante en Kazaa. Es, quizá, la red p2p más usada y conocida, y también la más polémica en los últimos tiempos gracias a la RIAA y sus constantes envites contra los usuarios de esta red. Aunque principalmente se intercambia música, también se intercambia todo tipo de material. Clientes: **Kazaa** (<http://www.kazaa.com/>) en sistemas Windows; **Gift** (<http://giftcurs.sourceforge.net/>) en sistemas Linux y **Mldonkey** (<http://www.freesoftware.fsf.org/mldonkey/>) en ambos sistemas.

**La red eDonkey**. Se trata de una de las principales redes hoy en día y aunque FastTrack la aventaja en términos globales, en España está mucho más extendida la red eDonkey. Se trata de una **red p2p híbrida** con un sistema de servidores que organizan a unos clientes que intercambian los ficheros según un **sistema de créditos** de forma que enviando más se obtienen más posibilidades de descargar. En esta red se intercambian todo tipo de ficheros. Clientes: **eDonkey 2000** (<http://www.edonkey2000.com/>), **eMule** (<http://www.emule-project.net/>), **eMule Plus** (<http://www.emuleplus.tk/>) en sistemas Windows; **Xmule** (<http://www.xmule.org/>), **aMule** (<http://amule.sourceforge.net/>) en sistemas Linux y **Mldonkey** (<http://www.freesoftware.fsf.org/mldonkey/>) en ambos sistemas.

**La red Overnet**. Esta red, nacida de los creadores de la red eDonkey, es parecida en su concepción, aunque con una gran diferencia: el sistema de servidores distribuidos es sustituido por un único servidor que coordina a todos los clientes. Se trata de una **red p2p híbrida** donde se comparten todo tipo de ficheros. Clientes: **eDonkey 2000 híbrido** (<http://www.edonkey2000.com/>), **Overnet** (<http://www.overnet.com/>) en sistemas Windows.

**La red Kademia.** Esta red aún está desarrollándose, y nace de mano de los creadores del cliente p2p más famoso para redes eDonkey: eMule. La nueva red convivirá con la actual red eDonkey en los clientes eMule. Se trata de una **red p2p híbrida** con un único servidor, al estilo de la red Overnet, pero en este caso se trata de una red libre, tanto en tecnología como en implementación. Se comparte todo tipo de ficheros. Clientes: **eMule** (<http://www.emule-project.net/>) en sistemas Windows.

**La red OpenNAP.** Intento de revivir el espíritu Napster a través de una red libre. Se trata de una **red p2p híbrida**. Clientes: **WinMX** (<http://www.winmx.com/>) en sistemas Windows; **Lopster** (<http://lopster.sourceforge.net/>) y **Gnapster** (<http://jasta.gotlinux.org/gnapster.html>) en sistemas Linux.

**La red WinMX.** Esta red es otra de las herederas de Napster. Se trata de una **red p2p híbrida** principalmente orientada al intercambio de música. Clientes: **WinMX** (<http://www.winmx.com/>) en sistemas Windows y **Loophole** (<http://www.loopholesoftware.com/>) en sistemas Linux.

**La red Souseek.** Red heredera de Audiogalaxy. Se trata de una **red p2p híbrida** principalmente orientada al intercambio de música, aunque se encuentran igualmente otro tipo de ficheros. Clientes: **Souseek** (<http://www.slsknet.org/>) en sistemas Windows; **PySoulSeek** (<http://www.sensi.org/%7Eak/pyslsk/>) en sistemas Linux y **Mldonkey** (<http://www.freesoftware.fsf.org/mldonkey/>) en ambos sistemas.

**La red Gnutella.** Esta **red p2p pura** está orientada a intercambio de todo tipo de ficheros. Clientes: **Morpheus** (<http://www.morpheussoftware.net/>) en sistemas Windows; **LimeWire** (<http://www.limewire.com/>), **Qtella** (<http://www.qtella.net/>), **Mutella** (<http://mutella.sourceforge.net/>) en sistemas Linux y **Mldonkey** (<http://www.freesoftware.fsf.org/mldonkey/>) en ambos sistemas.

**La red Bittorrent.** Esta red de nueva aparición tiene una concepción distinta del término cliente, puesto que no existe uno tal y como los entendemos, compartes cuando descargas. Se usa para todo tipo de ficheros y hoy en día está creciendo muy rápido. Se trata de una **red p2p híbrida**. Clientes: **BitTorrent** (<http://bitconjurer.org/BitTorrent/>) en sistemas Windows; **Snark** (<http://freshmeat.net/projects/snark/>) en sistemas Linux y **Mldonkey** (<http://www.freesoftware.fsf.org/mldonkey/>) en ambos sistemas.

**La red Freenet.** Se trata de una nueva y revolucionaria **red p2p pura totalmente anónima**, con conexiones encriptadas y descentralización total de nodos. Es, definitivamente, el futuro del p2p. Clientes: **Freenet** (<http://freenet.sourceforge.net/>) para sistemas Windows y Linux.

Como vemos, existe una enorme variedad de redes p2p donde elegir, así como clientes para todas ellas bajo cualquier plataforma. Adicionalmente, casi cualquier cliente de Linux cuyo código esté liberado, puede ser compilado bajo Windows, y cualquier cliente nativo de Windows puede ser usado desde Linux con herramientas como Wine, WineX o VMware. Existen así mismo clientes como Mldonkey cuya filosofía es aceptar el mayor número de protocolos posibles.

Hablando de seguridad en redes p2p, podemos distinguir entre tres tipos: **fallos o vulnerabilidades en el software**, que pueden llegar a comprometer el sistema; **virus que se distribuyen por redes p2p**, como los que nuestros simpáticos "amigos" de la RIAA introducen en la red FastTrack; y por último **ataques a la privacidad a los usuarios de la red**, bien sea por una monitorización de datos o por una interceptación de las conexiones.

En el caso de fallos y vulnerabilidades del software usado, no hay más que **estar al tanto de las actualizaciones** de dicho software a través de las páginas web de los desarrolladores, y en caso de detectarse una vulnerabilidad aún no parcheada, actuar según se recomiende en los sitios dedicados a la seguridad informática. Para los virus que circulan por ciertas redes p2p, especialmente la red FastTrack, no hay más remedio que remitirnos a la parte de virus: **mantener el software antivirus convenientemente actualizado** y alerta a los ficheros que descarguemos de la red.

El caso de los ataques a la privacidad es otra historia, y según están las cosas hoy por hoy, algo a tomarse muy en serio. No voy a entrar a discutir quién nos espía y porqué, pero está más que demostrado que **las redes p2p están siendo espiadas**. Por suerte hay muchas cosas que podemos hacer. Lo principal es **obtener una lista de IP's y rangos de IP's que sean potencialmente peligrosas** para la privacidad de nuestras comunicaciones. En cualquier foro dedicado al p2p será sencillo encontrar buenas listas actualizadas. Después, tenemos tres formas de protegernos:

En primer lugar **podemos usar nuestro propio firewall** para añadir reglas específicas que denieguen cualquier acceso a esos rangos de direcciones IP. Es efectivo, aunque introducir listas con cientos de entradas puede resultar tedioso. No obstante también pueden encontrarse archivos de reglas con los rangos introducidos para los firewalls más usados (he visto listas para Kerio Personal Firewall).

La segunda opción depende de la red y el cliente que usemos, pero ya está comenzando a ser una característica habitual la inclusión de un pequeño **motor de filtrado de direcciones en los clientes p2p**; eMule, eMule Plus, Xmule, aMule y otros clientes ya incorporan el sistema *ipfilter.dat* para este propósito. Existen muchas listas de filtrado por la red que son actualizadas casi diariamente.

La última opción, y quizá la más conveniente por su compatibilidad con cualquier red y cliente p2p, es el uso de **herramientas específicas de filtrado de direcciones**. Estas herramientas funcionan de forma similar a un firewall, pero son más específicas en su finalidad: solamente filtran direcciones o rangos de direcciones. Normalmente se aconseja su uso de forma temporal, mientras se tenga activado el sistema p2p. Los programas de este tipo más famosos son **PeerGuardian** (<http://www.peerguardian.net/>) y **P2P Hazard** (<http://adkiller.net/p2p>). Cualquiera de ellos dos permite actualizar su base de datos de direcciones desde ficheros que podemos descargar sencillamente a través de Internet o desde el propio cliente. Gracias a estas pequeñas utilidades, podemos estar más seguros en el uso de redes p2p.

Finalmente hay algunos consejos breves y simples, pero igualmente importantes para salvaguardar nuestra intimidad en redes p2p. Es importante en las redes que permitan el uso de nicks o seudónimos, **no usar bajo ninguna circunstancia nuestro nombre real o nuestro nick habitual**. Además es conveniente cambiar con frecuencia este alias. También es importante, en los clientes que lo soporten, **denegar por completo la posibilidad de que sea vista nuestra lista de ficheros compartidos**, ni tan siquiera permitirla a nuestra lista de amigos. Por último lugar, en los clientes que lo permitan, es importante **no usar nunca los puertos por defecto** para realizar las conexiones a la red p2p, y **cambiarlos con frecuencia**, como mínimo una vez a la semana.

Siguiendo estos consejos, es posible sentirse seguro en redes p2p.

## **3. Seguridad en las comunicaciones**

### **- Introducción: La privacidad, bien escaso**

Aunque sea difícil de aceptar, hoy en día nuestra privacidad no está asegurada. A lo largo de estos días hemos podido comprender mejor la importancia de la seguridad en Internet y cómo ésta ha evolucionado a lo largo de sus años de existencia, y es por ello que podemos entender mejor nuestra situación en la misma.

Al principio de la existencia de Internet, la privacidad no era una preocupación de los usuarios: confiaban en sus leyes y en la protección de sus ISP. Internet ha crecido en magnitud y en importancia, y los datos que viajan por sus líneas lo han hecho igualmente... pero nuestros ISP -con sistemas de proxy caché dudosos y unos servicios muy deficientes- y nuestras leyes abanderadas por la infame LSSI han evolucionado en el sentido contrario. Así pues, ¿quién nos garantiza nuestra privacidad hoy día? La respuesta es sencilla: como en el caso de la seguridad, debemos ser nosotros mismos los que nos la procuremos.

Existen métodos seguros de trabajar con información en web, y podemos comprar con nuestros números de tarjeta de crédito, consultar el saldo de nuestra cuenta de ahorros o el expediente en nuestra universidad. Pero, ¿sabemos realmente en qué consiste esa seguridad? En muy pocos casos. El correo postal cada vez se usa menos, y los correos electrónicos han aumentado su volumen, a la par que la información contenida en los mismos es más sensible. Nos sentimos seguros usando servicios de correo electrónico, pero muy poca gente comprende los riesgos de esas comunicaciones o las posibles opciones para disminuirlos que tenemos al alcance de nuestra mano.

Partimos de la base de que nada es absolutamente seguro o absolutamente inseguro. Todo es relativo, y es por ello que debemos comprender cuándo se hace necesario usar técnicas criptográficas en el correo electrónico o no, cuándo aceptar o rechazar un certificado de seguridad de una página web, hasta qué punto se pueden obtener datos nuestros en una simple visita web...

Comprender y valorar esos riesgos y actuar según las circunstancias es un día a día en la red de redes, y debemos saber valorarlos, pero también debemos conocer qué podemos hacer para mitigar ese halo de ausencia de privacidad que rodea todo lo concerniente a Internet.

