Sniffers

&

Snipho, a particular approach.

kiuman@gmail.com
(a.k.a Predicad0r)

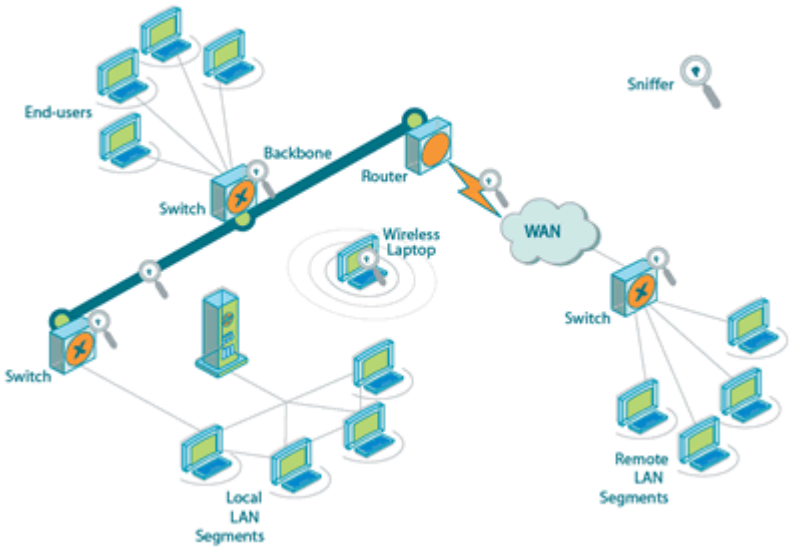
Indice

Introducción

El objetivo fundamental de esta documentación es adentrar al lector en los analizadores de paquetes o -por su denominación en inglés- *packet sniffers*, su modo de trabajo, sus incumbencias en seguridad y posibles formas de detectarlos y defendernos ante ellos.

Antes de adentrarnos en el tema, vamos a hacer una introducción a las redes de datos y especialmente a la pila TCP/IP que es la que más nos incumbe.

Finalmente veremos una implementación particular de un sniffer de imágenes, esto es, un sniffer al que solo le preocupa en capturar la transmisión de imágenes y se encarga de almacenarlas para un futuro análisis.



Familia de Protocolos de red

La Familia de protocolos de Internet es un conjunto de protocolos de red en la que se basa Internet y que permiten la transmisión de datos entre redes de computadoras. En ocasiones se la denomina conjunto de protocolos TCP/IP, en referencia a los dos protocolos más importantes que la componen: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP), que fueron los dos primeros en definirse, y que son los más utilizados de la familia. Existen tantos protocolos en este conjunto que llegan a ser más de 100 diferentes, entre ellos se encuentra el popular HTTP (HyperText Transfer Protocol), que es el que se utiliza para acceder a las páginas web, además de otros como el ARP (Address Resolution Protocol) para la resolución de direcciones, el FTP (File Transfer Protocol) para transferencia de archivos, y el SMTP (Simple Mail Transfer Protocol) y el POP (Post Office Protocol) para correo electrónico, TELNET para acceder a equipos remotos, entre otros.

El TCP/IP es la base de Internet, y sirve para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo PC, minicomputadoras y computadoras centrales sobre redes de área local (LAN) y área extensa (WAN). TCP/IP fue desarrollado y demostrado por primera vez en 1972 por el departamento de defensa de los Estados Unidos, ejecutándolo en ARPANET, una red de área extensa del departamento de defensa.

La familia de protocolos de Internet puede describirse por analogía con el modelo OSI, que describe los niveles o capas de la pila de protocolos, aunque en la práctica no corresponde exactamente con el modelo en Internet. En una pila de protocolos, cada nivel soluciona una serie de problemas relacionados con la transmisión de datos, y proporciona un servicio bien definido a los niveles más altos. Los niveles superiores son los más cercanos al usuario y tratan con datos más abstractos, dejando a los niveles más bajos la labor de traducir los datos de forma que sean físicamente manipulables.

El modelo de Internet fue diseñado como la solución a un problema práctico de ingeniería. El modelo OSI, en cambio, fue propuesto como una aproximación teórica y también como una primera fase en la evolución de las redes de ordenadores. Por lo tanto, el modelo OSI es más fácil de entender, pero el modelo TCP/IP es el que realmente se usa. Sirve de ayuda entender el modelo OSI antes de conocer TCP/IP, ya que se aplican los mismos principios, pero son más fáciles de entender en el modelo OSI.

Niveles en la pila TCP/IP

Hay algunas discusiones sobre como encaja el modelo TCP/IP dentro del modelo OSI. Como TCP/IP y modelo OSI no están delimitados con precisión no hay una respuesta que sea la correcta.

El modelo TCP/IP no está lo suficientemente dotado en los niveles inferiores como para detallar la auténtica estratificación en niveles: necesitaría tener una capa extra (el nivel de Red) entre los niveles de transporte e Internet. Protocolos específicos de un tipo concreto de red, que se sitúan por encima del marco de hardware básico, pertenecen al nivel de red, pero sin serlo. Ejemplos de estos protocolos son el ARP (Protocolo de resolución de direcciones) y el STP (Spanning Tree Protocol). De todas formas, estos son protocolos locales, y trabajan por debajo de las capas de Internet. Cierto es que situar ambos grupos (sin mencionar los protocolos que forman parte del nivel de Internet pero se sitúan por encima de los protocolos de Internet, como ICMP) todos en la misma capa puede producir confusión, pero el modelo OSI no llega a ese nivel de complejidad para ser más útil como modelo de referencia.

El siguiente diagrama intenta mostrar la pila OSI y otros protocolos relacionados con el modelo OSI original:

7	Aplicación	ej. HTTP, DNS, SMTP, SNMP, FTP, Telnet, SSH y SCP, NFS, RTSP, Feed, Webcal , POP3
6	Presentación	ej. XDR, ASN.1, SMB, AFP
5	Sesión	ej. TLS, SSH, ISO 8327 / CCITT X.225, RPC, NetBIOS, TELNET
4	Transporte	ej. TCP, UDP, RTP, SCTP, SPX
3	Red	ej. IP, ICMP, IGMP, X.25, CLNP, ARP, RARP, BGP, OSPF, RIP, IGRP, EIGRP, IPX, DDP
2	Enlace de datos	ej. Ethernet, Token Ring, PPP, HDLC, Frame Relay, RDSI, ATM, IEEE 802.11, FDDI
1	Físico	ej. cable, radio, fibra óptica

Normalmente, los tres niveles superiores del modelo OSI (Aplicación, Presentación y Sesión) son considerados simplemente como el nivel de aplicación en el conjunto TCP/IP. Como TCP/IP no tiene un nivel de sesión unificado sobre el que los niveles superiores se sostengan, estas funciones son típicamente desempeñadas (o ignoradas) por las aplicaciones de usuario. La diferencia más notable entre los modelos de TCP/IP y OSI es el nivel de

Aplicación, en TCP/IP se integran algunos niveles del modelo OSI en su nivel de Aplicación. Una interpretación simplificada de la pila TCP/IP se muestra debajo:

5	Aplicación	ej. HTTP, FTP, DNS (protocolos de enrutamiento como BGP y RIP, que por varias razones funcionen sobre TCP y UDP respectivamente, son considerados parte del nivel de red)
4	Transporte	ej. TCP, UDP, RTP, SCTP (protocolos de enrutamiento como OSPF, que funcionen sobre IP, son considerados parte del nivel de red)
3	Internet	Para TCP/IP este es el Protocolo de Internet (IP) (protocolos requeridos como ICMP e IGMP funcionan sobre IP, pero todavía se pueden considerar parte del nivel de red; ARP no funciona sobre IP)
2	Enlace	ej. Ethernet, Token Ring, PPP, HDLC, Frame Relay, RDSI, ATM, IEEE 802.11, FDDI
1	Físico	ej. medio físico, y técnicas de codificación, T1, E1

El nivel Físico

El nivel físico describe las características físicas de la comunicación, como las convenciones sobre la naturaleza del medio usado para la comunicación (como las comunicaciones por cable, fibra óptica o radio), y todo lo relativo a los detalles como los conectores, código de canales y modulación, potencias de señal, longitudes de onda, sincronización y temporización y distancias máximas.

El nivel de Enlace de datos

El nivel de enlace de datos especifica cómo son transportados los paquetes sobre el nivel físico, incluyendo los delimitadores (patrones de bits concretos que marcan el comienzo y el fin de cada trama). Ethernet, por ejemplo, incluye campos en la cabecera de la trama que especifican que máquina o máquinas de la red son las destinatarias de la trama. Ejemplos de protocolos de nivel de enlace de datos son Ethernet, Wireless Ethernet, SLIP, Token Ring y ATM.

PPP es un poco más complejo y originalmente fue diseñado como un protocolo separado que funcionaba sobre otro nivel de enlace, HDLC/SDLC.

Este nivel es a veces subdividido en Control de enlace lógico (Logical Link Control) y Control de acceso al medio (Media Access Control).

El nivel de Internet

Como fue definido originalmente, el nivel de red soluciona el problema de conseguir transportar paquetes a través de una red sencilla. Ejemplos de protocolos son X.25 y Host/IMP Protocol de ARPANET.

Con la llegada del concepto de Internet, nuevas funcionalidades fueron añadidas a este nivel, basadas en el intercambio de datos entre una red origen y una red destino. Generalmente esto incluye un enrutamiento de paquetes a través de una red de redes, conocida como Internet.

En la familia de protocolos de Internet, IP realiza las tareas básicas para conseguir transportar datos desde un origen a un destino. IP puede pasar los datos a una serie de protocolos superiores; cada uno de esos protocolos es identificado con un único "Número de protocolo IP". ICMP y IGMP son los protocolos 1 y 2, respectivamente.

Algunos de los protocolos por encima de IP como ICMP (usado para transmitir información de diagnóstico sobre transmisiones IP) e IGMP (usado para dirigir tráfico multicast) van en niveles superiores a IP pero realizan funciones del nivel de red e ilustran una incompatibilidad entre los modelos de Internet y OSI. Todos los protocolos de enrutamiento, como BGP, OSPF, y RIP son realmente también parte del nivel de red, aunque ellos parecen pertenecer a niveles más altos en la pila.

El nivel de Transporte

Los protocolos del nivel de transporte pueden solucionar problemas como la fiabilidad ("¿alcanzan los datos su destino?") y la seguridad de que los datos llegan en el orden correcto. En el conjunto de protocolos TCP/IP, los protocolos de transporte también determinan a qué aplicación van destinados los datos.

Los protocolos de enrutamiento dinámico que técnicamente encajan en el conjunto de protocolos TCP/IP (ya que funcionan sobre IP) son generalmente

considerados parte del nivel de red; un ejemplo es OSPF (protocolo IP número 89).

TCP (protocolo IP número 6) es un mecanismo de transporte fiable y orientado a conexión, que proporciona un flujo fiable de bytes, que asegura que los datos llegan completos, sin daños y en orden. TCP realiza continuamente medidas sobre el estado de la red para evitar sobrecargarla con demasiado tráfico. Además, TCP trata de enviar todos los datos correctamente en la secuencia especificada. Esta es una de las principales diferencias con UDP, y puede convertirse en una desventaja en flujos en tiempo real (muy sensibles a la variación del retardo) o aplicaciones de enrutamiento con porcentajes altos de pérdida en el nivel de Internet.

Más reciente es SCTP, también un mecanismo fiable y orientado a conexión. Está relacionado con la orientación a byte, y proporciona múltiples sub-flujos multiplexados sobre la misma conexión. También proporciona soporte de multihoming, donde una conexión puede ser representada por múltiples direcciones IP (representando múltiples interfaces físicas), así si hay una falla la conexión no se interrumpe. Fue desarrollado inicialmente para aplicaciones telefónicas (para transportar SS7 sobre IP), pero también fue usado para otras aplicaciones.

UDP (protocolo IP número 17) es un protocolo de datagramas sin conexión. Es un protocolo no fiable (best effort al igual que IP) - no porque sea particularmente malo, sino porque no verifica que los paquetes lleguen a su destino, y no da garantías de que lleguen en orden. Si una aplicación requiere estas características, debe llevarlas a cabo por sí misma o usar TCP.

UDP es usado normalmente para aplicaciones de streaming (audio, video, etc) donde la llegada a tiempo de los paquetes es más importante que la fiabilidad, o para aplicaciones simples de tipo petición/respuesta como el servicio DNS, donde la sobrecarga de las cabeceras que aportan la fiabilidad es desproporcionada para el tamaño de los paquetes.

DCCP está actualmente bajo desarrollo por el IETF. Proporciona semántica de control para flujos TCP, mientras de cara al usuario se da un servicio de datagramas UDP.

TCP y UDP son usados para dar servicio a una serie de aplicaciones de alto nivel. Las aplicaciones con una dirección de red dada son distinguibles entre sí

por su número de puerto TCP o UDP. Por convención, los puertos bien conocidos (well-known ports) son asociados con aplicaciones específicas.

RTP es un protocolo de datagramas que ha sido diseñado para datos en tiempo real como el streaming de audio y video que se monta sobre UDP.

El nivel de Aplicación

El nivel de aplicación es el nivel que los programas más comunes utilizan para comunicarse a través de una red con otros programas. Los procesos que acontecen en este nivel son aplicaciones específicas que pasan los datos al nivel de aplicación en el formato que internamente use el programa y es codificado de acuerdo con un protocolo estándar.

Algunos programas específicos se considera que se ejecutan en este nivel. Proporcionan servicios que directamente trabajan con las aplicaciones de usuario. Estos programas y sus correspondientes protocolos incluyen a HTTP (HyperText Transfer Protocol), FTP (Transferencia de archivos), SMTP (correo electrónico), SSH (login remoto seguro), DNS (Resolución de nombres de dominio) y a muchos otros.

Una vez que los datos de la aplicación han sido codificados en un protocolo estándar del nivel de aplicación son pasados hacia abajo al siguiente nivel de la pila de protocolos TCP/IP.

En el nivel de transporte, las aplicaciones normalmente hacen uso de TCP y UDP, y son habitualmente asociados a un número de puerto bien conocido (well-known port). Los puertos fueron asignados originalmente por la IANA.

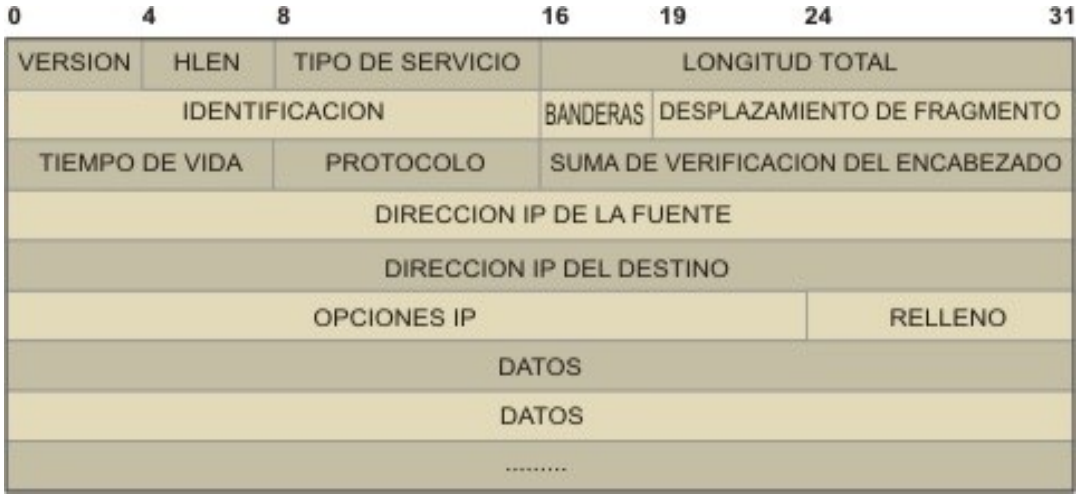
Cómo funciona TCP/IP

IP

IP a diferencia del protocolo X.25, que está orientado a conexión, es sin conexión. Está basado en la idea de los datagramas, los cuales son transportados transparentemente, pero no siempre con seguridad, desde el host fuente hasta el host destinatario, quizás recorriendo varias redes mientras viaja.

El protocolo IP trabaja de la siguiente manera; la capa de transporte toma los mensajes y los divide en datagramas, de hasta 64 Kbytes cada uno. Cada datagrama se transmite a través de la red, posiblemente fragmentándose en unidades más pequeñas, durante su recorrido normal. Al final, cuando todas las piezas llegan a la máquina destinataria, la capa de transporte los reensambla para así reconstruir el mensaje original.

Un datagrama IP consta de una parte de cabecera y una parte de texto. La cabecera tiene una parte fija de 20 octetos y una parte opcional de longitud variable. En la siguiente figura se muestra el formato de la cabecera.



El campo **Versión** indica a qué versión del protocolo pertenece cada uno de los datagramas. Mediante la inclusión de la versión en cada datagrama, no se excluye la posibilidad de modificar los protocolos mientras la red se encuentre en operación.

Debido a que la longitud de la cabecera no es constante, un campo de la cabecera, **HLEN**, permite que se indique la longitud que tiene la cabecera en palabras de 32 bits. El valor mínimo es de 5. Tamaño 4 bits.

El campo **Tipo de servicio** le permite al host indicarle a la subred el tipo de servicio que desea. Es posible tener varias combinaciones con respecto a la seguridad y la velocidad. Para voz digitalizada, por ejemplo, es más importante la entrega rápida que corregir errores de transmisión. En tanto que, para la transferencia de archivos, resulta más importante tener la transmisión fiable que entrega rápida. También, es posible tener algunas otras combinaciones, desde un tráfico rutinario, hasta una anulación instantánea. Tamaño 8 bits.

La **Longitud total** incluye todo lo que se encuentra en el datagrama -tanto la cabecera como los datos. La máxima longitud es de 65536 bytes. Tamaño 16 bits.

El campo **Identificación** se necesita para permitir que el host destinatario determine a qué datagrama pertenece el fragmento recién llegado. Todos los fragmentos de un datagrama contienen el mismo valor de identificación. Tamaño 16 bits.

Enseguida vienen 3 bits de **banderas** donde el primero que no se utiliza, y después dos campos de 1 bit. El primero de estos, DF, quiere decir no fragmentar. Esta es una orden para que las pasarelas no fragmenten el datagrama, porque el extremo destinatario es incapaz de poner las partes juntas nuevamente. Por ejemplo, supóngase que se tiene un datagrama que se carga en un micro pequeño para su ejecución; podría marcarse con DF porque la ROM de micro espera el programa completo en un datagrama. Si el datagrama no puede pasarse a través de una red, se deberá encaminar sobre otra red, o bien, desecharse. El segundo bit, MF, significa más fragmentos. Todos los fragmentos, con excepción del último, deberán tener ese bit puesto. Se utiliza como una verificación doble contra el campo de Longitud total, con objeto de tener seguridad de que no faltan fragmentos y que el datagrama entero se reensamble por completo.

El **desplazamiento de fragmento** indica el lugar del datagrama actual al cual pertenece este fragmento. En un datagrama, todos los fragmentos, con excepción del último, deberán ser un múltiplo de 8 bytes, que es la unidad elemental de fragmentación. Dado que se proporcionan 13 bits, hay un máximo de 8192 fragmentos por datagrama, dando así una longitud máxima de datagrama de 65536 bytes, que coinciden con el campo **Longitud total**. Tamaño 16 bits.

El campo **Tiempo de vida** es un contador que se utiliza para limitar el tiempo de vida de los paquetes. Cuando se llega a cero, el paquete se destruye. La unidad de tiempo es el hop y equivale a un paso por un router, permitiéndose un tiempo de vida máximo de 255 hops. Tamaño 8 bits.

Cuando la capa de red ha terminado de ensamblar un datagrama completo, necesitará saber qué hacer con él. El campo Protocolo indica, a qué proceso de transporte pertenece el datagrama. El TCP es efectivamente una posibilidad, pero en realidad hay muchas más.

Protocolo: El número utilizado en este campo sirve para indicar a qué protocolo pertenece el datagrama que se encuentra a continuación de la cabecera IP, de manera que pueda ser tratado correctamente cuando llegue a su destino. Tamaño 8 bits.

La **suma de verificación del encabezado** es necesaria para verificar que los datos contenidos en la cabecera IP son correctos. Por razones de eficiencia este campo no puede utilizarse para comprobar los datos incluidos a continuación, sino que estos datos de usuario se comprobarán posteriormente a partir del código de redundancia de la cabecera siguiente, y que corresponde al nivel de transporte. Este campo debe calcularse de nuevo cuando cambia alguna opción de la cabecera, como puede ser el tiempo de vida. Tamaño 16 bits.

La **Dirección IP de la fuente** contiene la dirección del host que envía el paquete. Tamaño 32 bits.

La **Dirección IP del destino** es la dirección del host que recibirá la información. Los routers o gateways intermedios deben conocerla para dirigir correctamente el paquete. Tamaño 32 bits.

El campo **Opciones IP** se utiliza para fines de seguridad, encaminamiento fuente, informe de errores, depuración, sellado de tiempo, así como otro tipo de información. Esto, básicamente, proporciona un escape para permitir que las versiones subsiguientes de los protocolos incluyan información que actualmente no está presente en el diseño original. También, para permitir que los experimentadores trabajen con nuevas ideas y para evitar, la asignación de bits de cabecera a información que muy rara vez se necesita.

Direcciones IPv4

En su versión 4, una dirección IP se representa mediante un número binario de 32 bits (IPv4). Las direcciones IP se pueden expresar como números de notación decimal: se dividen los 32 bits de la dirección en cuatro octetos. El valor decimal de cada octeto puede ser entre 0 y 255 (el número binario de 8 bits más alto es 11111111 y esos bits, de derecha a izquierda, tienen valores decimales de 1, 2, 4, 8, 16, 32, 64 y 128, lo que suma 255 en total).

En la expresión de direcciones IPv4 en decimal se separa cada octeto por un carácter ".". Cada uno de estos octetos puede estar comprendido entre 0 y 255, salvo algunas excepciones. Los ceros iniciales, si los hubiera, se pueden obviar.

Ejemplo de representación de dirección IPv4: 85.167.179.93

Hay tres clases de direcciones IP que una organización puede recibir de parte de la Internet Corporation for Assigned Names and Numbers (ICANN): clase A, clase B y clase C. En la actualidad, ICANN reserva las direcciones de clase A para los gobiernos de todo el mundo (aunque en el pasado se le hayan otorgado a empresas de gran envergadura como, por ejemplo, Hewlett Packard) y las direcciones de clase B para las medianas empresas. Se otorgan direcciones de clase C para todos los demás solicitantes. Cada clase de red permite una cantidad fija de equipos (hosts).