¿Qué motivos tenemos para querer esa privacidad? Supongo que cada uno tiene los suyos, pero creo que el más importante es que tenemos derecho a ella, y en mi opinión eso nos da el deber de defenderla. Merece la pena.

**Death Master**

## - Spyware: qué es y cómo evitarlo

Spyware, un concepto relativamente joven, pero que se ha ganado una merecida muy mala fama en su corto tiempo de existencia. Pero, ¿qué es el spyware? ¿Es tan peligroso? ¿Qué podemos hacer para evitarlo?

**Denominamos spyware a cualquier tecnología que sirva para recolectar información sobre una persona u organización sin su conocimiento.** Hay que distinguir sin embargo el spyware, de lo que es el adware y el badware, pues usualmente se utilizan como sinónimos estos tres términos y no lo son en absoluto.

**Entendemos por adware cualquier tipo de software que incluya banners o carteles publicitarios** con fines de financiación. Puede ocurrir no obstante que esos carteles lleven algún tipo de spyware a su vez, en cuyo caso ya no encajarían en la definición de adware, y serían considerados un spyware en toda regla.

**Se denomina badware a cualquier tecnología destinada a controlar o hacer daño a un usuario.** Esta definición es muy "difusa", pues realmente este término no es muy usado, y el software denominado badware resulta extremadamente difícil de erradicar, y algunos ni tan siquiera residen en nuestra máquina (como los misterios de **Echelon** y **Carnivore**).

Cabe reseñar que el problema del spyware es **prácticamente exclusivo de sistemas Microsoft Windows**, pues el spyware es normalmente diseñado para los mismos. En sistemas Linux el máximo problema que podemos tener son determinadas **cookies** sospechosas.

Una vez comprendida esa diferencia, pasemos a centrarnos en el spyware. Como ya hemos dicho, es un tipo de software destinado a obtener información de un usuario sin su consentimiento. Antes de nada, conviene conocer las formas en las que este tipo de programas entra en nuestro sistema para poder prevenirlos:

**a)** A través de **software instalado en nuestro sistema.** Usualmente se trata de software **shareware** (software de evaluación), **freeware** (que ya deja de resultar tan "free") o **adware** (ya comentado) que incluyen funciones, programas o scripts subyacentes y que constituyen el propio spyware. Este tipo de programas son comúnmente descargados de Internet con total confianza. El ejemplo más común es el software **GATOR**.

**b)** A través de **correo electrónico.** Similar al anterior, solo que en lugar de tratarse de software, se trata de algún **fichero o archivo que contiene oculto el spyware.** También pueden tratarse de mails que **aprovechen vulnerabilidades concretas de algún cliente de correo** para introducirse de forma silenciosa en el sistema.

**c)** A través de **redes p2p.** Hoy en día un método en auge, debido a los fallos de redes como FastTrack (Kazaa) que han dado lugar a un caldo de cultivo compuesto por virus, troyanos, spyware... Estos medios resultan ideales debido a la **enorme difusión de estas redes**, y a que es **difícilmente denunciabile** la introducción de spyware dado el "poco afecto" que tienen las autoridades al p2p.

**d)** A través de **webs visitadas.** Generalmente mediante cookies "sospechosas" o mediante vulnerabilidades del navegador que permitan la implantación del spyware (típicamente vulnerabilidades de Internet Explorer).

Obviamente pueden darse otras formas de propagación de spyware, pero son muchísimo menos frecuentes, y las principales fuentes del mismo son las cuatro anteriores.

Ahora es el momento de explicar un término aparecido arriba, el término **cookie.** **HTTP (Hyper Text Transfer Protocol)** es el protocolo encargado de la gestión y visualización de páginas web. Este protocolo, a diferencia de otros, **es un protocolo sin estado**, es decir, **cada vez que se sirve una petición, se cierra la conexión y el servidor olvida todo dato del usuario.** Esto supone una ventaja porque **elimina carga del servidor en favor del cliente**, pero supone un problema porque **el servidor no es capaz de recordar datos** del cliente, por lo que mantener una sesión en un foro o ver una web según el estilo que queremos sería imposible.

Para paliar este problema se idearon las **cookies**, archivos de texto plano que almacenan información acerca de nuestra relación con distintos servidores webs para poder ofrecérsela al mismo al iniciar una conexión y que éste nos “recuerde”. A primera vista, eso de almacenar información puede sonar a spyware, pero no: las cookies siempre **pueden ser controladas, aceptadas, rechazadas, consultadas y modificadas por un usuario**. Hay unas sencillas reglas que siguen las cookies: no pueden ocupar más de 4 Kb, un servidor sólo puede leer las que él envió, cada servidor o dominio sólo puede almacenar 20, y sólo puede haber un total de 300. Así a bote pronto alguno pensará que si un servidor puede escribir esa información de “memoria de usuario” en nuestro disco duro, podría escribir lo que quisiera, pero para tranquilidad vuestra os diré que no, puesto que las cookies son escritas al disco por el navegador, nunca por el servidor. Esto es a grandes rasgos lo que necesitamos saber acerca de las “galletitas”.

Ahora bien, **¿porqué algunas cookies son consideradas spyware?** Porque muchas de ellas almacenan información que no controla ninguna sesión, contador o información de relación usuario-servidor. Pongamos por ejemplo la empresa de publicidad DoubleClick. Prácticamente todo el mundo tiene cookies suyas sin haber visitado jamás su servidor. Esto es debido a que DoubleClick gestiona los banners publicitarios de muchísimos sitios web, y estos banners se sirven directamente desde su servidor. Al servirse, se encargan de **almacenar de paso información acerca de nuestros hábitos de navegación** en Internet y **usan esa información para personalizar la publicidad que recibimos** en nuestro navegador. Puede parecer útil y con sentido, pero ¿qué hay de nuestra privacidad? Pues eso.

La información que el spyware toma de nuestro sistema suele ir orientada a la **recopilación de estadísticas** de hábitos de navegación para la personalización de banners y pop-ups, **obtención de direcciones de correo electrónico para envío de spam** (correo generalmente publicitario que es enviado sin autorización ni petición del usuario), o incluso **obtención de datos personales sensibles** para estadísticas, tales como edad, sexo, empleo, remuneración...

Hay otro tipo de spyware menos notorio pero más peligroso y sigiloso que es **el que las grandes empresas de software incluyen** de forma silenciosa en sus productos para controlar su uso y difusión. Es notorio el caso de varias **funciones cuando menos sospechosas del sistema operativo Microsoft Windows**, y algunos programas como **Windows Media Player** o **Windows Messenger**. Sólo necesitamos  **echar un vistazo al log del firewall** si denegamos la conexión a estos programas, o  **echar mano de un oportuno sniffer...** De hecho algunas de estas funciones “sospechosas” del sistema Windows y sus programas  **están incluidas en las bases de datos de programas antispyware**. Para evitar estos contratiempos con el sistema Windows XP (donde más se pone de manifiesto), recomiendo la descarga y utilización de un sencillo programa gratuito llamado **XP-Antispy** (<http://www.xp-antispy.org/>).

Es hora de hablar de las precauciones a tomar para salvaguardar nuestra privacidad del infame spyware:

- **Instalación de un software específico** especializado en la detección y eliminación de spyware, **actualización periódica**, así como **revisión completa del sistema cada cierto tiempo**. Los programas especializados en esta tarea más conocidos son:
  - **Spybot – Search & Destroy** (<http://security.kolla.de/>): Se trata de uno de los mejores de su clase. Es un programa completamente gratuito y cuya base de datos se actualiza con frecuencia para estar siempre al día. Posee un sistema de “vacunación” del sistema contra elementos peligrosos. Es mi favorito y el que recomiendo a todo el mundo.
  - **Lavasoft Ad-aware** (<http://www.ad-aware.com/>): Es, quizá, el más famoso entre este tipo de software. Incluye una poderosa base de datos que se actualiza con frecuencia.
  - **Spyware Blaster** (<http://www.wilderssecurity.net/spywareblaster.html>): No se trata de una herramienta de escaneo y detección, sino que se orienta a la prevención y protección contra el spyware.
  - **SpywareGuard** (<http://www.wilderssecurity.net/spywareguard.html>): Herramienta de protección en tiempo real contra el spyware. Funciona de manera similar a un antivirus, con alertas en tiempo real.
  - **BPS Spyware Remover** (<http://www.bulletproofsoft.com/>): Otro gran software en su clase, si bien ha tenido problemas legales por unas bases de datos de spyware robadas a Spybot – Search & Destroy.
  - **SpyCop** (<http://www.spycop.org/>): Uno de los primeros, aunque hoy por desgracia no es tan común ni tan bueno como las demás opciones.

- **Evitar el uso del navegador Microsoft Internet Explorer**, debido a sus constantes **fallos de seguridad** y la consiguiente exposición de nuestra máquina, así como su **deficiente sistema de gestión de cookies**. Es recomendable el uso de un navegador que permita una gestión más potente y personalizada de cookies. Una gran opción es **Mozilla Firebird** (<http://www.mozilla.org>) que además es software libre y gratuito.
- **Restringir las cookies y permitir las únicamente a los sitios de confianza** en los que sea imperiosamente necesario su uso (webs en las que queramos poder realizar uso de sesiones y usuarios).
- **Evitar la descarga de ficheros de Internet cuando su procedencia sea dudosa** (al igual que ocurría con las precauciones contra los virus).
- **Asegurarse que el software instalado en el sistema no posee spyware** o elementos que atenten contra nuestra privacidad (es muy recomendable para evitar este tipo de incidentes el uso de software libre).

## - Privacidad en Internet

Por el mero hecho de navegar por Internet, **estamos revelando gran cantidad de información** sobre nosotros mismos, desde la IP e ISP hasta nuestro navegador y sistema operativo. Estos datos pueden parecer inofensivos, pero bajo ciertas circunstancias y en determinados servidores, el **revelar por ejemplo el sistema operativo y el navegador constituye un gran riesgo**. Este dato viaja en una cabecera (**User Agent**) y tiene más o menos este aspecto para distintos valores: **IE 6.0 en Windows XP**, *Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)*; **Netscape 4.8 en Windows XP**, *Mozilla/4.8 [en] (Windows NT 5.1; U)*; **Opera 7.21 en Windows XP**, *Opera/7.21 (Windows NT 5.1; U) [en]*... Es conveniente el uso de un sistema operativo que permita, bien por si mismo o bien mediante plugins, la modificación de este dato para no revelar ninguna información al respecto. Yo por ejemplo al navegar dejo como valor de esa cabecera mi nick, *Death Master*.

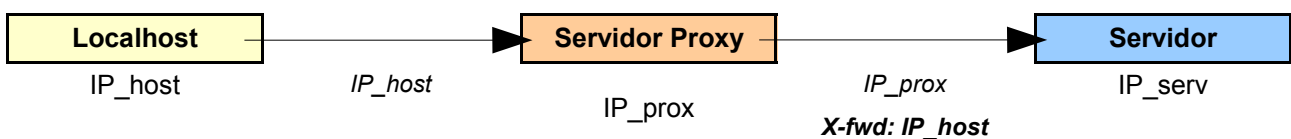
El tema de la IP es algo más complicado, y es hora de hablar de los Proxys.