- *En una red de clase A*, se asigna el primer octeto para identificar la red, reservando los tres últimos octetos (24 bits) para que sean asignados a los hosts, de modo que la cantidad máxima de hosts es 2^{24} - 2 (las direcciones reservadas de broadcast [últimos octetos a 255] y de red [últimos octetos a 0]), es decir, 16777214 hosts.
- *En una red de clase B*, se asignan los dos primeros octetos para identificar la red, reservando los dos octetos finales (16 bits) para que sean asignados a los hosts, de modo que la cantidad máxima de hosts es 2^{16} - 2, o 65534 hosts.
- *En una red de clase C*, se asignan los tres primeros octetos para identificar la red, reservando el octeto final (8 bits) para que sea asignado a los hosts, de modo que la cantidad máxima de hosts es 2^{8} - 2, o 254 hosts.

Clase	Rango	Nro Redes	Nro Hosts	Máscara	Broacast
A	1.0.0.0 - 127.255.255.255	126	16777214	255.0.0.0	x.255.255.255
B	128.0.0.0 - 191.255.255.255	16384	65534	255.255.0.0	x.x.255.255
C	192.0.0.0 - 223.255.255.255	2097152	254	255.255.255.0	x.x.x.255
D	224.0.0.0 - 239.255.255.255	Dirección de multicast			
E	240.0.0.0 - 255.255.255.255	Reservado para uso futuro			

- La dirección 0.0.0.0 es utilizada por las máquinas cuando están arrancando o no se les ha asignado dirección.
- La dirección que tiene su parte de host a cero sirve para definir la red en la que se ubica. Se denomina dirección de red.
- La dirección que tiene su parte de host a unos sirve para comunicar con todos los hosts de la red en la que se ubica. Se denomina dirección de broadcast.
- Las direcciones 127.x.x.x se reservan para pruebas de retroalimentación. Se denomina dirección de bucle local o loopback.

Hay ciertas direcciones en cada clase de dirección IP que no están asignadas y que se denominan direcciones privadas. Las direcciones privadas pueden ser utilizadas por los hosts que usan traducción de dirección de red (NAT) para conectarse a una red pública o por los hosts que no se conectan a Internet. En una misma red no puede existir dos direcciones iguales, pero sí se pueden repetir en dos redes privadas que no tengan conexión entre sí o que se vean a través de NAT. Las direcciones privadas son:

- Clase A: 10.0.0.0 a 10.255.255.255 (8 bits red, 24 bits hosts)
- Clase B: 172.16.0.0 a 172.31.255.255 (16 bits red, 16 bits hosts)
- Clase C: 192.168.0.0 a 192.168.255.255 (24 bits red, 8 bits hosts)

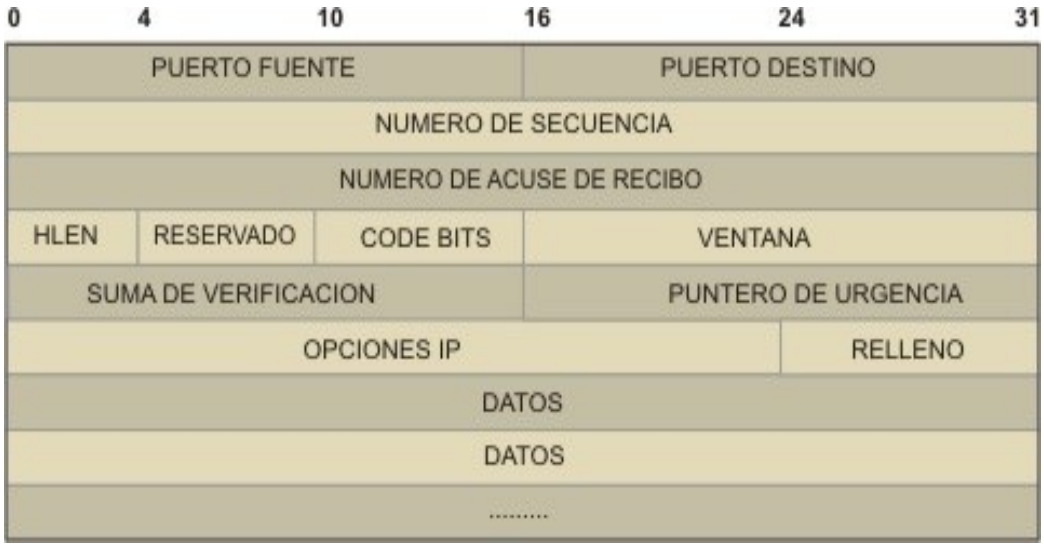
Las direcciones privadas se pueden utilizar junto con un servidor de traducción de direcciones de red (NAT) para suministrar conectividad a todos los hosts de una red que tiene relativamente pocas direcciones públicas disponibles. Según lo acordado, cualquier tráfico que posea una dirección destino dentro de uno de los intervalos de direcciones privadas no se enrutará a través de Internet.

TCP

Una entidad de transporte TCP acepta mensajes de longitud arbitrariamente grande procedentes de los procesos de usuario, los separa en pedazos que no excedan de 64 Kbytes y, transmite cada pedazo como si fuera un datagrama separado. La capa de red, no garantiza que los datagramas se entreguen apropiadamente, por lo que TCP deberá utilizar temporizadores y retransmitir los datagramas si es necesario. Los datagramas que consiguen llegar, pueden hacerlo en desorden; y dependerá de TCP el hecho de reensamblarlos en mensajes, con la secuencia correcta.

Cada octeto de datos transmitido por TCP tiene su propio número de secuencia privado. El espacio de números de secuencia tiene una extensión de 32 bits, para asegurar que los duplicados antiguos hayan desaparecidos, desde hace tiempo, en el momento en que los números de secuencia den la vuelta. TCP, sin embargo, sí se ocupa en forma explícita del problema de los duplicados retardados cuando intenta establecer una conexión, utilizando el protocolo de ida-vuelta-ida para este propósito.

En la siguiente figura se muestra la cabecera que se utiliza en TCP.



Los campos **Puerto fuente** y **Puerto destino** identifican los puntos terminales de la conexión. Cada host deberá decidir por sí mismo cómo asignar sus puertos.

El campo **Número de secuencia** tiene un doble rol, si la bandera SYN esta activada entonces es el número de secuencia inicial y **Acuse de recibo** en superposición efectúan sus funciones usuales. Estos tienen una longitud de 32 bits, debido a que cada octeto de datos está numerado en TCP.

La Longitud de la cabecera TCP, **HLEN**, indica el número de palabra de 32 bits que están contenidas en la cabecera de TCP. Esta información es necesaria porque el campo **Opciones** tiene una longitud variable, y por lo tanto la cabecera también.

Después aparecen seis **code bits** o banderas de 1 bit. Si el **Puntero de urgencia** se está utilizando, entonces **URG** se coloca a 1. El **puntero de urgencia** se emplea para indicar un desplazamiento en octetos a partir del número de secuencia actual en el que se encuentran datos urgentes. Esta facilidad se brinda en lugar de los mensajes de interrupción. El bit **SYN** se utiliza para el establecimiento de conexiones pero más adelante veremos el establecimiento de una conexión. El bit **RST** se utiliza para reiniciar una conexión que se ha vuelto confusa debido a SYN duplicados y retardados, o a caída de los hosts. En bit **ACK** se utiliza como acuse de recibo. El bit **FIN** se utiliza para liberar la conexión; especifica que el emisor ya no tiene más datos por enviar. Después de cerrar una conexión, un proceso puede seguir recibiendo datos indefinidamente. El bit **PSH** indica empujar un paquete para priorizarlo en el envío.

El control de flujo en TCP se trata mediante el uso de una **ventana** deslizante de tamaño variable. La ventana indica el número de octetos que se pueden transmitir más allá del octeto asentido por el campo ventana..

La **suma de verificación** también se brinda como un factor de seguridad

extrema. El algoritmo de código de redundancia consiste en sumar simplemente todos los datos, considerados como palabras de 16 bits, y después tomar el complemento a 1 de la suma.

El campo de **Opciones** se utiliza para diferentes cosas, por ejemplo para comunicar el máximo tamaño de segmento o timestamps entre otras cosas.

El campo **datos** contiene la carga útil a enviar.

Las aplicaciones más comunes que hacen uso de este tipo de protocolo son Navegadores web, aplicaciones de transmisión de archivos, envío de emails, etc.

Funcionamiento de TCP

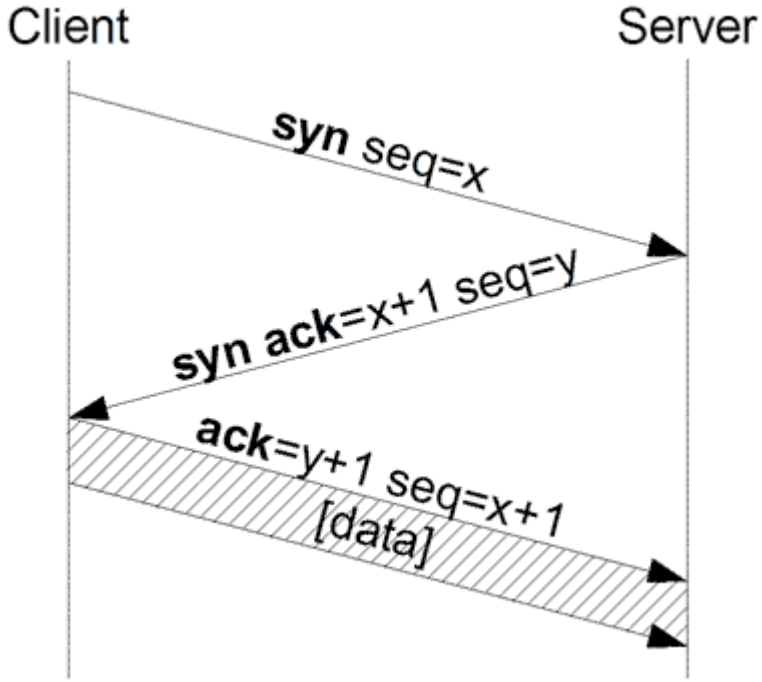
Las conexiones TCP se componen de tres etapas: establecimiento de conexión, transferencia de datos y fin de la conexión. Para establecer la conexión se usa el procedimiento llamado negociación en tres pasos (Three-way handshake). Una negociación en cuatro pasos (Four-way handshake) es usada para la desconexión. Durante el establecimiento de la conexión, algunos parámetros como el número de secuencia son configurados para asegurar la entrega ordenada de los datos y la robustez de la comunicación.

Establecimiento de la conexión (Three-way handshake)

Aunque es posible que un par de entidades finales comiencen una conexión entre ellas simultáneamente, normalmente una de ellas abre un socket en un determinado puerto tcp y se queda a la escucha de nuevas conexiones. Es común referirse a esto como apertura pasiva, y determina el lado servidor de una conexión. El lado cliente de una conexión realiza una apertura activa de un puerto enviando un paquete SYN inicial al servidor como parte de la negociación en tres pasos; también envía un número de secuencia inicial (x en este ejemplo). En el lado del servidor se comprueba si el puerto está abierto, es decir, si existe algún proceso escuchando en ese puerto. En caso de no estarlo, se envía al cliente un paquete de respuesta con el bit RST activado, lo que significa el rechazo del intento de conexión. En caso de que sí se encuentre abierto el puerto, el lado servidor respondería a la petición SYN válida con un paquete SYN/ACK y ademas su propio número de secuencia inicial (y en este ejemplo) y reconociendo el número de secuencia del cliente + 1 (x+1). Finalmente, luego que el cliente reciba el SYN/ACK enviado por el servidor, éste debería responderle con un ACK y reconociendo el número de secuencia del servidor + 1 (y+1) y su propio número de secuencia + 1 (x+1), completando así la negociación en tres pasos (SYN, SYN/ACK y ACK) y la fase

de establecimiento de conexión.

Es interesante notar que existe un número de secuencia generado por cada lado donde cada sistema operativo implementa un algoritmo diferente para la generación de éste número, intentando ayudar de este modo a que no se puedan establecer conexiones falseadas. Aún así algunos sistemas operativos tienen generadores de secuencia de escasa efectividad[1] donde el número de secuencia es fácilmente predecible permitiendo de este modo a que un atacante inserte información en una comunicación no encriptada o producir ataques de denegación de servicio enviando paquetes de reset.



Transferencia de datos

Durante la etapa de transferencia de datos, una serie de mecanismos claves determinan la fiabilidad y robustez del protocolo. Entre ellos están incluidos el uso del número de secuencia para ordenar los segmentos TCP recibidos y detectar paquetes duplicados, checksums para detectar errores, y asentimientos y temporizadores para detectar pérdidas y retrasos.

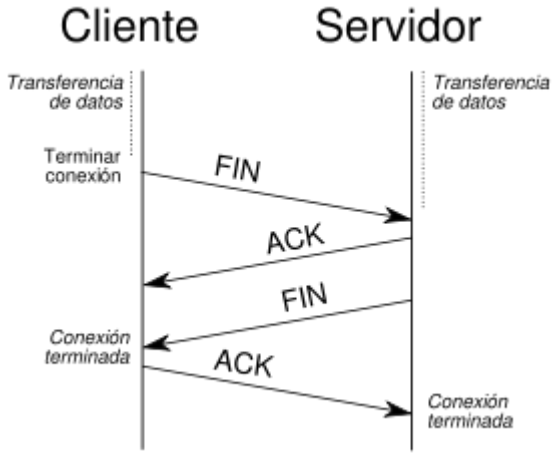
Finalización de la conexión

La fase de finalización de la conexión usa una negociación en cuatro pasos (Four-way handshake), terminando la conexión desde cada lado independientemente. Cuando uno de los dos extremos de la conexión desea

1 Ver: http://lcamtuf.coredump.cx/newtcp/

parar su "mitad" de conexión transmite un paquete FIN, que el otro interlocutor asentirá con un ACK. Por tanto, una desconexión típica requiere un par de segmentos FIN y ACK desde cada lado de la conexión.

Una conexión puede estar "medio abierta" en el caso de que uno de los lados la finalice pero el otro no. El lado que ha dado por finalizada la conexión no puede enviar más datos pero la otra parte si podrá.



UDP

UDP (User Datagram Protocol), es un protocolo de datagrama sin corrección; no provee las garantía de fiabilidad y ordenamiento de TCP a los protocolos del Nivel de Aplicación y los datagramas pueden llegar en desorden o perderse sin notificación. Como consecuencia de lo anterior es que UDP es un protocolo más rápido y eficiente para tareas ligeras o sensibles al tiempo proveyendo una interfaz muy simple entre el Nivel de Red y Nivel de Aplicación. Si se requiere algún tipo de fiabilidad para los datos transmitidos, esta debe ser implementada en los niveles superiores de la pila.

Al igual que IP, y a diferencia de TCP, es un protocolo de mejor esfuerzo o no-fiable. El único problema de fiabilidad que resuelve es la corrección de errores en la cabecera y datos transmitidos a través de un campo de 16 bits para suma de verificación (checksum), una forma de control de redundancia con la finalidad de proteger la integridad de datos verificando que no hayan sido corrompidos.

La estructura de paquetes UDP es muy simple y consiste de 4 campos.

Puerto de origen. Encargado de identificar el puerto que envía y que se asume será el puerto hacia donde se envía la respuesta si se necesita. Este campo es opcional: si no se utiliza, el valor del campo debe ser 0.

Puerto de destino. Identifica el puerto de destino. Es obligatorio.

Longitud. Un campo de 16 bits que especifica la longitud del datagrama completo: cabecera y datos. La longitud mínima es de 8 bytes ya que es la longitud misma de la cabecera.

Suma de verificación. Un campo de 16 bits que se utiliza para verificar errores en cabecera y datos.

Las aplicaciones más comunes que hacen uso de este tipo de protocolo son DNS, aplicaciones de transmisión de medios, voz sobre IP (VoIP), TFTP y juegos en línea.

Sniffers

Que es un Sniffer?

Un sniffer es un programa de computadora o un dispositivo de hardware que puede interceptar el tráfico transmitido en una red de datos (o en parte de ella). En cuanto la información viaja por la red, el sniffer captura cada paquete de datos y lo analiza o guarda para futura inspección.

Usos más comunes

Los sniffer son bastante versátiles y pueden ser usados para:

- Analizar problemas de red.
- Detectar intentos de intrusión de red.
- Obtener información para invadir una red.
- Monitorear uso de una red.
- Obtener estadísticas de red.
- Filtrar tráfico sospechoso en una red.
- Espiar a otros usuarios de la red y obtener información sensitiva tales como passwords.
- Realizar ingeniería inversa de protocolos de red.
- Depurar comunicaciones cliente/servidor.

Ejemplos de uso

- Un sniffer en una red token ring podría detectar la pérdida del token o la presencia de varios tokens.
- Un sniffer podría detectar que varios mensajes están siendo enviados a un adaptador de red; si el adaptador no recibe los mensajes entonces entonces esto nos serviría para detectar el punto de la falla (o el adaptador de red o un tramo de la red).
- El sniffer podría detectar una cantidad excesiva de mensajes siendo enviados a un port, detectando así errores de implemetación.
- Un sniffer podría calcular estadísticas de cantidad de tráfico sobre una red para poder determinar ampliaciones de ancho de banda o

reestructuraciones de la red para lograr una mejor performance.

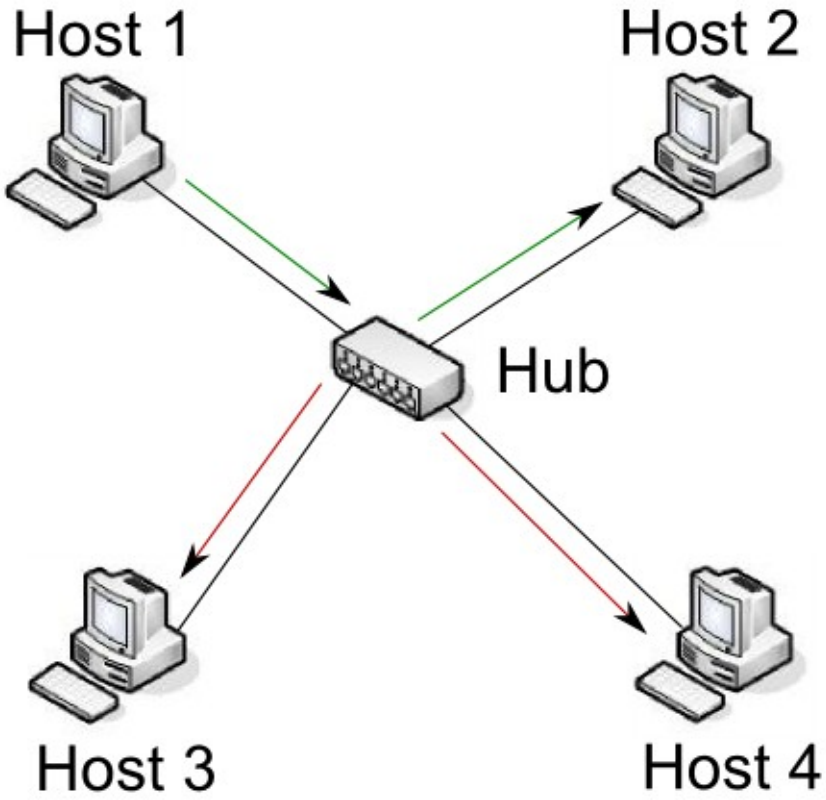
- Un sniffer podría ser usado para auditar los contenidos siendo transferidos por una red. Éste es el caso de Snipho que veremos más adelante.

Como funciona un Sniffer

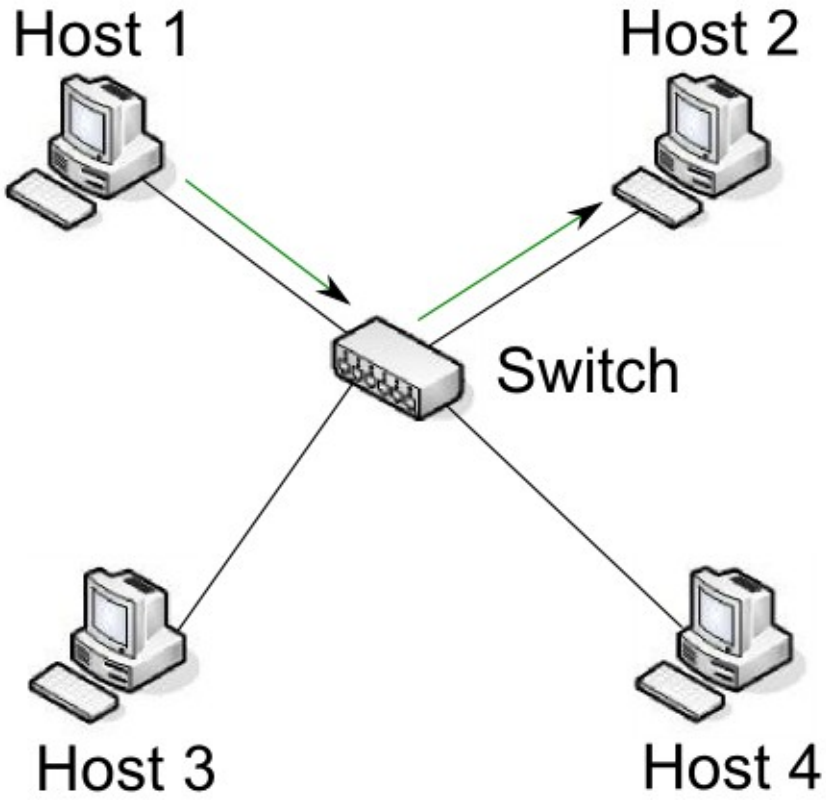
Una computadora conectada a una red de área local (LAN) tiene dos direcciones. Una es la dirección MAC (Media Access Control) que identifica unívocamente cada nodo en una red y esta dirección esta almacenada en el adaptador de red mismo. Es la dirección MAC la que es usada por el protocolo Ethernet al construir los frames que son transferidos desde y hacia otras máquinas. La otra dirección es la IP, la cual es usada por las aplicaciones. La capa de datos usa un encabezado Ethernet con dirección MAC de origen y destino en vez de la dirección IP. La capa de red es la encargada de mapear una dirección IP en una dirección MAC tal cual lo requiere el protocolo de enlace de datos. Inicialmente busca la dirección MAC del host destino en una tabla, normalmente llamada ARP (Address Resolution Protocol) cache. Si no se encuentra ninguna entrada para esa dirección IP, el ARP envía un pedido a todas las máquinas de la LAN, el host que posee esa dirección IP responde al host origen con su dirección MAC. Esta dirección MAC luego es agregada a la tabla de ARP cache del host origen para un uso futuro. Ahora, el host origen usa esa dirección MAC cada vez que se quiere comunicar con el otro host.