### Uso de proxys anónimos

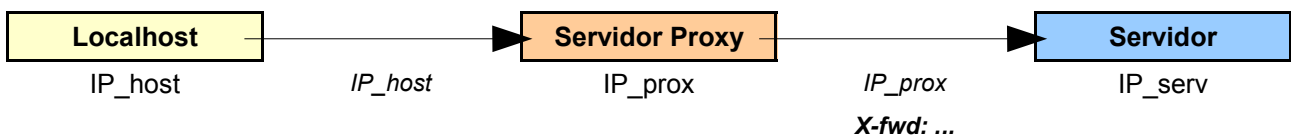
La definición de proxy es muy general, pues existen **muchos tipos de conexiones proxy**: caché en disco, un firewall (sí, también puede ser un proxy), control de accesos, pasarela de protocolos seguros... y no siempre en Internet. Puede estar perfectamente en LAN o incluso en un equipo localmente. Así pues, **definiremos como proxy una tecnología, física o lógica, que actúa como pasarela de información**. Ahora vamos a definir lo que "normalmente" se entiende como proxy: **Sistema capaz de actuar como puente en la conexión entre la máquina local y un servidor remoto**. ¿Y qué diferencia hay entonces con los distintos routers que atraviesa una conexión hasta llegar a un servidor? Que estos son conexiones físicas impuestas por la topología de la red, mientras que en el caso de los proxys son conexiones lógicas que normalmente son elegidas por el usuario. Y digo normalmente porque últimamente **las ISP se dedican a imponer un sistema de proxy caché transparente entre los usuarios e Internet**, con el fin de actuar como una inmensa cache de disco y ahorrar ancho de banda en las conexiones a otros servidores, al menos esa es la teoría...

Ahora que tenemos una idea de qué es un proxy en su acepción más común... llega la hora de entender qué diferencia existe entre un proxy normal y uno anónimo. Un **proxy normal se interpondrá** entre nosotros y la máquina a la que nos conectamos, y aparecerá su IP como origen, pero la nuestra aparecerá como redireccionada a través de él (campo **X-Forwarded**). Un **proxy anónimo está específicamente diseñado para que nuestra dirección IP no pueda ser conocida** por la máquina a la que nos conectamos. Esto es muy útil para poder navegar de forma anónima: al no tener nuestra IP, tampoco pueden averiguar con un simple *whois* el ISP y más datos de importancia. Si además cambiáis el User Agent como recomendé (y para lo cual recomiendo nuevamente **Mozilla**), el anonimato a la hora de visitar webs es mucho mayor.

Proxy normal:



Proxy anónimo:



¿Dónde encontrar listas con proxys anónimos que podamos usar? Hay multitud de webs con listas actualizadas, y aquí dejo unas cuantas:

- <http://www.multiproxy.org/>
- <http://www.atomintersoft.com/products/alive-proxy/proxy-list/>
- <http://www.cyervonet.com.ar/misc/proxys.php>
- <http://www.proxybench.com/proxy/proxylist.asp>
- <http://www.samair.ru/proxy/fresh-proxy-list.htm>

Existen también programas para **buscar proxys de forma manual**, como **YAPH (Yet Another Proxy Hunter)** que podemos encontrar en <http://yaph.sourceforge.net/> y que usa el motor de **NMAP** (<http://www.insecure.org>) para buscar proxys. Sólo para plataformas Unix. ;-)

Es importante señalar que **no siempre un proxy supuestamente anónimo lo es**, pues en algunas ocasiones al usar uno de ellos y comprobar si aparecemos como redireccionados en el campo X-Forwarded (por ejemplo a través de <http://www.cualesmiip.com/>) nos encontramos con que se trata de un proxy normal. Esto es debido a que estos proxys aparecen y desaparecen como la espuma, algunos de ellos son establecidos de forma "poco legal" (por ser suaves) y muchas veces no son lo que parecen ser. Así pues es importante **comprobar el supuesto anonimato usando el proxy** (en el menú de opciones de conexión de cualquier navegador encontramos un campo para el uso de proxys) y asegurarse de que hace lo que debería.

Aunque un proxy anónimo es un método suficientemente seguro para no dejar huellas en un servidor web al visitarlo, ese servidor puede almacenar datos de sus conexiones aunque no se las muestre al servidor conectado, con lo cual ese anonimato no es más que aparente. **Algunos proxys guardan logs** de sus conexiones (la mayoría) y otros no los guardan. Así pues la mejor opción para navegar de una forma realmente anónima es el **uso de cadenas de proxys y cascadas de proxys**.

En realidad aunque se guarden logs, muchas veces estos servidores están establecidos en países con **leyes informáticas bastante permisivas e incluso inexistentes**. Esto unido a que algunos de esos países no mantienen relaciones muy cordiales con occidente ni sus empresas, aumenta la dificultad de acceso a sus logs. Por último, una cadena suficientemente grande supone enlazar gran cantidad de estos servidores, por lo que poca gente tendrá dinero y ganas como para ir de un país a otro preguntando por unos logs que quizá no existan o que no se les facilitarán (pero hay que tener presente que sí hay gente dispuesta a hacerlo y que dispone de los medios). Si las conexiones mediante proxys anónimos no son para fines ilegales, nadie se molestará en intentar seguir el rastro, en caso contrario lo perseguirán y lo encontrarán.

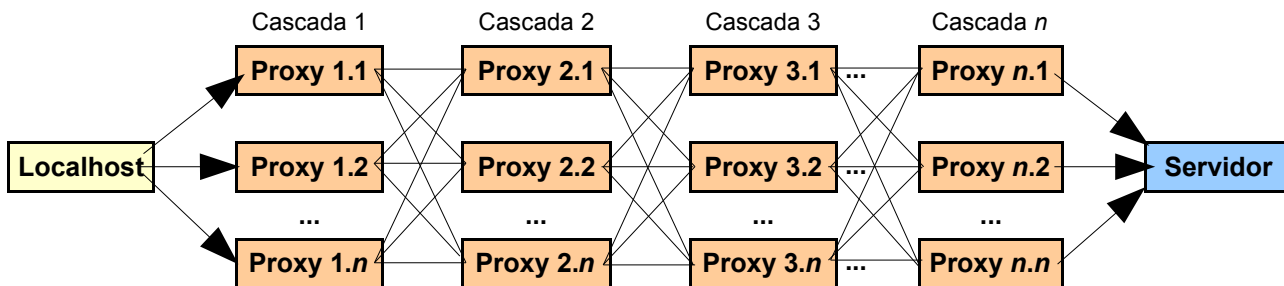
Una **cadena de proxys** sitúa un proxy detrás de otro, cada uno se conecta al siguiente, y **el último se conecta al servidor**, con un camino definido que resultaría sencillo de seguir a la inversa en el caso de tratarse de proxys que almacenen logs, al menos en teoría. Si alguno de ellos no almacena logs, se rompería esa cadena inversa y sería prácticamente imposible (que no totalmente) seguir el rastro de esa conexión.

Cadena de proxys:



Una **cascada de proxys** es una lista de proxys en la que al ser lanzada la petición, ésta **atraviesa aleatoriamente uno de ellos** y después continúa su camino en la red. Usualmente su uso se da en **cadenas de cascadas de proxys**, uno de los métodos más seguros y anónimos de conexión.

Cadena de cascadas de proxys:



Mediante este tipo de cadenas de cascadas, cada petición atraviesa todas las cascadas pero un proxy aleatorio en cada una de ellas, lo que tiene dos consecuencias principales: **se genera una cantidad de caminos distintos enorme** para cada petición (y una sesión HTTP, por ejemplo, contiene muchas peticiones), y **la IP que aparece en el servidor final es una aleatoria perteneciente a la última cascada**, con lo que cada petición que generemos se hará con una IP distinta.

El software que permite crear a mano cadenas no suele tratar cadenas de cascadas debido a la complejidad de las mismas, pero sí nos permite la creación de cadenas de proxys normales. En **sistemas Windows** la forma más fácil de realizar cadenas y anonimizar una conexión es mediante los programas **SocksChain** (<http://www.ufasoft.com/socks/>), que permite **crear las cadenas de proxys** propiamente dichas y **SocksCap** (<http://www.socks.permeo.com/Download/SocksCapDownload/index.asp>) que permite **interceptar la conexión de un programa que no admita uso de proxys y obligar a la conexión a pasar por SocksChain**. El uso de ambos programas es muy sencillo e intuitivo, y como en la mayoría de los casos actúa como un proxy local que queda a la escucha y al que tenemos que conectarnos (127.0.0.1:xxxx). En **sistemas Linux** contamos con un programa llamado **ProxyChains** (<http://proxychains.sourceforge.net/>) que realiza ambas funciones: **creación de cadenas de proxys e intercepción de llamadas TCP**.

Existen también **programas especializados en la automatización de estas tareas** mediante la creación y gestión de las cadenas. Son mucho más sencillos de usar y permiten (algunos) el uso de cadenas de cascadas de proxys, pero generalmente **están gestionados por alguna empresa**, y eso de por sí puede ser ya un punto en contra de la privacidad que no supone la creación a mano. Cada uno puede decidir. Para sistemas Windows contamos (entre otros) con **MultiProxy** (<http://www.multiproxy.org/>), uno de los primeros pero hoy en día no muy usado y **Stealthier** (<http://www.photon-software.de/Stealthier/>), en mi opinión el mejor sistema de automatización, puesto que es capaz de gestionar cadenas de cascadas de proxys combinadas con nodos de encriptación (de criptografía hablaremos más adelante). Mi única pega para este software es que es exclusivo de Windows :-P. Así mismo contamos con **JAP** ([http://anon.inf.tu-dresden.de/index\\_en.html](http://anon.inf.tu-dresden.de/index_en.html)), un software de anonimato que al funcionar bajo Java (un lenguaje de programación) es compatible con Windows, Unix, Linux, Mac, OS/2...

La última opción disponible para anonimizar la navegación es usar **sitios webs que redireccionan las peticiones**, de forma que desde el propio navegador y **sin uso de software alguno** puedes navegar sin dejar tu IP. Las pegadas a este tipo de anonimato son muchas, como que sólo lo es respecto al servidor puesto que estas webs registran las conexiones, no aceptan el uso de sesiones a través de protocolos seguros, desactivan todo el java o javascript, no aceptan cookies... Pero aún así a veces son útiles, por lo que indico algunos ejemplos bastante conocidos: **@nonymouse** (<http://anonymouse.ws/anonwww.html>), **Anonymizer** (<http://www.anonymizer.com/>), **beHidden!** (<http://behiddn.com/>).

### Protocolos seguros y certificados de seguridad

A veces cuando navegamos por Internet necesitamos intercambiar información sensible con un servidor. En este caso no nos preocupa nuestro anonimato, si se ve o no la IP, o el User Agent... en este caso nos interesa que esos datos viajen por un canal de comunicación seguro, de forma que el servidor y nuestra máquina puedan verlos pero no el resto de Internet. Esto se logra mediante los protocolos de seguridad.

Existen multitud de plataformas seguras para la comunicación en diversos protocolos: **SHTTP (Secure Hypertext Transfer Protocol)**, **SSL (Secure Socket Layer)**, **TLS (Transport Layer Security)**, **IPSec...** Nosotros hablaremos brevemente de SHTTP y de SSL. Una explicación profunda de ambos protocolos podría perfectamente ocupar un mínimo de cincuenta hojas de texto, por lo que trataremos simplemente de entender cómo funcionan, sin entrar en detalles. Pero antes de entrar a una descripción más profunda de protocolos seguros, debemos hablar un poco de criptografía para poder entender conceptos como algoritmo de encriptación, firma digital, criptografía simétrica o asimétrica...