Hay dos tipos básicos de entornos Ethernet y el modo en que los sniffers trabajan en estos entornos es un tanto diferente.

- **Shared Ethernet**: en un entorno de Ethernet compartido todos los hosts están conectados al mismo bus de datos y compiten uno con otro por el acceso al medio. En tal entorno los paquetes enviados a un host son recibidos por todos los hosts en el segmento. De este modo cuando el host 1 quiere hablar con el host 2, esta envía un paquete a la red con la dirección MAC destino del host 2. Todas las máquinas en el segmento Ethernet reciben el paquete y comparan la dirección MAC destino del paquete con su propia dirección MAC; si ambas son distintas entonces el paquete es descartado, si son iguales el paquete es aceptado. Un host corriendo un sniffer rompe esta regla y acepta todos los paquetes. Tal host se dice que esta en modo promiscuo y puede escuchar todo el tráfico en la red. Sniffing en una red Ethernet compartida es totalmente pasivo y por ende difícil de detectar pero no imposible, más adelante veremos algunas técnicas.



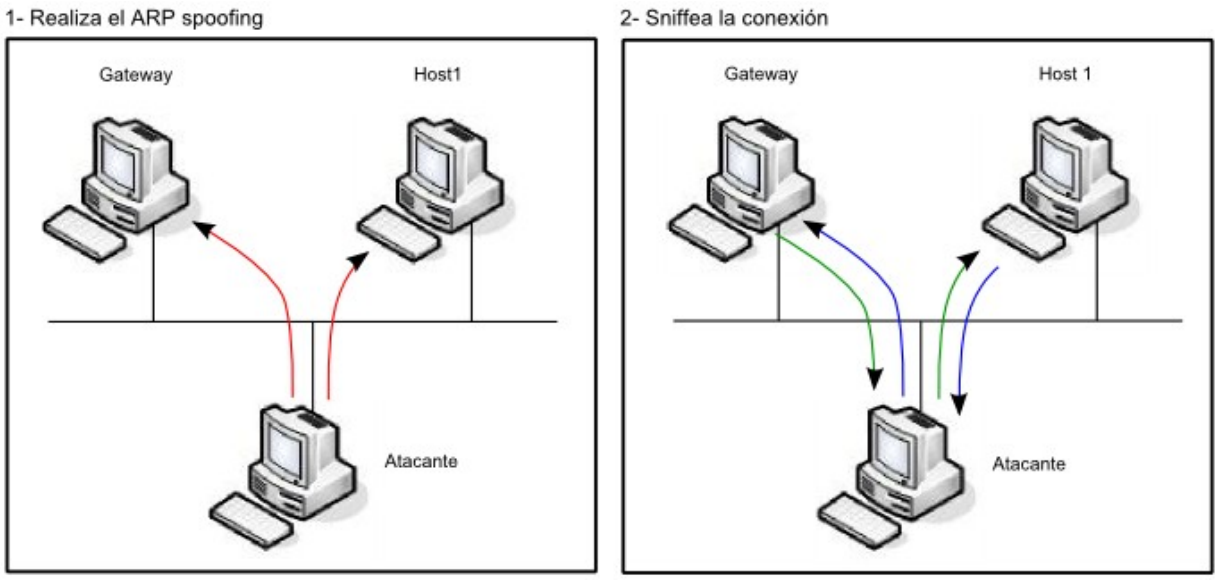
- **Switched Ethernet**: un entorno Ethernet donde los hosts están conectados a un switch en vez de a un hub se llama Switched Ethernet. Un switch mantiene una tabla llevando cuenta de cada dirección MAC de cada host y el puerto físico en el switch al que cada host está conectado y entrega los paquetes destinados a un host particular. El switch es un dispositivo "inteligente" que envía los paquetes al host destino únicamente y no envía el paquete a todas los hosts conectados a la red como en el caso del hub. Esto resulta en una mejor utilización del medio y aprovechamiento del ancho de banda disponible mejorando también la seguridad. Por ende siguiendo el proceso anterior de poner un host en modo promiscuo para obtener los paquetes no funciona. Como resultado de esto, muchos administradores de sistemas caen en el error de creer que las redes basadas en switches son totalmente seguras e inmunes al sniffing. Desgraciadamente esto no es verdad como veremos a continuación.



Aun cuando un switch es más seguro que un hub, los siguientes métodos pueden ser usados para hacer sniffing en un red basada en switch:

1. **ARP spoofing**: ya explicamos anteriormente como ARP es usado para obtener la dirección MAC de un host destino con el cual nos queremos comunicar. ARP no tiene estado, por ende se puede enviar una respuesta ARP aun cuando nadie envió una consulta y tal respuesta será aceptada. El principio del ARP Spoofing es enviar mensajes ARP falsos (falsificados, o spoofed) a la Ethernet. Normalmente la finalidad es asociar la dirección MAC del atacante con la dirección IP de otro host (el host atacado), como por ejemplo la puerta de enlace predeterminada (gateway). Cualquier tráfico dirigido a la dirección IP de ese host, será erróneamente enviado al atacante, en lugar de a su destino real. El atacante, puede entonces elegir, entre reenviar el tráfico a la puerta de enlace predeterminada real (ataque pasivo o sniffing), o modificar los datos antes de reenviarlos (ataque activo). El atacante puede incluso lanzar un ataque de tipo DoS (Denial of Service - Denegación de Servicio) contra una víctima, asociando una dirección MAC inexistente con la dirección IP de la puerta de enlace predeterminada de la víctima. Cabe destacar que con este procedimiento solo se ve el tráfico con origen en el host y destino el gateway, pero no el inverso, para ver el inverso también se debe envenenar el ARP cache del gateway. El único método de prevenir

completamente el ARP Spoofing, es el uso de tablas ARP estáticas. Una buena herramienta para este tipo de ataques en el arpspoof[2].



2. **MAC flooding**: los switches mantienen una tabla que mapea una (o varias) direcciones MAC con los puertos físicos en el switch. Gracias a esto puede enrutar paquetes de una manera más inteligente de un host a otro. Los switchs tienen una memoria limitada para almacenar estas tablas. El MAC Flooding hace uso de esta limitación para bombardear el switch con varias direcciones MAC falsas hasta que el switch no puede almacenar ninguna más entonces el switch entra en un estado conocido como "failopen mode" donde comienza a comportarse como un hub enviando los paquetes a todos los hosts de la red. Una vez que esto sucede, el sniffing puede ser realizado sin mayor esfuerzo. Algunos switches avanzados proveen la oportunidad de activar ciertas protecciones contra este tipo de ataques, como por ejemplo, limitar un subconjunto de direcciones MAC dedicadas a un puerto o desactivar un puerto en caso que este reciba demasiadas direcciones MAC. Una Buena herramienta para este tipo de ataques es macof[3].

2 Parte de Dsniff - http://www.monkey.org/~dugsong/dsniff/ - o http://arpspoof.sourceforge.net/
3 Parte de Dsniff - http://www.monkey.org/~dugsong/dsniff/

Detectando sniffers

Un sniffer es normalmente pasivo, solo colecta información, por lo tanto se torna extremadamente difícil detectar los sniffers cuando funcionan en un red basada en hubs; sin embargo es un poco más sencillo cuando el sniffer esta corriendo en un red basada en switch. A continuación veremos algunos de los métodos de detección.

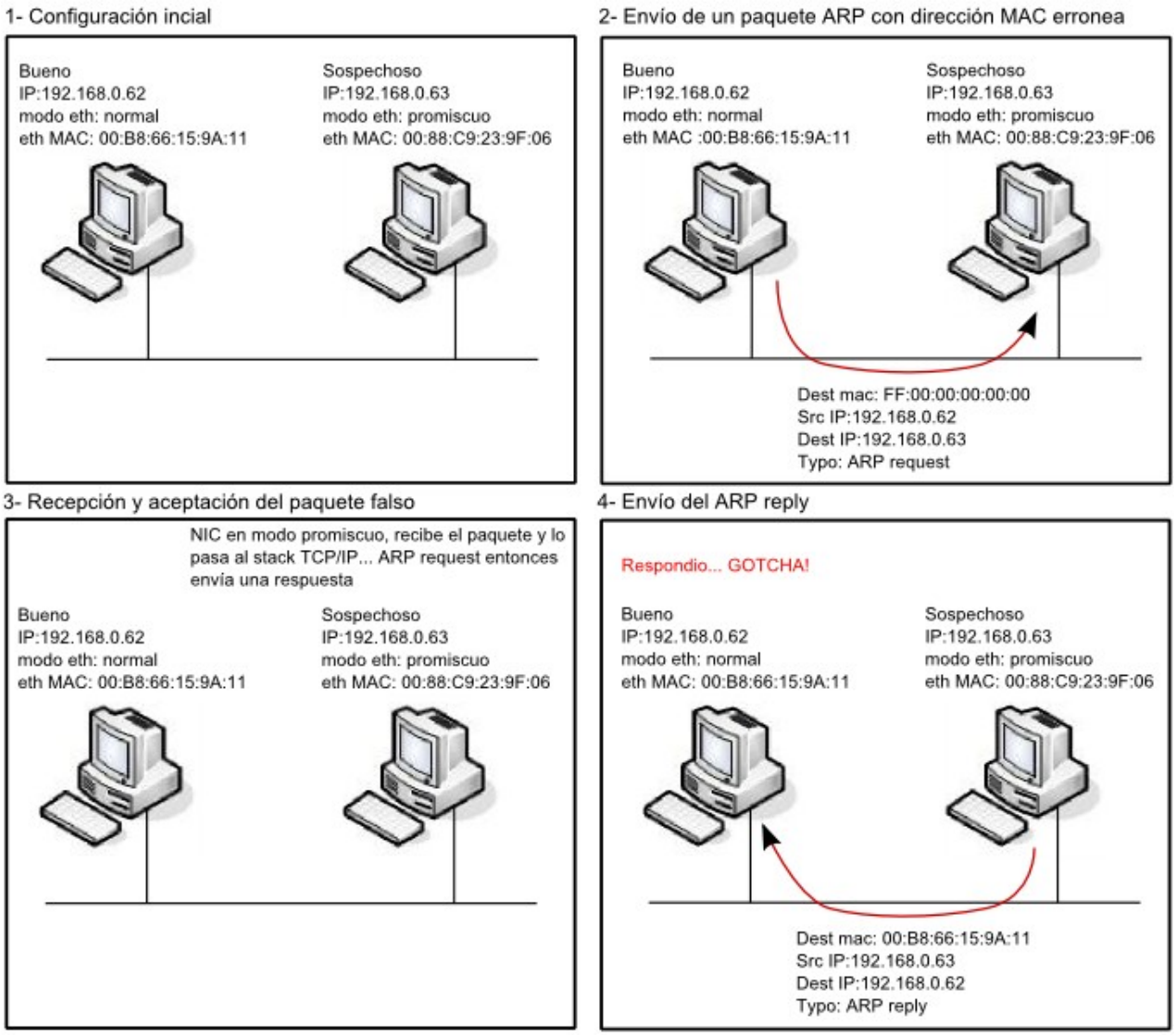
Métodos basados en MAC

El método de detección basado en MAC funciona basándose en fallas en la implementación de la pila TCP/IP por parte de algunos sistemas operativos. En algunas pilas TCP/IP y bajo circunstancias especificas la dirección MAC destino del encabezado ethernet no es chequeado o es chequeado insuficientemente cuando un adaptador de red esta en modo promiscuo. Debido a esto es posible generar un paquete Ethernet con una dirección MAC errónea que es pasado al código de procesamiento TCP/IP. Normalmente ese paquete sera descartado por el adaptador de red y nunca alcanzara a ser procesado por el sistema operativo. Sin embargo, cuando el adaptador de red esta en modo promiscuo es posible que esos paquetes sean procesados como si tuviesen una dirección MAC correcta. El truco de esta técnica es forzar una respuesta desde la pila TCP/IP y de este modo determinar si un paquete erróneamente direccionado es reconocido por algún host.

Normalmente hay dos método basados en esta técnica, el método de detección ARP y el método de detección Etherping.

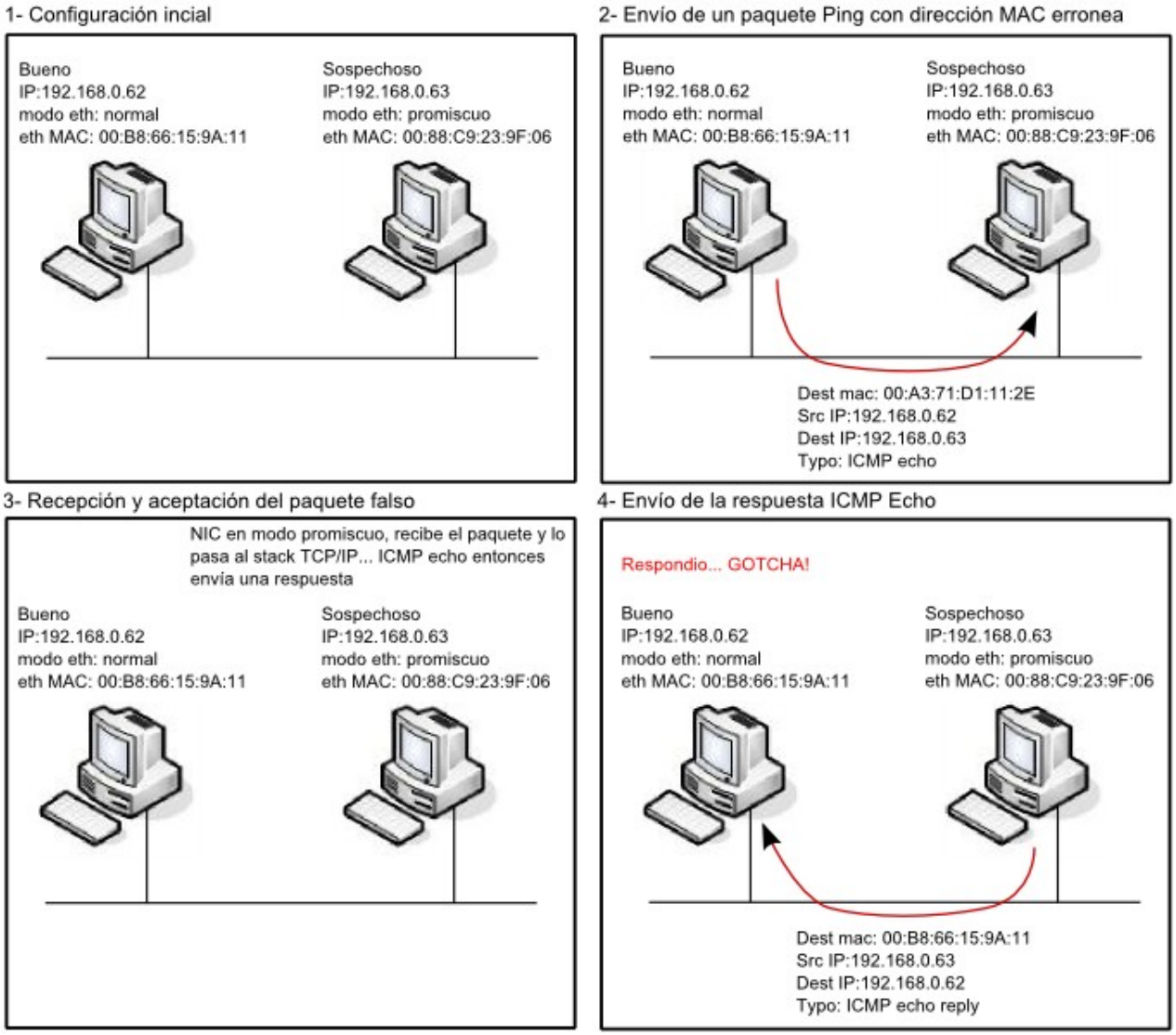
Método de detección ARP

Este método se basa en la falla que algunos sistemas operativos analizan los paquetes ARP. Este método usa los paquetes ARP request que son enviados a la IP del host destino con una dirección MAC. Normalmente estos paquetes serian descartados pero, estando en modo promiscuo, algunos sistemas operativos verán dichos paquetes como reales y responderán como es debido. Si el host destino responde a tal ARP request con un paquete ARP reply entonces sabemos que esta en modo promiscuo, por ende fácilmente podría haber un sniffer corriendo en ese host.



Método de detección Etherping

Este método se basa en la falla de como algunos sistemas operativos analizan los paquetes ICMP. El método usa paquetes ICMP echo que son enviados a la dirección IP correcta pero con una dirección MAC falsa. Normalmente tal paquete seria descartado, pero cuando el adaptador de red de un host se encuentra en modo promiscuo, algunos antiguos kernel tomaban esos paquetes como verdaderos ya que a dirección MAC nunca era chequeada y su IP si era correcta y respondía como debe ser dejándonos ver que tal host esta en modo promiscuo y por ende fácilmente podría haber un sniffer corriendo en ese host.



Métodos basados en señuelos

Las técnicas de detección basadas en señuelos trabajan en las bases del engaño o los henoypots[4]. La idea de detrás de estas técnicas es enviar tráfico carnada que sea atractivo para el sniffer como falsos passwords, falsos nombre usuarios, conexiones TCP falsas, etc, y esperar a que el dueño del sniffer lance un ataque reutilizando de este modo la información falsa. Debido al hecho que nadie a excepción del posible dueño del sniffer sabe sobre esta falsa información, el atacante puede ser fácilmente identificado.

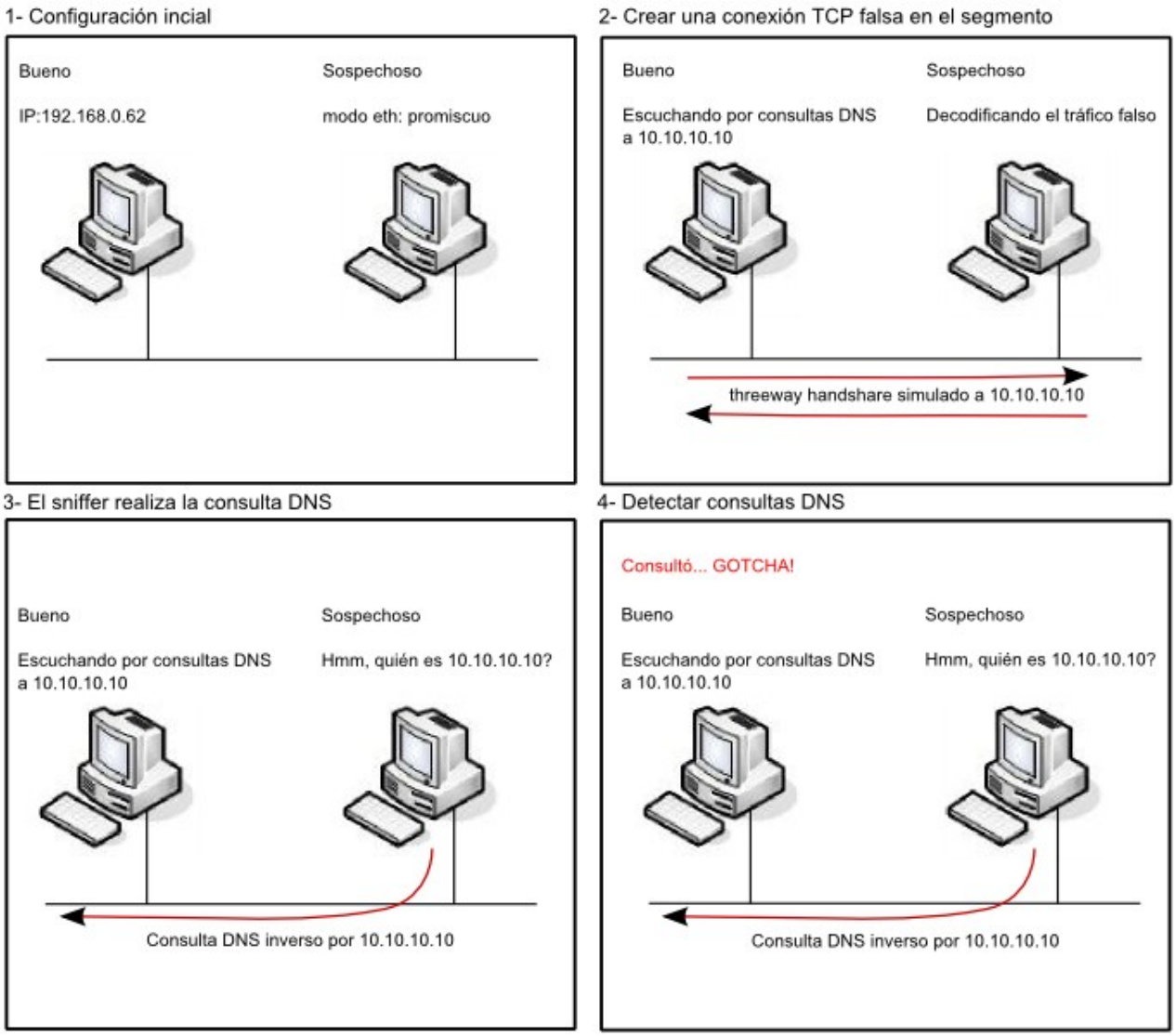
Generalmente, el método de detección basado en DNS es más usado hoy en día.

4 Ver: http://en.wikipedia.org/wiki/Honeypot_(computing)

Método de detección basado DNS

Este método se aprovecha del comportamiento presente en la mayoría de los sniffers. El hecho que la mayoría de los sniffers no es completamente pasivo y que tiene a generar algún tráfico, aun cuando es difícil distinguir cuando el tráfico generado proviene de un sniffer o no. Por defecto, la mayoría de los sniffers hacen consultas DNS inversas sobre el tráfico que esta capturando. Ya que este tráfico es generado por el sniffer, el truco detrás se basa en de algún modo detectar este tráfico y distinguirlo de las consultas DNS normales.

Al generar tráfico falso en el segmento Ethernet a alguna dirección de IP en desuso es posible detectar la presencia del sniffer ya que el tráfico generado normalmente es ignorado por los hosts en el segmento, si alguna petición DNS por esa IP es generada entonces hay un sniffer. Un modo fácil de implementar esto es generar un Threeway handshake falso y luego sniffear el tráfico tratando de encontrar una consulta DNS inversa por la IP usada en la conexión falsa.