La palabra criptografía deriva del griego **kriptos** (oculto) y **graphos** (escribir): el arte de escribir mensajes ocultos. La criptografía fue considerada un arte hasta que **Shannon publicó en 1949 la “Teoría de las comunicaciones secretas” [SHA49]**, cuando empezó a considerarse como una ciencia aplicada que se relaciona con otras como la estadística, la teoría de números, la teoría de la información y la teoría de la complejidad computacional.

Entendemos como **criptografía** la ciencia que sirviéndose de cálculos matemáticos y computacionales, es capaz de **transformar un mensaje legible** (que denominamos **texto en claro**) **en otro que sólo sea comprensible por determinadas personas** (que denominamos **criptograma**). Esta transformación se basa en unos fundamentos y procedimientos de aplicación que denominamos **criptosistema**. En base a ese criptosistema se desarrolla un algoritmo (entendiendo algoritmo como especificación unívoca de los pasos a seguir para realizar una actividad) que detalla la forma de aplicar estos cambios. Este algoritmo se denomina **algoritmo de encriptación**.

**La criptografía se complementa con el criptoanálisis**, otra rama de la ciencia que estudia la técnica de **descifrar textos encriptados** sin autorización para ello. El conjunto de criptografía y criptoanálisis es lo que se conoce como **criptología**.

La criptografía clásica usaba métodos como la sustitución o la trasposición para alterar los mensajes, pero con la llegada de las máquinas de cálculo (y no sólo los ordenadores como los entendemos hoy) esto cambió. Se hizo necesario **aumentar la complejidad de los criptosistemas**, para lo cual se ideó un sistema mediante el cual **una palabra o conjunto de caracteres implementaba un mecanismo de sustitución o permutación que complementaba al algoritmo de encriptación**. Estas palabras se denominan **claves de encriptación**, y la cantidad de caracteres que poseen se denomina **longitud de la clave**. A mayor longitud de la clave, mayor complejidad del criptosistema.

Un ejemplo aplicado de esta teoría sería la aplicación de los conceptos de la criptografía en los sistemas informáticos (que al fin y al cabo es lo que nos interesa).

- El **texto en claro** podría ser **cualquier información** (binario, hexadecimal...) que fuera legible.
- El **criptograma** sería cualquier **información encriptada**.
- El **criptosistema** sería el **conjunto de operaciones computacionales y matemáticas** que realizan los ordenadores.
- El **algoritmo de encriptación** es **cualquiera debidamente implementado** en un sistema informático. Más adelante trataremos los diversos algoritmos de encriptación.
- La **clave de encriptación** es **cualquier información que controla la unicidad de un sistema criptográfico** y lo distingue de los demás que se basan en el mismo algoritmo.
- La **longitud de la clave** es **el tamaño de esa información, y se mide en bits**.

Queda algo por ver. Una clave de encriptación tal y como se ha explicado antes sirve para encriptar y desencriptar una información mediante un determinado algoritmo. **Esta clave “tradicional” se denomina clave simétrica**. Por contra, **existe otro sistema que idearon los matemáticos Diffie y Hellman en 1976 [DIH76]** y que denominamos **criptografía asimétrica**. La criptografía asimétrica es un tema complejo que trataremos adecuadamente más tarde, pero de momento para que entendamos los protocolos seguros baste decir que **estos sistemas usan dos claves de encriptación: una denominada clave pública que permite la encriptación de la información y otra que denominamos clave privada y que permite la desencriptación**. En estos sistemas existe un **concepto nuevo denominado firma digital**, y que es procedimiento que encripta a la clave privada un hash de algún tipo de información, de forma que con la clave pública pueda ser desencriptado y comprobado por comparación. Ahora aparcamos para más adelante la criptografía asimétrica y las firmas digitales y empezamos con los protocolos seguros. ;-)

El protocolo **SHTTP (Secure Hypertext Transfer Protocol)** nació de la mano de **Enterprise Integration Technologies (EIT)** y es una versión modificada del HTTP normal que incluye características de seguridad. Esta extensión trabaja debajo del HTTP y se ubica en la capa de aplicación del modelo TCP/IP. Las ampliaciones de SHTTP comprenden la codificación de documentos web así como soporte para firmas digitales, y la posibilidad de que el cliente verifique la integridad de un mensaje mediante el **MAC (Message Authentication Code)**.

SHTTP gestiona las negociaciones entre cliente y servidor mediante texto formateado. Este texto tiene **dos partes**: el **cuerpo del mensaje (firmado y codificado)** y la **cabecera** que contiene los datos sobre cómo se ha codificado e instrucciones para que el cliente descifre la información. Para crear el mensaje, el servidor integra las preferencias de seguridad del cliente con las suyas, es decir, si el servidor acepta varios sistemas de codificación como **PKCS-7 (Public Key Codification Standard 7)**, **RSA (Rivest-Shamir-Adleman)**, **DH (Diffie-Hellman)**... y el cliente tiene como primera preferencia el sistema PKCS-7, éste será el sistema de cifrado usado.

En primer lugar tiene lugar el proceso de intercambio de las claves a usar. Existen varias formas de que esto ocurra:

- **Intercambio de claves "fuera de banda"**: El servidor y el cliente han acordado e intercambiado de antemano y por otro canal las claves de sesión. En este caso no tiene lugar ningún nuevo intercambio de claves y se inicia la sesión SHTTP.
- **Intercambio de claves "en banda"**: El cliente envía al servidor su clave pública. El servidor comprueba su lista de sistemas criptográficos para comprobar si puede ser procesado, en cuyo caso encripta la clave de sesión con la clave pública del usuario y se la envía, dando lugar al **inicio de sesión con la clave de sesión**. En caso de que el servidor no pueda procesar la clave pública del cliente, se da por finalizada la conexión.
- **Claves criptográficas predefinidas**: Claves RSA o Krb prefijadas que sirven para la codificación de la clave de sesión a utilizar

Así mismo ambos hosts pueden y deben **verificar la integridad** del mensaje y la autenticidad del remitente. La **integridad** del mensaje se comprueba mediante la **comprobación del valor hash relacionado con la clave** (más adelante hablaremos de las funciones criptográficas de tipo hash). La **autenticidad** del remitente se comprueba mediante los pertinentes **algoritmos de firma digital** que implementa SHTTP. Estas medidas de seguridad nos permiten comprobar que nadie haya modificado el documento en el canal de transmisión.

La parte más importante de un paquete SHTTP es su cabecera **MIME (Multi-purpose Internet Mail Extensions)**. MIME es un sistema de codificación (es importante **no confundir codificación con encriptación**: codificación es un medio para representar la información, en este caso texto codificado en bits) que implementa tanto **BASE64** como **UUENCODE (Unix to Unix Encode)**. Veamos los campos que contiene esta cabecera:

CAMPO	DATOS Y ALGORITMOS
<b>Dominios privados SHTTP</b>	Especifica la clase de algoritmos de cifrado, así como la forma de encapsulamiento de los datos: <i>PEM (Privacy Enhanced Mail)</i> o <i>PKCS-7 (Public Key Codification Standard 7)</i> .
<b>Tipos de certificado SHTTP</b>	Especifica el formato de certificado aceptable, actualmente X.509.
<b>Algoritmos de intercambio de claves SHTTP</b>	Indica los algoritmos que se usarán para el intercambio de claves: <i>RSA, fuera de banda, en banda</i> o <i>Krb</i> .
<b>Algoritmos de firmas SHTTP</b>	Especifica el algoritmo para la firma digital: <i>RSA</i> o <i>NIST-DSS (NIST Digital Standard Signature)</i> .
<b>Algoritmos de hash del mensaje SHTTP</b>	Identifica el algoritmo que proporciona integridad a los datos usando funciones hash: <i>RSA-MD2, RSA-MD5</i> o <i>NIST-SHS</i> .
<b>Algoritmos de contenido simétrico SHTTP</b>	Especifica el algoritmo simétrico de cifrado en bloque usado para cifrar los datos: <i>DES-CBC, DES-EDE-CBC, ES-EDE3-CBC, DESX-CBC, IDEA-CFB, RC2-CBC, RC4</i> o <i>CDMF</i> .
<b>Algoritmo de cabecera simétrica SHTTP</b>	Algoritmos de cabecera simétrica de S-HTTP, que proporciona una lista del cifrado de clave simétrica utilizada para cifrar las cabeceras: <i>DES-ECB, DES-EDE-ECB, DES-EDE3-ECB, DESX-ECB, IDEA-ECB, RC2-ECB</i> o <i>CDMF-ECB</i> .
<b>Mejoras de intimidad de SHTTP</b>	Especifica las mejoras en la intimidad asociadas con los mensajes, como firmar, cifrar o autenticar.

El protocolo **SSL (Secure Socket Layer)** fue ideado por Netscape Communications Corporation. Hoy en día es, con diferencia, mucho más usado que SHTTP. Sus **principales ventajas** son:

- El protocolo SSL **es abierto**, mientras el protocolo SHTTP es propietario.
- SSL **proporciona compatibilidad con cualquier protocolo** del nivel de aplicación al funcionar como capa del modelo TCP/IP a nivel de socket, mientras que SHTTP solamente trabaja con HTTP, puesto que es una ampliación de este.
- SSL **es transparente al usuario y al protocolo**, por lo que es **compatible con firewalls** y **permite conexiones tunneling** (se denominan conexiones tunneling a las que nos permiten acceder a redes WAN e Intranets desde Internet).
- SSL **utiliza extensiones S-MIME (Secure Multi-purpose Internet Mail Extensions)** para transmitir datos seguros.