Métodos basados en latencia de redes y máquinas

La técnica de detección basada en la latencia de máquinas y la red funciona por medio de la detección de la degradación de la performance causada por la interrupción del adaptador de red al sistema operativo. Cuando los sniffers están ejecutándose, ponen al adaptador de red en modo promiscuo como ya fue mencionado. Cuando se está en modo promiscuo, todo el tráfico ethernet que pase por el adaptador de red generará una interrupción de hardware que causará que el driver se ejecute. Es más, todos los paquetes capturados deben ser pasados al programa sniffer. Es sabido que todo este proceso de cambios de contexto son caros (en recursos y tiempo de procesador) para el sistema, por ende, en ocasiones de alta cantidad de tráfico un sniffer puede degradar significativamente la performance de un sistema.

El truco de este tipo de técnicas es de algún modo medir de forma remota y con precisión la carga de un sistema cuando el segmento de red se encuentra con altas cantidades de tráfico. Esto puede ser realizado midiendo el tiempo de respuesta del host que es sospechoso de estar ejecutando un sniffer. Una de las mayores dificultades en la implementación de esta técnica es el como la medida es tomada.

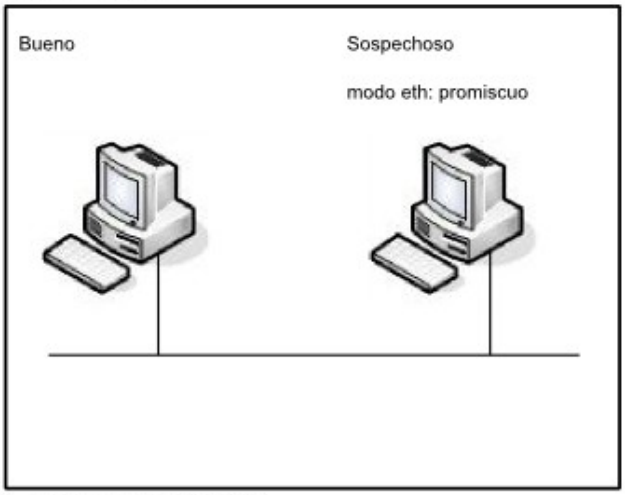
Método de detección de carga

En el método de detección de carga, dos medidas de tiempo de respuesta son tomadas. Una medida es tomada para determinar el tiempo de respuesta sin demasiado tráfico en la red, y otra medida de tiempo de respuesta es tomada durante un alto tráfico en la red.

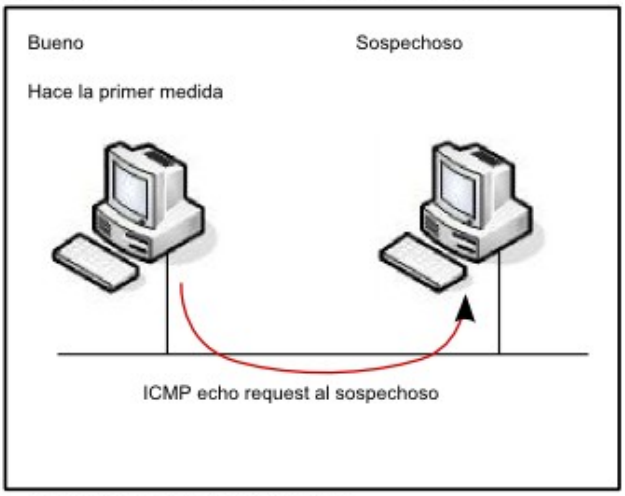
El método de detección de carga se basa en la asunción que el sniffer hace una interpretación de los datos. Primero se envía un paquete ICMP request al host sospechoso de ejecutar un sniffer y se mide el tiempo de respuesta. Luego, una gran cantidad de paquetes de ICMP request con una dirección de destino en desuso es enviada inundando la red; mientras tanto también se envía un paquete ICMP request al host sospechoso. El host sospechoso interpretara los paquetes enviados incrementando así la carga de ésta e incrementando así el tiempo de respuesta a los ICMP request. La diferencia entre el tiempos de respuesta del host sospechoso y los demás hosts indica que la máquina sospechosa esta en modo promiscuo. En tal caso, es probable que un sniffer este siendo ejecutado en el host sospechoso.

El mayor problema con éste tipo de técnicas es que puede degradar la performance general de la red, aun más, es susceptible a timeouts y falsos positivos debido que los paquetes pueden ser retrasados debido a la misma carga de la red.

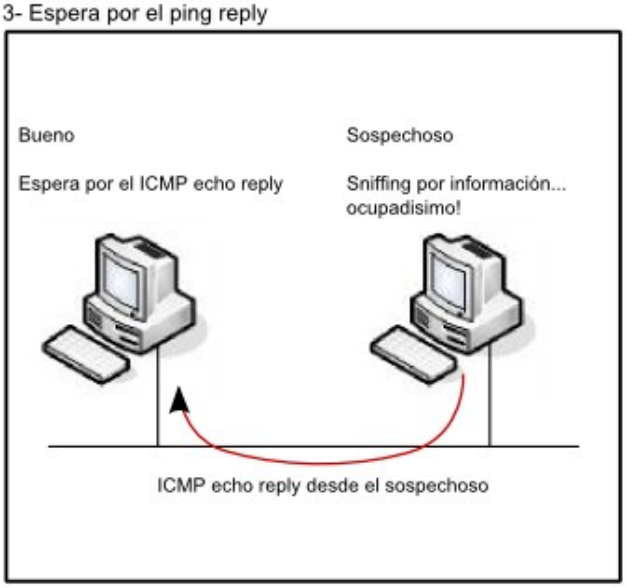
1- Configuración incial



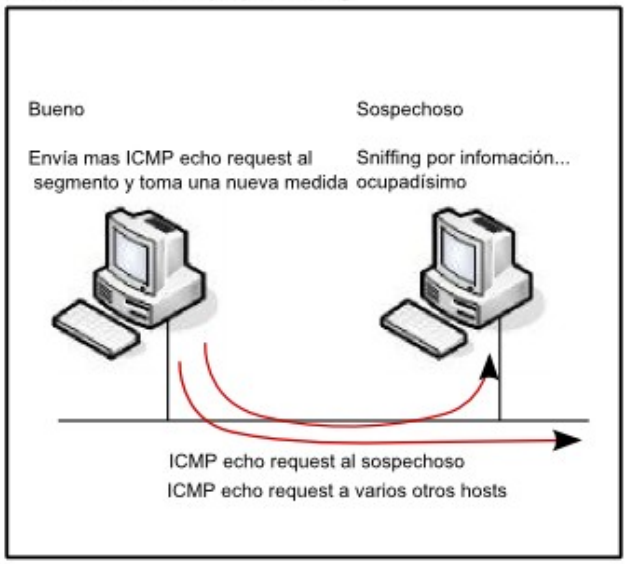
2- Envía un ping para medir el tiempo de respuesta



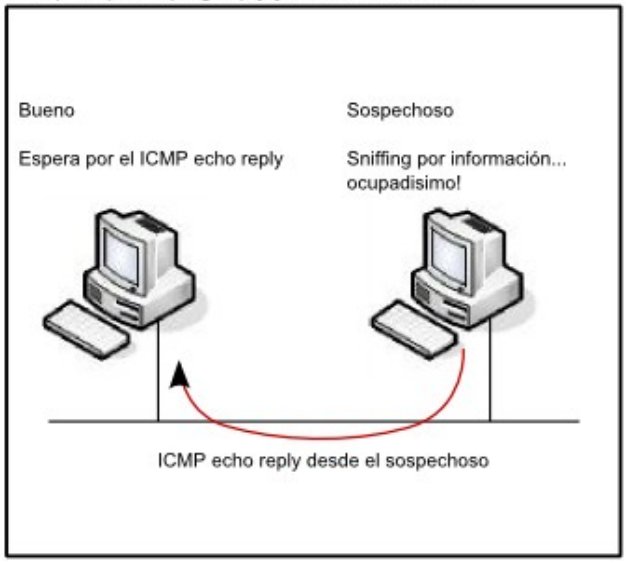
3- Espera por el ping reply



4- Inunda la red con paquetes ping



5- Espera por el ping reply y mide nuevamente



Intrusion Detection Systems (IDS)

Los sistemas de detección de intrusos (IDS por sus siglas en inglés) son un importante mecanismo de seguridad. Normalmente son empleados junto a otros mecanismo de seguridad para implementar una política de seguridad. Los IDS no sólo ayudan a la detección por parte de otros mecanismos de seguridad (como firewalls o service wrappers) pero también proveen información de nuevos ataques que hayan podido traspasar otros mecanismos de seguridad. Los IDS también pueden ser usados en casos de análisis forenses gracias a la información que colectan, haciendo de este modo que los atacantes piensen dos veces antes de realizar una acción en contra de nuestro sistema.

Hay dos modelos principales de IDS:

1. Misuse detection model: la detección se basa en monitorear los puntos débiles de los sistemas que pueden ser descriptos por un patrón o una secuencia de eventos o información, llamado firma.
2. An anomaly detection model: la detección se basa en detectar cambios en los patrones de uso y comportamiento de los sistemas por parte de los usuarios.

Los IDS son o host-based o network-based. Los host-based toman sus decisiones basados en información obtenida de un solo host (normalmente en auditorias), mientras que los network-based obtienen información monitoreando el tráfico de información en la red a la cual los host están conectados. Los IDS son usados para detectar ataques tanto activos como pasivos; los sniffers están incluidos dentro de las herramientas para realizar ataques pasivos.

SNORT como un ejemplo de IDS

Snort es un sniffer de paquetes y un detector de intrusos basado en red (se monitorea todo un dominio de colisión). Es un software muy flexible que ofrece capacidades de almacenamiento de sus bitácoras tanto en archivos de texto como en bases de datos abiertas como lo es MySQL. Implementa un motor de detección de ataques y barrido de puertos que permite registrar, alertar y responder ante cualquier anomalía previamente definida. Así mismo existen herramientas de terceros para mostrar informes en tiempo real (ACID) o para convertirlo en un Sistema Detector y Preventor de Intrusos.

Este IDS implementa un lenguaje de creación de reglas flexible, potente y sencillo. Durante su instalación ya nos provee de cientos de filtros o reglas para backdoor, DDoS, finger, FTP, ataques web, CGI, Nmap...

Puede funcionar como sniffer (podemos ver en consola y en tiempo real qué ocurre en nuestra red, todo nuestro tráfico), registro de paquetes (permite guardar en un archivo los logs para su posterior análisis, un análisis offline) o como un IDS normal (en este caso NIDS). Cuando un paquete coincide con algún patrón establecido en las reglas de configuración, se logea. Así se sabe cuando, de donde y cómo se produjo el ataque.

Aún cuando tcpdump es considerada una herramienta de auditoria muy útil, no se considera un verdadero IDS puesto que no analiza ni señala paquetes por anomalías. tcpdump imprime toda la información de paquetes a la salida en pantalla o a un archivo de registro sin ningún tipo de análisis. Un verdadero IDS analiza los paquetes, marca las transmisiones que sean potencialmente maliciosas y las almacena en un registro formateado, así, Snort utiliza la librería estándar libcap y tcpdump como registro de paquetes en el fondo.

Snort está disponible bajo licencia GPL, gratuito y funciona bajo plataformas Windows y UNIX/Linux. Dispone de una gran cantidad de filtros o patrones ya predefinidos, así como actualizaciones constantes ante casos de ataques, barridos o vulnerabilidades que vayan siendo detectadas a través de los distintos boletines de seguridad.