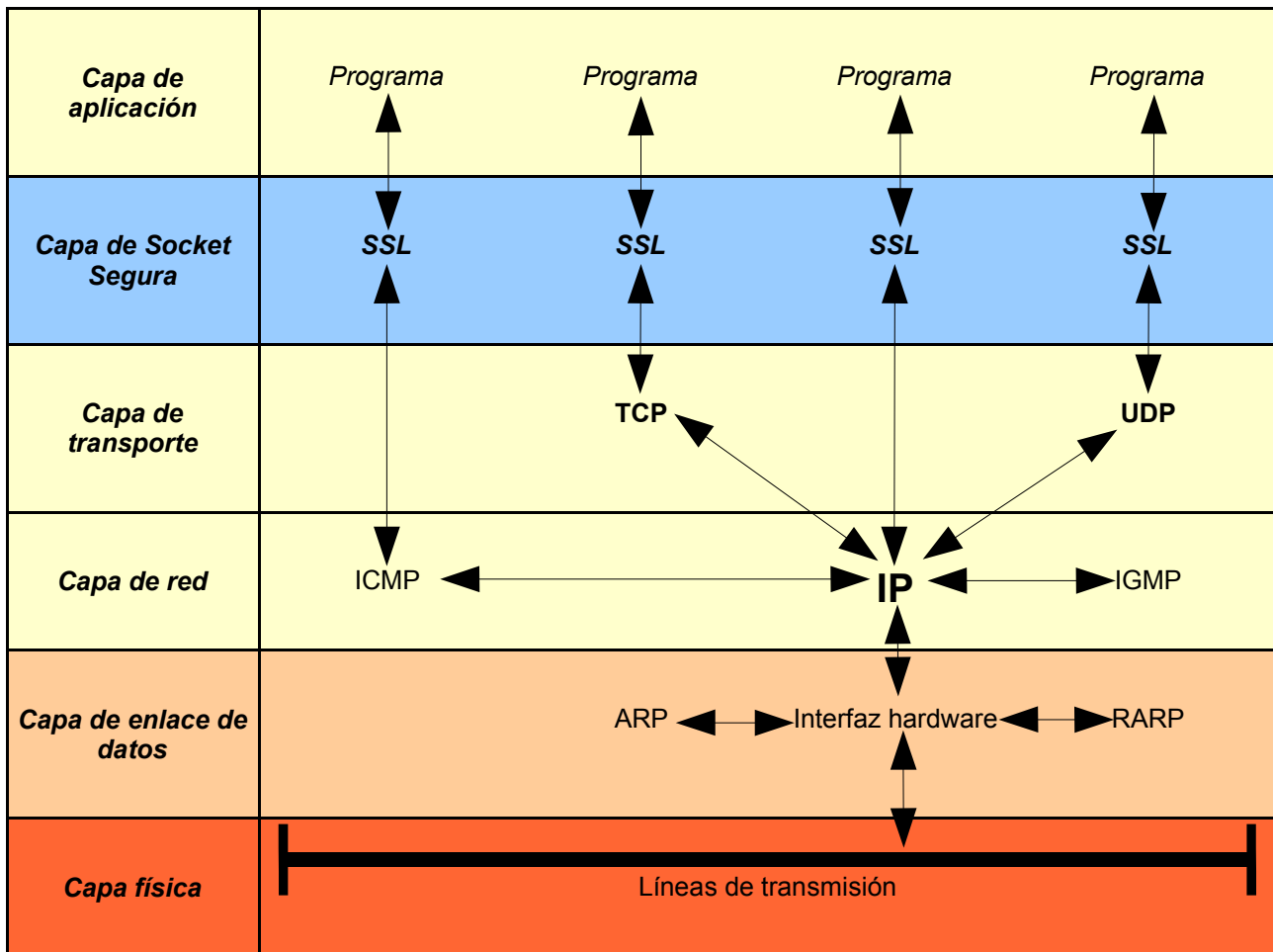
SSL nació ante la necesidad de dotar a protocolos habituales que no eran HTTP (FTP, SMTP, POP3...) de seguridad en las comunicaciones mediante un protocolo seguro. Este protocolo es abierto no propietario, por lo que cualquiera puede desarrollar aplicaciones seguras bajo SSL, lo que unido a esa compatibilidad multi-protocolo, ha aumentado su uso enormemente. Actualmente se trabaja con el SSL versión 3.0.

El protocolo SSL **se implanta como una capa más de la pila TCP/IP** entre la capa de aplicación y todas las demás. Esta capa actúa de forma transparente a las aplicaciones porque funciona a nivel de socket.

Un **socket** es, desde el punto de vista de la red, el **punto final de un enlace de comunicación** de dos vías entre dos programas que se ejecutan a través de la red. Desde el punto de vista de un programa, un socket es la **subrutina encargada de gestionar las conexiones de red**. Existen dos tipos de sockets:

- **Orientado a conexión:** se establece un camino virtual fiable cliente-servidor. La información llega en el orden en que se envió. El cliente abre una sesión en el servidor y éste **recuerda el estado** del cliente.
- **No orientado a conexión:** se usa para envío de datagramas de tamaño fijo, con lo que eso implica (pérdida de información, alteración del orden...). **No se guardan estados**.

Integración de Secure Socket Layer 3.0 en la pila TCP/IP:



SSL usa como **algoritmo de encriptación de sesión DES, triple DES, RC2, RC4 o IDEA (International Data Encryption Algorithm)**. Estas claves de sesión son a su vez encriptadas mediante mecanismos de clave pública (típicamente **RSA**) para ser enviadas al inicio de sesión. El algoritmo de hash usado es **MD5 o SHA**.

Las **fases del inicio de una conexión SSL** son las siguientes:

- **Hello:** El cliente envía el mensaje *Client Hello* al servidor con su clave pública y sus preferencias criptográficas. El servidor verifica que pueda procesarlas, y en caso contrario cierra la sesión.
- **Intercambio de claves:** El servidor ya ha verificado que puede procesar las claves y que las preferencias criptográficas suyas y del cliente son compatibles. El servidor envía su clave pública en el mensaje *Server Hello*.
- **Producción de la clave de sesión:** Se genera en el servidor una clave única para la sesión que se destruirá una vez concluida y se envía al cliente encriptada con su clave pública.
- **Verificación del servidor:** Sólo se da cuando se usa el algoritmo RSA, y en ella el cliente debe autenticar al servidor.
- **Autenticación del cliente:** El servidor exige al cliente un certificado de seguridad X.509v3, sólo en caso de que sea necesario.
- **FIN:** Indica que ya puede comenzar la conexión segura.

El paquete de datos es **encapsulado en tres partes** (en SHTTP era en dos): **MAC-DATA**, código de autenticación del mensaje; **ACTUAL-DATA**, datos que se transmiten y **PADDING-DATA**, campo de datos requerido en caso de usar cifrado en bloque.

SSL está muy extendido hoy en día en Internet, y **genera variaciones de los protocolos inseguros tradicionales para nuevos protocolos seguros que corren bajo conexiones SSL**, como por ejemplo: HTTP -> HTTPS; FTP -> FTPS...

El uso de conexiones seguras, sean del tipo que sean, asegura en mayor o menor medida la comunicación entre el cliente y el servidor, pero... **¿cómo podemos saber que el servidor es seguro?** Para ello existe el **mecanismo denominado Autoridades de Certificación (AC)**, que son las encargadas de emitir **certificados de seguridad** a servidores o usuarios, de forma que podamos estar seguros de que nos estamos comunicando con quien queremos.

Un **certificado de seguridad** se expide a una entidad cuando se comprueba que ésta **cumple una serie de requisitos de seguridad** con el fin de poder asegurar las conexiones de protocolos seguros. Un certificado de seguridad **se compone de la clave pública del servidor** junto a unos cuantos datos básicos, que es **firmada con la clave privada de la autoridad de certificación**. Estas firmas siempre tienen una caducidad, y han de renovarse periódicamente mediante nuevos controles a la entidad solicitante.

Cuando establecemos por ejemplo una conexión en nuestro navegador mediante SSL y entramos en el protocolo HTTPS, en la parte inferior del navegador aparece una pequeña llave o candado (dependiendo del software que usemos). Ese icono indica la presencia de un certificado de seguridad y un protocolo seguro, pero **es muy importante revisar ese certificado** y comprobar que no esté caducado (aunque esta eventualidad suele advertirla el software navegador automáticamente), que los datos de la entidad sean correctos y que la firma digital de la autoridad de certificación sea válida. Yo recomiendo siempre que vayamos a tratar información sensible, como compras en Internet o cualquier gestión con números como tarjetas de crédito, la revisión del certificado de seguridad.

Una conexión mediante protocolo seguro complementada con un certificado de seguridad puede asegurarnos tranquilidad a la hora de realizar movimientos de información comprometida en Internet. Para obtener información más detallada sobre certificados de seguridad y obtención de uno, es útil visitar la web de **Verisign** (<http://www.verisign.com/>).

## **- Criptografía y autenticación de ficheros: MD5**

Ya hemos hablado brevemente antes de las funciones criptográficas de tipo hash y del algoritmo MD5, pero ahora queremos conocer un poco mejor su funcionamiento y su utilidad práctica, pues en el caso de checksums de paquetes de protocolos seguros, nosotros no manejamos ni comprobamos el **MD5**, pero puede resultarnos útil para otras funciones de autenticación... pero vamos a ver primero un poco de teoría.

Una **función criptográfica tipo hash** acepta como **entrada una cadena de información** y devuelve como **salida una cadena de caracteres denominada fingerprint, huella digital, hash o resumen**. Las características de las funciones hash son:

- **Unidireccional:** Conocido un hash, es computacionalmente imposible la reconstrucción del mensaje original.
- **Compresión:** A partir de un mensaje de cualquier longitud se obtiene un hash de un tamaño fijo, normalmente menor que el del mensaje original.
- **Difusión:** El resumen es una función compleja de todos los bits del mensaje.
- **Colisión simple:** Se conoce como resistencia débil a las colisiones el hecho de que dado un mensaje cualquiera, es computacionalmente imposible encontrar otro mensaje cuyo hash sea igual.
- **Colisión fuerte:** Se conoce como resistencia fuerte a las colisiones el hecho de que sea computacionalmente difícil encontrar dos mensajes cuyo hash sea idéntico.

En el caso aplicado de **MD5 (Message Digest 5)**, se trata de una **función criptográfica tipo hash** que **acepta como entrada una cadena de texto** (un fichero puede ser convertido a cadena de texto mediante MIME) y **devuelve un número de 128 bits**. Habrá quien piense que 128 bits no es un número muy alto y que pudiera darse el caso de que dos ficheros tuvieran un mismo hash MD5. Es prácticamente imposible que esto ocurra, pues aunque 128 bits parezca una cantidad pequeña, **representa  $3,4 * 10^{34}$  combinaciones** diferentes (más de trescientos sextillones de combinaciones diferentes), además de ser una de las características definitorias de una función hash: la colisión fuerte.

**MD5 fue ideado en 1992 por Ron Rivest** como ampliación a los algoritmos MD4 y MD2, siendo MD5 mucho más seguro y computacionalmente algo más lento, si bien no requiere grandes tablas de sustitución y se puede cifrar absolutamente compacto.

El otro gran algoritmo de hash que existe hoy día es el **SHA-1 (Secure Hash Algorithm 1)**. Las principales diferencias entre SHA-1 y MD5 son:

- **SHA-1 genera una salida de 160 bits frente a los 128 bits de MD5**, lo cual le hace más resistente a colisiones fuertes y débiles.
- **SHA-1 realiza mayor número de operaciones y maneja un número mayor de bits en el hash**, por lo que resulta computacionalmente más complejo.
- **La longitud máxima del mensaje para SHA-1 es  $2^{64}$  bits (2 PB)**, mientras que **MD5 no tiene limitación** en ese sentido.

El hash MD5 suele presentarse como un **número de 128 bits en representación hexadecimal**, como por ejemplo: **3F30FA256D2B82EC4DAA111ACF35977F, B83C221E0A1908B7910113AAAE6B6DADC**.

Al representarse mediante hexadecimal, la cadena de hash queda muy reducida y es muy fácil de manejar en presentaciones web, por ejemplo. En Internet está muy extendido el uso de hash MD5 en la distribución de ficheros, y es el algoritmo de hash que siempre uso para autenticar mis documentos. A la hora de descargar ficheros importantes para el ordenador, como una distribución de Linux, un driver, un parche al kernel... es importante verificar el MD5 para asegurarnos que hemos descargado lo que queríamos descargar antes de instalar nada.

Para poder comprobar las sumas MD5, necesitamos algún software que lo implemente: **en sistemas Linux disponemos de md5sum**, un paquete que viene prácticamente con todas las distribuciones; y **en sistemas Windows tenemos MD5 Command Line Message Digest Utility** (<http://www.fourmilab.ch/md5/>) y la implementación para win32 de **md5sum** (<http://www.etree.org/md5com.html>).

## **- Criptografía asimétrica: el sistema PGP**

Estos sistemas iniciaron sus andaduras por Internet gracias a Diffie y Hellman (DH), los matemáticos que idearon el sistema de clave pública, pero fue con el algoritmo de Rivest, Shamir y Adleman (RSA) cuando se comenzó a usar de una forma mucho más habitual, pues RSA además de ser un sistema de clave asimétrica, era también un sistema criptográfico de firma digital. Más tarde gracias a la combinación de DH con el Digital Standard Signature (DSS) nació la DH/DSS que combinaba la criptografía de clave asimétrica de DH con el sistema criptográfico de firma de DSS.

Era solamente cuestión de tiempo que estos sistemas se implementaran como algo cotidiano al usuario de Internet, y ahí nació el sistema PGP...

### **Introducción a la clave asimétrica: sistema PGP/gnuPG**

Un criptosistema de clave pública necesita dos claves relacionadas y complementarias. Una de ellas, denominada **clave pública**, es distribuida libremente y permite encriptar mensajes al destinatario propietario de esa clave. La segunda clave es denominada **clave privada** y permite desencriptar los mensajes previamente cifrados con la clave pública. Ambas claves son complementarias y serían perfectamente intercambiables. Existe un tercer mecanismo añadido a los dos anteriores, el **sistema de firma digital** que consiste en una función criptográfica de tipo hash que aplicada un mensaje y encriptada a la clave privada da lugar a un bloque conocido como firma digital. Pero veamos un poco de historia.

Todo comenzó en **1976**, cuando **Whitfield Diffie** y **Martin Hellman** [DHF76] desarrollan el algoritmo **DH (Diffie-Hellman)**. Esto supuso el inicio de la criptografía asimétrica. Dos años después, en **1978**, **Ron Rivest**, **Adi Shamir** y **Leonard Adleman** [RSA78] idearon el potente algoritmo **RSA (Rivest-Shamir-Adleman)** en el **Massachusetts Institute of Technology (MIT)**. El algoritmo era tan potente que la **NSA (National Security Agency)** de los Estados Unidos recomendó que no fuera publicado. Los tres científicos hicieron caso omiso y lo publicaron en la revista *Scientific American*, en parte por miedo a que esa recomendación pasara a ser una prohibición total. El algoritmo se patentó a nombre de la compañía **RSA Data Security Inc.**, patente sólo válida en Canadá y Estados Unidos. Varios años después Ron Rivest ideó el algoritmo de hash **MD5 (Message Digest 5)** que ya hemos visto.

En 1991 surgen rumores acerca de una posible prohibición de la criptografía en las comunicaciones en EEUU, por lo que el programador **Philip Zimmermann** ideó un sistema criptográfico que aúna los siguientes elementos: el mejor algoritmo disponible de clave simétrica, **IDEA**; el mejor algoritmo de clave asimétrica, **RSA**, el algoritmo de hash **MD5** y un generador de números aleatorios estándar. Ese sistema se llamaba **PGP (Pretty Good Privacy)** y fue inmediatamente distribuido como freeware, lo que propició que rápidamente se convirtiera en el estándar en Internet en cuanto a encriptación y firma digital, con lo que el gobierno tuvo que dar marcha atrás: era demasiado tarde para detener el avance de **PGP**.

Como anécdota cabe señalar que en Junio de ese mismo año se distribuyó a través de USENET una copia de PGP que comenzó a ser **distribuido y usado fuera de Estados Unidos**, con lo cual se incumplía la legislación **ITAR** del Departamento de Estado acerca de exportación de armas. Y es que en Estados Unidos la criptografía es considerada un arma de igual magnitud que tanques, armas químicas o armamento pesado. Además, RSA Data Security Inc. reclamaba a Zimmermann el copyright del algoritmo RSA usado en PGP.

Se decidió el desarrollo paralelo de **dos versiones de PGP**, una para **Estados Unidos y Canadá donde se usa la biblioteca de funciones RSAREF**, y otra para **el resto del mundo** que distribuye el MIT y que **usa la biblioteca MPILIB**. Para la solución de la violación de ITAR Zimmermann tuvo que esperar hasta 1996, cuando una escueta carta de dos líneas y un comunicado de prensa **cerraron definitivamente el caso** sin volver a hacer declaraciones. Ese mismo año **fue absuelto de la supuesta violación de patentes** de RSA y Zimmermann fundó su propia compañía, **Pretty Good Privacy Inc.** Al cabo de los años se ha sabido que PGP salió también de otra forma de Estados Unidos: en forma de 12 tomos de código fuente que fueron reescritos y recompilados en Europa.

Han habido varias inclusiones de nuevos algoritmos, como **AES (Advanced Encryption Standard)** y **DES (Data Encryption Standard)** como algoritmos de encriptación simétrica en complemento a IDEA (cuya patente está aún vigente); **SHA-1** como algoritmo de hash complementando a MD5 o **DH/DSS (Diffie-Hellman/Digital Standard Signature)** como algoritmo de encriptación asimétrica junto a RSA. La inclusión de DH/DSS tuvo lugar en la versión 5.0 y sus motivos no eran de seguridad. **La patente de RSA no expiró hasta el año 2000** y DH/DSS suponía un algoritmo libre, con lo que se evitaban los continuos pleitos con RSA Data Security Inc. **PGP/GPG hoy día es compatible con ambos algoritmos**, pues RSA ya es libre, y cada cual puede elegir el tipo de clave que más le guste.

Ha pasado el tiempo y PGP ha avanzado mucho desde las versiones 2.x hasta las 8.x de hoy en día. Aunque PGP tiene una versión comercial con muchas funciones extras, como discos encriptados virtuales, **nunca se ha dejado de distribuir una versión freeware** del software para todos los usuarios que la deseen.

**gnuPG es una implementación libre y de código abierto (GPL) del sistema openPGP (RFC2440)**, capaz de manejar sus algoritmos (excepto IDEA, que aún es un algoritmo patentado.) y realizar las mismas funciones que PGP. Actualmente ambos se desarrollan paralelamente para varias plataformas.

Hoy en día el uso de PGP/GPG está principalmente orientado a la **encriptación y firma de ficheros y correos electrónicos**, así como la generación y gestión de claves mediante el sistema **keyring (anillo de claves)**, que almacena la información correspondiente a todas las claves públicas y privadas. PGP/GPG **gestiona las claves mediante ficheros de texto ASCII** que contienen la información de clave pública y/o privada y que sirven para el intercambio de las mismas.

Podemos **conseguir PGP** desde su **web internacional** (<http://www.pgpi.org/>) o desde la Americana en caso de residir en Estados Unidos o Canadá (<http://www.pgp.com/>). Podemos así mismo **descargar gnuPG** desde su web (<http://www.gnupg.org/>).

## Los algoritmos: RSA y DH/DSS

Ya conocemos la historia del sistema PGP, así como sus bases y los algoritmos que maneja. En este apartado hablaremos del corazón de este sistema, sus algoritmos criptográficos, y por último en el siguiente apartado hablaremos de su utilización. Aunque el sistema DH es anterior en el tiempo al sistema RSA, el algoritmo compuesto DH/DSS es posterior, y además su unión al sistema PGP también lo es.

El algoritmo **RSA (Rivest-Shamir-Adleman)** nació en 1978 de mano de **Ron Rivest, Adi Shamir y Leonard Adleman [RSA78]**. Su seguridad se basa en que no existe una forma eficiente de factorizar números que sean productos de dos grandes primos. El algoritmo RSA se basa en su definición en **cuatro propiedades**:

1. Al decodificar la forma codificada de un mensaje, se obtiene el mensaje original, lo que puede expresarse mediante la expresión:

$$D ( E ( M ) ) = M$$

Donde **D** representa la decodificación mediante clave privada, **E** representa la codificación mediante clave pública, y **M** representa un mensaje cualquiera.

2. **E** y **D** son relativamente fáciles de calcular.
3. Si se revela **E** de forma pública, no existe una forma fácil de calcular **D**. Por tanto, sólo el usuario que tiene el valor **D** puede decodificar un valor codificado con **E**.
4. Se se decodifica el mensaje **M** y se vuelve a codificar, se obtiene **M**, es decir, se ha de cumplir:

$$E ( D ( M ) ) = M$$

Echemos un vistazo al algoritmo (no es necesario comprenderlo, que nadie se asuste):

<b>1</b>	Escoger dos números primos muy grandes $p$ y $q$ (secretos) y calcular el número $n$ (público) correspondiente a su producto, $n = p * q$
<b>2</b>	Escoger la clave de descifrado constituida por un gran número entero $d$ (secreto), que es primo con el número $\Phi(n)$ (secreto) obtenido mediante: $\Phi(n) = (p-1) * (q-1)$
<b>3</b>	Calcular el entero $e$ (público) tal que $1 \leq e \leq \Phi(n)$ , mediante la fórmula: $e * d = 1 \pmod{\Phi(n)}$
<b>4</b>	Hacer pública la clave de cifado $(e,n)$
<b>5</b>	Para cifrar texto, es necesario previamente codificar el texto en un sistema numérico en base $b$ dividiéndolo en bloques de tamaño $j-1$ de forma que $b^{j-1} < n < b^j$
<b>6</b>	Cifrar cada bloque $M_i$ transformándolo en un nuevo bloque de tamaño $j$ $C_i$ de acuerdo con la expresión $C_i \equiv M_i^e \pmod{n}$
<b>7</b>	Para descifrar el bloque $C_i$ , se usa la clave privada $d$ según la expresión: $M_i \equiv C_i^d \pmod{n}$

La importancia de la longitud de las claves, como señalan Rivest, Shamir y Adleman, se pone de manifiesto en el cumplimiento de la tercera propiedad, pues **si alguien deseara descifrar** sin permiso un mensaje, **debería probar todas las claves posibles**. En números simples puede ser sencillo, pero hoy día las claves más pequeñas que se manejan son de **1024 bits, lo que supone  $1,8 * 10^{308}$  claves distintas**, y el algoritmo RSA en sistema PGP hoy día soporta claves de hasta **4096 bits, lo que supone  $1 * 10^{1233}$  claves**, una cantidad a todas luces descomunal (yo uso una clave RSA/4096).

El algoritmo **DH/DSS (Diffie-Hellman/Digital Standard Signature)** tiene una larga historia... **DH (Diffie-Hellman)** nació de la mano de **Dres. W. Diffie y M.E. Hellman [DIH76]** en 1976 como un algoritmo de intercambio de claves. Según los propios Diffie y Hellman, el **sistema del logaritmo discreto** (en que se basa DH) resultaba intratable como sistema criptográfico de clave pública completo. Varios años después, **en 1985, ElGamal [ELG85] demostró y presentó que era posible desarrollar el sistema completo** a partir de DH. Ese sistema ideado por ElGamal es lo que hoy se conoce como DH. DH resultaba un sistema de cifrado que al contrario de RSA, no podía adaptarse a la firma digital. **ElGamal ideó un sistema de firma** para DH que no podía adaptarse para cifrado pero que complementaba al algoritmo. El problema de ese sistema de firma es que **duplicaba la longitud del mensaje firmado**, lo que resultaba extremadamente ineficiente. **En 1991 el NIST (National Institute of Standards and Technology) ideó una variante** del sistema de firma de ElGamal que corregía dicho problema y **se denominó DSS (Digital Standard Signature)**. Así, en 1991, nació el algoritmo actual DH/DSS.

Es importante tener en cuenta que DH/DSS cuenta con cuatro tipos de claves: **secreta, pública, de cifrado y de descifrado**. Veamos el algoritmo propiamente dicho de DH/DSS:

<b>1</b>	Serán de información pública un número primo grande $p$ y $a$ , una raíz primitiva de <b>1 módulo <math>p</math></b> (es decir, tal que $a^{p-1} \equiv (mod p)$ y $a^{d-1} \not\equiv (mod p)$ para todo $d$ tal que $1 < d < p$ )
<b>2</b>	La clave privada del usuario B es un entero $k_B$ escogido dentro del intervalo $[1, p-1]$
<b>3</b>	La clave pública de B es el entero $K_B \equiv a^{k_B} (mod p)$
<b>4</b>	Se supone que el emisor A quiere enviar un mensaje $M$ ( $1 \leq M \leq p-1$ ) al receptor B
<b>Proceso de Cifrado (e)</b>	
<b>1e</b>	Escoger aleatoriamente un entero $k_A$ tal que $1 \leq k_A \leq p-1$ , que constituye su clave secreta
<b>2e</b>	Calcular la clave de cifrado a partir de su propia clave privada y la clave pública de B, $Q \equiv K_B^{k_A} (mod p)$
<b>3e</b>	Cifrar el mensaje M según la expresión $C \equiv Q * M (mod p)$
<b>Proceso de Descifrado (d)</b>	
<b>1d</b>	Obtener Q gracias a la clave pública de A, $K_A$ , y a su propia clave secreta mediante la fórmula $Q \equiv K_A^{k_B} (mod p)$
<b>2d</b>	Recuperar M a partir de $M \equiv (Q^{-1})_p * C (mod p)$ donde $(Q^{-1})_p$ denota el inverso de Q en módulo p

El sistema de firmas digitales en ambos sistemas se basa en **aplicación de una función hash al mensaje a firmar y su encriptación a la clave privada** del firmante. Esto garantiza que únicamente el dueño de la clave (y poseedor del password correspondiente) pueda firmar el mensaje. Al ser recibido, se descifra con la clave pública del firmante y se vuelve a aplicar la función hash al texto firmado. Si ambos hashes son idénticos, la firma es válida.

Las firmas RSA poseen un defecto de base algorítmica mediante el cual conocidos dos mensaje y sus firmas, es posible crear la firma de un tercer mensaje mediante aritmética modular, aunque es importante señalar que esto **solamente es posible en mensajes exclusivamente numéricos**.

Hace poco tiempo se descubrió igualmente un **fallo de implementación en gnuPG de las firmas ElGamal** que compromete la clave privada de un sistema firma+cifrado ElGamal. Afortunadamente las claves que se crean **hoy día y desde hace bastantes años, usan el sistema de firma DSS**.

Queda por hablar del eterno debate: ¿RSA o DH/DSS? Ambos sistemas basan su seguridad en la enorme dificultad que entraña calcular la clave privada a partir de una clave pública. En RSA esta fuerza se basa en la **dificultad de encontrar los factores primos de un entero muy grande**, mientras que en DH/DSS dependen de la dificultad de **calcular los logaritmos únicos en un campo finito generado por un número primo de gran tamaño**.

En sí, hablando de la base teórica matemática, el problema del logaritmo único es más complicado de resolver, así como RSA requiere cálculos más complejos. Por contra, el sistema de firma DSS solamente soporta 1024 bits de longitud de clave, mientras que una firma RSA soporta 4096. En definitiva, aún basándose en principios matemáticos distintos, ambos sistemas criptográficos resultan igual de seguros y poderosos a la hora de usarlos como mecanismos de cifrado. La conclusión, como en todo, es que para gustos están los colores y cada cual debe elegir el tipo de clave que prefiera.

## Encriptación de ficheros, correo electrónico y comunicaciones

Los algoritmos RSA y DH/DSS que maneja PGP han demostrado en los últimos tiempos ser relativamente inmunes a los ataques de fuerza bruta, pues utilizan un número de combinaciones de claves públicas y privadas descomunal. Un ordenador con un procesador a 3 Gigaherzios necesitaría tres años ininterrumpidos de trabajo para romper por fuerza bruta una clave RSA de 428 bits. Hoy en día ninguna clave RSA manejada por un sistema criptográfico maneja menos de 512 bits, y en el caso de PGP el número mínimo de bits es de 1024, llegando a 4096. Pero sabemos que ningún sistema es completamente seguro, y es posible que futuras tecnologías computacionales (como los ordenadores cuánticos) o teorías matemáticas permitan la ruptura de los actuales sistemas criptográficos (incluyendo el **potente sistema de Miller [MIL86] ideado en 1986 y basado en el uso de curvas elípticas**), pero hoy por hoy podemos decir que PGP es un sistema seguro.

PGP introduce el concepto de **anillo de claves o keyring**. Este sistema gestiona los ficheros de claves públicas y privadas de un sistema, y permite la generación de una jerarquía de confianza basada en la posibilidad de firmar claves públicas con nuestra propia clave privada. Al firmarse mutuamente dos usuarios sus claves públicas con las privadas, establecen un vínculo de confianza entre ellos, que se va extendiendo (si yo confío en ti y tú confías en él, yo confío en él...). Este sistema permite igualmente la gestión de encriptación o firma dirigida a múltiples claves públicas, de forma que podamos cifrar un correo electrónico con más de un destinatario.

El sistema **PGP gestiona las claves para su exportación e importación mediante ficheros de armadura de texto ASCII** que contienen la clave y que pueden ser reconocidos e importados por un sistema compatible. Así mismo, los mensajes encriptados y firmas se representan mediante texto ASCII que es enviado e interpretado por el receptor. Veamos algunos formatos de distintos mensajes PGP:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
<Comentario>

<Clave>

-----END PGP PUBLIC KEY BLOCK-----

-----BEGIN PGP SIGNED MESSAGE-----
<Tipo de hash>

<Mensaje en claro>

-----BEGIN PGP SIGNATURE-----
<Comentario>

<Firma>

-----END PGP SIGNATURE-----

-----BEGIN PGP MESSAGE-----
<Comentario>

<Mensaje encriptado>

-----END PGP MESSAGE-----
```

En el campo de comentario puede haber cualquiera que el usuario decida colocar, o a veces un comentario que el propio software incluye indicando la versión del mismo usada (por ejemplo, *Version: GnuPG v1.3.4 (GNU/Linux)* que es el comentario de mi clave). En el campo hash se incluye el tipo de hash usado: MD5, SHA-1... En el campo clave, mensaje encriptado o firma van los símbolos ASCII que componen la clave, mensaje o firma. Pensaba haber pegado mi clave pública para que sirviera de ejemplo, pero a este tamaño de fuente ocuparía más de una hoja y media (RSA/4096).

Las propias claves **se marcan con una huella digital o fingerprint**, que es un hash que **identifica a la propia clave pública de forma unívoca**, y que es usado principalmente para comprobar la autenticidad de claves descargadas de un servidor de claves o anillo de claves público. Este fingerprint suele representarse mediante un número hexadecimal (de 128 bits para MD5 y de 160 bits para SHA-1) aunque actualmente PGP también lo representa como un conjunto de palabras. Recordemos que **128 bits son  $3,4 * 10^{38}$  combinaciones**, y **160 bits son  $1,5 * 10^{48}$** , cifras lo suficientemente elevadas como para que no se produzcan colisiones, especialmente en el caso de SHA-1. Como ejemplo de fingerprint pongo el correspondiente a mi clave pública:

`6FAB 9799 C61A A7D2 A409 8A53 6F6F 3938 AF95 93E1`

De esta forma si alguien descargara mi clave, podría comprobar que el fingerprint efectivamente es ese y así estaría seguro de que es la clave correcta. Para ello no está de más que ese fingerprint se entregue en persona o por un medio seguro, así el vínculo de confianza es totalmente cerrado.

Hoy en día el sistema PGP se integra perfectamente en los principales sistemas del mercado, incluyendo **Microsoft Windows, MS-DOS, GNU/LINUX, Unix, MacOS X, Amiga, BeOS, OS/2, PalmOS, EPOC...** y muchos otros. Por tanto la encriptación, firma, desencriptación y comprobación de ficheros funciona perfectamente en cualquier plataforma. Además, es común hoy en día la **compatibilidad de los clientes de correo electrónico con el sistema PGP**, de forma que no sea necesario encriptar o firmar a mano los mensajes desde ficheros de texto. Entre los clientes de correo electrónico compatibles con el sistema PGP/GPG tenemos: **Microsoft Outlook Express, Microsoft Outlook, Mozilla Mail, Mozilla Thunderbird, Eudora, The Bat!, Ximian Evolution, Kmail...** y muchos otros, bien mediante soporte nativo o bien mediante la aplicación de algún tipo de plugin, como Enigmail en sistemas Mozilla.

También existen diversos tipos de software que permiten el uso de claves PGP como medio de encriptación de la información en comunicaciones a través de Internet, ya sea a través de túneles SSL o medios de conexión normales.

Mi recomendación es siempre encriptar los mensajes para destinatarios cuya clave pública poseamos, y por defecto firmar absolutamente cualquier mensaje que enviemos. Igualmente recomiendo el uso de **passphrases** (contraseñas que activan las claves privadas) que cumplan las características que indiqué sobre contraseñas seguras y que sean tan largas como sea posible, para evitar cualquier intento de ruptura por fuerza bruta. Yo cambio con asiduidad el passphrase, y actualmente tiene una longitud de 25 caracteres, si bien a partir de 10 caracteres podría considerarse suficientemente seguro.

Es importante **hacer copia de seguridad de las claves privadas que poseamos**, pues si bien las claves públicas pueden ser descargadas de los servidores de claves donde hayamos decidido alojarlas, las privadas no se alojan en ningún sitio que no sea nuestro disco duro, y en caso de desastre debemos mantener copias de seguridad de esa clave actualizada, pues de lo contrario perderemos absolutamente cualquier información cifrada con la misma. **Tanto PGP como GPG permiten la exportación de ficheros ASCII de claves privadas** con finalidad de copia de seguridad.

Por último, recomiendo poseer al menos dos claves criptográficas: una para usar en encriptación de correo electrónico o comunicaciones en general a través de Internet, y otra para usarla de forma local y personal en la encriptación de ficheros sensibles. Así en caso de que la clave usada en Internet se viera comprometida, los ficheros de nuestro disco duro seguirían siendo seguros.

Podemos concluir diciendo que la criptografía es uno de los medios más seguros de que disponemos para mantener información privada de cara al resto del mundo, así como autenticar nuestras comunicaciones de forma que sea imposible que nadie falsee nuestra identidad. El sistema PGP es, desde hace unos años, imprescindible en Internet.

## Concluyendo

Esto es todo lo que he podido contar en estos tres días, aunque me hubiera gustado poder contar muchas más cosas, pues si bien en estos tres días hemos aprendido mucho juntos, quizá no sea tanto en conocimientos informáticos como el haber aprendido a valorar la importancia de algo que a priori para muchos es un tema completamente banal: la seguridad informática. Esa es quizá la lección más importante que pueda obtenerse de todo esta cantidad de palabras, gráficos y textos que he aglutinado para tratar de extender el conocimiento.

Pocas cosas otorgan más satisfacción personal que ayudar a los demás, y quizá esto haya ayudado a alguien a sentirse más seguro en Internet, a conocer cómo funciona lo que antes era la "Caja de Pandora" que no debía ser abierta por los mortales, e incluso puede que haya animado a alguien a buscar más allá. A ver más allá de la mundanalidad de Internet, a escuchar esa voz interior que le pide saber más, que le pide saciar un ansia de conocimiento. No todo el mundo tiene porqué sentirlo: ni todos estamos hechos de la misma pasta, ni valemos para las mismas cosas. Yo nunca llegaré a ser cantante, eso está más que claro (podéis rezar para no tener que oírme cantar), pero hubo varios momentos en mi vida en que esa llamita de la curiosidad se encendió, y siempre agradeceré a ciertas personas el que, sabiéndolo o sin saberlo, provocaran eso en mí. Y, quién sabe, quizá yo haya provocado esa reacción en alguno de los asistentes a las conferencias o los lectores de este documento.

En cualquier caso, el fin principal de estas conferencias y este documento no era despertar la conciencia hacker (herida de muerte en los últimos tiempos gracias a la desinformación), sino distribuir un conocimiento. Este conocimiento es (y si no lo fuera, debería serlo) libre. No solo no deseo que mi "creación" sea celosamente guardada y sellada bajo el pesado yugo del copyright, sino que ha sido mi voluntad darle las alas de la libertad y que estas alas se llamen GPL (GNU General Public License).

Ahora todos aquellos que han aprendido algo, tienen el deber moral de hacer llegar ese conocimiento a aquellos que no lo poseen, porque todos hemos necesitado en algún momento de alguien que nos tendiera una mano. Así mismo, presto mi mano para hacer aquello que en ella esté y ayudar a la gente que lo necesite.

Ya lo he dicho, pero lo repito como conclusión general: El conocimiento nos hace libres.

**Death Master**

## Agradecimientos

En esta sección quiero hacer público mi agradecimiento a todos aquellos que de alguna manera, directa o indirectamente, me han ayudado en la creación de este documento. Bien sea aportando ideas, revisándolo, o simplemente porque son amigos que considero que lo merecen, no quería dejar pasar la ocasión de darles las gracias.

El primer lugar corresponde por derecho propio a **mi novia Laura**, que hasta se ha leído el documento completo (jeje). También quiero añadir en un lugar destacado a **Iván**, mi hermano. Gracias. ;-)

En segundo lugar quiero lanzar un agradecimiento a ciertos grupos o colectivos de Internet que pertenezco:

A la comunidad de jugadores **Zona-Hispana** y a todos los miembros del clan Rey. Mucho juego y tiempo he compartido con la mayoría de ellos. Gracias chic@s.

A la comunidad de seguridad informática **HpN**, que resurgió de las cenizas de HackingParaNovatos. Más que un foro, más que una comunidad "hacker" o "de seguridad informática" o como queramos llamarla. Es un grupo de amigos, y os doy las gracias a todos los compañeros moderadores, colaboradores y miembros. Gracias chic@s.

A la recién nacida comunidad de software libre **sigT** (también conocida como Club Computacional de Vagos Informáticos :-P). Aún somos jóvenes, pero ésta es la clase de comunidad de software libre de la que siempre quise formar parte. Gracias chic@s.

También quiero incluir una comunidad de la que no soy miembro en el concepto de responsabilidad que implican las anteriores, si bien me siento miembro del grupo de gente, pues al igual que dije en HpN, esto es más que una comunidad hacker. Me refiero a la gente de **Hack X Crack**. Sé que aquí se puede aprender y enseñar mucho. Gracias chic@s.

En tercer y último lugar quiero dar los agradecimientos personales que van más allá de los colectivos anteriormente citados. Como aquí nadie es más importante que nadie, y como hay que elegir una manera de ordenar esto, creo que lo haré alfabéticamente mediante el nick -o el nombre en su defecto-. Con muchos de vosotr@s comparto más que un nick y una amistad por cables, pero creo que el nick bastará para que os deis por aludidos. Allá va:

**Abián, AcidBorg, Anakin2001, BraSSoY, Cripto, Dark\_Archangel, Dragons, Elena, Exley, Fisterran, Kamikaze, KaSiDy, Maki (Alfredo), Miguel Ángel, Moleman, Negative, Serpico / Yer, Vic\_Thor, Xurro.**

Lo peor en estos casos es que siempre te olvidas de alguien, así que a todos aquellos que lo merecen pero no han sido nombrados, sabed que en mi agradecimiento sí estáis incluidos.

Gracias a todos y a todas.

**Death Master**

## Distribución de este documento

Este documento se distribuye en formato PDF realizado bajo OpenOffice.org 1.1.0.

### Ficheros a distribuir:

Nombre: "conf\_zh04.pdf"

Descripción: Documento principal.

Nombre: "hash.txt"

Descripción: Contiene la cadena hash MD5 del fichero "conf\_zh04.pdf"

Nombre: "conf\_zh04.pdf.sig"

Descripción: Firma digital PGP del fichero "conf\_zh04.pdf" realizada por el autor.

### Datos adicionales:

El hash MD5 puede resultar útil para comprobar la integridad del fichero descargado, pero no es garantía de la inalterabilidad del documento, pues puede haber sido alterado junto a la cadena de hash.

Para comprobar la completa autenticidad e inalterabilidad del fichero, usar el sistema PGP para validar el fichero .sig de firma. Cualquier modificación no autorizada del documento hará que la firma del mismo no sea válida, y ésta es imposible de falsificar.

**Death Master**

### Autenticación:

-----BEGIN PGP MESSAGE-----

```
qANQR1DBwUwDJoT5ygJgu7ABEACcxL+/kYEUZgF/WtBRma9PirZRBExCH4X7r2gz
rJMF6JKfIvP7zBR/GUPvAvz0ZnWuL0Rzpd4EOuPKVNLsD8JHXmcMS9IBmcRdJ1my
KY1UPVWbh+7rhPRmtee5z0k/z4mvGO5DWGPYbeqhxxoLfqWZDcd/I39x4x3808g
o8pfQ7QxWH5ahvdg4i8PA8yEg9iqfm2vDa+0ChiOwxH89O8VHISW6AZ0bb+y8OIG
4SafRtXYqOBZb7QHciPw9aYnot7UMFyv4cYsUSkPqy/xm7X1T0TAeqhJTnZvvLLW
2crGRzIQdWz/ctlaY1Gmi4Wv3MpemQceWr4IYC+cF+MAAat1HBOG4HUWjT8IEE7O
j+sPoSNTmEH2GT0QAeZzKFIPdPTHGRGKzVhYbuJqzTd/3td15eWkfWp9sQKWzYBY
Ah5e0q/LNcln14iJfpBc+WWMKvQDrcMDaFPKxxaoYwpzp7piGqeenPkTW5EZDHnJ
pvxvzYOUuornypgNct2fOenwi0SG0mlgWVn4XT4jZfVa+UfAtseMu7aMJDiDeY
bktSSNdRwnKYnFjt4dJoLe2grFdxm6OCHTUSKsbVX8yAtPNvembz2E0Fq0ORPd
zBS8wjkoOQuWV8hP3d6g7y4SBv/NmBBesPhSlvXQCax4curturc/RdLrSgGoUrJ
H9lpatLCywHd3o1JekmZ6OIZRhecwA5ThUwLJavXYLaTnbuut2X9I2RfzFetJBik
sXl6Kt2uoVDLd61ytMXruijQ6sSLiCttlvsQ+Avs7wshuz5gPJ13U7bqjXptx8yQ
MbkyYlcO9kRwdFIPBrjAepuK+MPJM43R4Os9zJxozZzp9mAF8Y3i8slmb7eDdWDB
eOsCJNxoAVZS/8cdNhbCaRgQnEwU9njLpe852uNHu2mVfonl8sKfIHJ3mmipGsk
fCl4Ytsvv05itQaVUAgVgk71mO3fpkZCGihMIKJ6l42871TOWCM8lp+SAZ3EpC3z
m7193w5tLPz2gjt5/DGx9nzovQzzbDQQNJ9RZ5L2cf+nX0E5At7nZIZDajksLKg0
W1i69Oqa3ZwhfqXjUfingJsfWE+whBh/AhVEnx0zrbzYDteN8CFyavpVvhBJaDw
ukJGxfAZd+UK/OKy989y38iy9RWrr/bdlqxJKgGQ8+qglDpj2Eii7OhBamdXCxAnP
VzCTSFL3xT5wrVvN8L42gwr0oLra43YJw1lssbRbbwyFci610L9uDVGDgEbBdFTQ
4c5slX/OZP38hzbhb/aTr5nkZ1R6DXPtQLPRzJeT7KgYDSzEJVpA9zksPDSXclbf
fy7LpU9spa+L7/OFuzKA8jv5NYbFxl2u6CBaCVczhErrNWdLzfRm3Px+flcoxZ
1NEc1T5WFVwydM1dLYWlj4w8EEC1B4P0F3U4DOeoAQRvI7BI9AKv3C9YV+thBgfY
6jHQZY9h3XWfuBhc4Pf7n2DZVC4IL5Tmh/4phtthjSAGc6UJSwEwPJf8q4kCGZTL
47WK9j1mag1HT8DosH5DTQKcULMVQg2OZ6as75LJcwpjTF5AS24qycrb4on80F07
JWUis4WPb/5NHhvEgs8vR7LcUfpx1eBzxMMQC8y+D1sz7J/0P62XcjC4RGPGiRHf
NMNm7KPEZVdCTaQpX98dCXtx3QjXwJhlv9SX/in5if0MiJowwkYeZ0JelbLI9X
H/mmpbRBzoNXe/LnbPep8UeIWGqzok7ZxVmrgnoR06V1LFRRkkvOapqOn9u651a
bl7savZ0E7S6F+rBJU3kHXanwRzGGguaiE1NncTotQyFNPh66ZXCmKWMQV2b0rEJ
Y6aWjHVVYhhd/ESocasNfjlxludk/wV85Uvef4IUvn5/YrjxxLsOy226zdyy4Dwe
O20=
```

=y/fy

-----END PGP MESSAGE-----

## Licencia

### Conferencias de Seguridad Informática – Zona-Hispana Party 2004 – Versión 1.1

Este documento contiene los temas impartidos, así como información adicional relacionada con las **Conferencias de Seguridad Informática** impartidas los días 2, 3 y 4 de Enero de 2004 en la **Zona-Hispana Party 2004** (Sabadell) por **Death Master**.

Este documento ha sido liberado por su autor bajo la licencia GNU General Public License (GPL), y su utilización, copia o reproducción queda sujeta a los términos de la citada licencia, que puede ser consultada en el siguiente sitio web:

- **GNU General Public License:** <http://www.gnu.org/copyleft/gpl.html>  
GPL Version 2, June 1991  
Copyright (C) 1989, 1991 Free Software Foundation, Inc.

Cualquier copia, modificación, distribución o utilización en general de este documento debe respetar la autoría original del mismo, correspondiente a **Death Master**.

---

### Computer Security Conferences – Zona-Hispana Party 2004 – Version 1.1

This document contains the imparted topics, as well as additional information related with the imparted **Computer Security Conferences** the days 2, 3 and 4 of January of 2004 in the **Zona-Hispana Party 2004** (Sabadell) by **Death Master**.

This document has been freed by its author under the license GNU General Public License (GPL), and its use, copy or reproduction is subject to the terms of the mentioned license that can be consulted in the following website:

- **GNU General Public License:** <http://www.gnu.org/copyleft/gpl.html>  
GPL Version 2, June 1991  
Copyright (C) 1989, 1991 Free Software Foundation, Inc.

Any copy, modification, distribution or general purpose use of this document should respect the original responsibility of it, corresponding to **Death Master**.



\* End Of File \*