La característica más apreciada de Snort, además de su funcionalidad, es su subsistema flexible de firmas de ataques. Snort tiene una base de datos de ataques que se está actualizando constantemente y a la cual se puede añadir o actualizar a través de Internet. Los usuarios pueden crear 'firmas' basadas en las características de los nuevos ataques de red y enviarlas a la lista de correo de firmas de Snort, para que así todos los usuarios de Snort se puedan beneficiar. Esta ética de comunidad y compartir ha convertido a Snort en uno de los IDSes basados en red más populares, actualizados y robustos.

Previniendo los sniffers

La mejor manera de prevenir, no al sniffer pero si que la información capturada le sea útil al sniffer, es usando enriptación, de este modo la

información capturada será inentendible para el atacante.

Si se esta en una red switcheada, la posibilidad de que se use el arpspoofing es muy elevada y el punto más probable de ataque sería el gateway. Para prevenir que esto suceda se podría agregar estáticamente la dirección MAC del gateway en los hosts. También se podría tener un host dedicado a detectar MAC floods o arp spoofings de modo que alerte al administrador en el momento que algo fuera de lo normal suceda en la red.

Pasarse a SSH. SHH es hoy en día prácticamente un modo standard de conectarse a un host GNU/Linux debido a que usa encriptación en contraposición a su alternativa no segura Telnet que envía todo el tráfico plano (sin encriptación) y es muy susceptible a la captura de nombres de usuarios y passwords.

En vez de usar HTTP usar HTTPS si la página visitada lo soporta. HTTPS es en resumen igual a HTTP solo que la comunicación entre el host y el servidor es encriptada. Este es un punto sumamente importante para los servicios de Home Banking.

Para la transferencia de archivos el FTP podría ser reemplazado por SFTP donde se implementa encriptación tanto para la autenticación como para la transferencia de los archivos. Para transferencias simples entre servidores usar el comando scp basado en SSH.

En el caso de e-mail se debería configurar (siempre y cuando el servidor lo soporte) para que haga autenticación en modo seguro para que el password no sea visible a ningún sniffer. El protocolo de envío de e-mails (SMTP) y el de recepción de e-mails (POP3) son planos, esto es, la información viaja sin ser encriptada, por ende un sniffer situado entre el punto de origen y destino podría interceptarlos y leerlos sin problema. Para evitar esta situación se recomienda el uso de PGP (Pretty Good Privacy) o GnuPG de modo que los e-mails sean encriptados en el punto de origen y desencriptados en el destino. Si se prefiere el uso de un sistema de webmail, Hushmail (https://www.hushmail.com/) es un servidor de webmail gratuito donde toda su comunicación se realiza por medio de HTTPS. Sistemas como Gmail y Hotmail sólo realizan la autenticación en modo encriptado y luego todo el tráfico es plano (no encriptado).

En caso de ser posible, usar IPsec. Los protocolos de IPsec actúan en la capa

de red, la capa 3 del modelo OSI. Otros protocolos de seguridad para Internet de uso extendido, como SSL, TLS y SSH operan de la capa de transporte (capas OSI 4 a 7) hacia arriba. Esto hace que IPsec sea más flexible, ya que puede ser utilizado para proteger protocolos de la capa 4, incluyendo TCP y UDP, los protocolos de capa de transporte más usados.

Algunos sniffers conocidos

tcpdump

tcpdump es un herramienta en línea de comandos cuya utilidad principal es analizar el tráfico que circula por la red.

Permite al usuario capturar y mostrar a tiempo real los paquetes transmitidos y recibidos en la red a la cual la computadora está conectada. Está escrito por Van Jacobson, Craig Leres, y Steven McCanne que trabajaban en ese momento en el Grupo de Investigación de Red del Laboratorio Lawrence Berkeley. Más tarde el programa fue ampliado por Andrew Tridgell.

tcpdump funciona en la mayoría de los sistemas operativos UNIX: Linux, Solaris, BSD, Mac OS X, HP-UX y AIX entre otros. En esos sistemas, tcpdump hace uso de la librería libpcap para capturar los paquetes que circulan por la red.

Existe una adaptación de tcpdump para los sistemas Windows que se llama WinDump y que hace uso de la librería Winpcap.

En UNIX y otros sistemas operativos, es necesario tener los privilegios del root para utilizar tcpdump.

El usuario puede aplicar varios filtros para que sea más depurada la salida. Un filtro es una expresión que va detrás de las opciones y que nos permite seleccionar los paquetes que estamos buscando. En ausencia de ésta, el tcpdump volcará todo el tráfico que vea el adaptador de red seleccionado.

Wireshark

Wireshark, antes conocido como Ethereal, es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones para desarrollo de software y protocolos, y como una herramienta didáctica para educación. Cuenta con todas las características estándar de un analizador de protocolos.

La funcionalidad que provee es similar a la de tcpdump, pero añade una interfaz gráfica y muchas opciones de organización y filtrado de información. Así, permite ver todo el tráfico que pasa a través de una red (usualmente una red Ethernet, aunque es compatible con algunas otras) estableciendo la configuración en modo promiscuo. También incluye una versión basada en texto llamada tshark.

Permite examinar datos de una red viva o de un archivo de captura guardado en disco. Se puede analizar la información capturada, a través de los detalles y sumarios por cada paquete. Wireshark incluye un completo lenguaje para filtrar lo que queremos ver y la habilidad de mostrar el flujo reconstruido de una sesión de TCP.

Wireshark es software libre, y se ejecuta sobre la mayoría de sistemas operativos Unix y compatibles, incluyendo Linux, Solaris, FreeBSD, NetBSD, OpenBSD, y Mac OS X, así como en Microsoft Windows.

Ettercap

Ettercap es un interceptor/sniffer/registrador para LANs con switch. Soporta disecciones activas y pasivas de varios protocolos (incluso aquellos cifrados, como SSH y HTTPS). También hace posible la inyección de datos en una conexión establecida y filtrado al vuelo aun manteniendo la conexión sincronizada gracias a su poder para establecer un ataque Man-in-the-middle (mediante arp-spoofing). Muchos modos de sniffing fueron implementados para darnos un conjunto de herramientas poderoso y completo de sniffing.

Kismet

Kismet es un detector de redes wireless 802.11 de capa 2 (Data Link), un sniffer y un IDS. Kismet funciona con cualquier adaptador de red inalámbrico que soporte el modo de monitorización raw, y puede rastrear tráfico 802.11b, 802.11a y 802.11g. El programa corre bajo Linux, FreeBSD, NetBSD,

OpenBSD, y Mac OS X.

Conclusión

Para ser honestos debemos aclarar que no es fácil detectar un sniffer. A menudo debemos depender de nuestra intuición para darnos cuenta de la existencia de un sniffer ejecutándose en nuestra red. Si la performance de la red decae súbitamente es posible que alguien haya causado que uno de los switchs entre en el estado de ”failopen” o algunos usuarios se quejan que sus passwords fueron cambiados o material alterado, podemos sospechar de la presencia de un sniffer en la red.

El viejo dicho ”*mas vale prevenir que curar*” es muy cierto en este caso. Cada sniffer necesita privilegios de root para ejecutarse (al menos en sistemas basados en GNU/Linux, no asi los basados en Windows), por ende una buena política sería no dar permisos de administrador a los usuarios comunes. Si se tiene acceso de root en cada una de las máquinas, periódicamente se podría chequear localmente si el adaptador de red se encuentra en modo promiscuo o no con el solo uso del comando ifconfig.; incluso esto podría ser automatizado con scripts que se ejecuten periódicamente y reporten en caso de encontrar algo fuera de lo normal.

A continuación, y para finalizar, veremos la implementación de Snipho, un sniffer un tanto particular.

Snipho :'/

Alcances de Snipho

Snipho es una herramienta de auditorías de contenido. La herramienta puede ser adaptada para reconocer una amplia variedad de cosas (archivos) que transitan por una red de datos basada en TCP/IP.

En esta implemetación solo "presta atención" de las imágenes GIF y JPEG y las muestra en un mural (ademas de guardarlas). De éste modo se puede auditar el tipo de imágenes transitando por una red en busca de material ilegal; también puede ser usado como método intimidatorio en recintos públicos (bibliotecas, oficinas, etc) donde una pantalla va mostrando las imágenes que circulan por la red; de éste modo las personas se sentirán más reacias a mirar material para adultos.

Una variación podría ser detectar archivos de música (mp3, ogg, ACC, etc), leer los TAGs internos y determinar si es tráfico legal o no, en uno u otro caso hará determinadas acciones (logueo de IP, reportes, estadísticas, etc).

Snipho no solo captura las imágenes planas transmitidas vía una red, sino que también las que se encuentran dentro de un encabezado HTTP (o sea, el modo en que las obtenemos por medio de un navegador web como podría ser Opera o FireFox), por ende empezaremos describiendo como es tratado éste tipo de pedidos.

Pedido de imágenes en HTTP

Básicamente hay dos clases de paquetes HTTP, de pedido y de respuesta. En los primeros se especifica que es lo que se quiere y en los segundos se envía la respuesta o la causa del fallo de no ser posible cumplir el pedido.

Formato del pedido:

<Metodo> [sp] <Camino_de_Acceso> [sp] <versión> [cr][lf]

<Nombre_de_campo_1> : <valor> [cr][lf]

...

<Nombre_de_campo_n> : <valor> [cr][lf]

[cr][lf]

[Cuerpo]

ie:

```
GET / HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) Opera
7.50  [en]
Host: www.cs.uns.edu.ar
Accept: text/html, application/xml;q=0.9, application/xhtml+xml,
image/png, image/jpeg, image/gif, image/x-xbitmap, */*;q=0.1
Accept-Language: en;q=1.0
Accept-Charset: windows-1252, utf-8, utf-16, iso-8859-1;q=0.6,
*;q=0.1
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0
Connection: Keep-Alive
```

Formato de la respuesta:

<Linea de estado> [cr][lf]

<Nombre_de_campo_1> : <valor> [cr][lf]

...

<Nombre_de_campo_n> : <valor> [cr][lf]

[cr][lf]

[Cuerpo]

ie:

```
HTTP/1.1 200 OK
Date: Sat, 11 Sep 2004 07:46:06 GMT
Server: Apache/1.3.26 (Unix) PHP/4.2.3 mod_perl/1.27 mod_ssl/2.8.10
OpenSSL/0.9.6g
Last-Modified: Tue, 09 Dec 2003 09:26:44 GMT
ETag: "1035dc-192-3fd59554"
Accept-Ranges: bytes
Content-Length: 402
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

<HTML>
...
</HTML>
```

Conexión básica:

1. El cliente inicia una conexión.
2. El servidor la acepta o la rechaza.

   - si la conexión es aceptada:

3. El cliente envía un pedido.
4. El servidor responde el pedido.
5. La conexión se cierra.

Teniendo en cuenta estos conceptos introductorios, pasamos a ver el funcionamiento de Snipho.

Algoritmo principal

(Snipho fue desarrollado usando Delphi 6.0)

Lo primero que realiza Snipho es determinar los adaptadores de red y esperar a que el usuario elija uno sobre el que Snipho trabajara; luego pone el adaptador seleccionado en modo promiscuo y activa la heurística principal por cada paquete de información recibido.

La heurística principal del algoritmo se basa en mirar cada paquete. Los que inician una conexión son agregados a una lista para analizarlos cuando haya más información disponible. Los que contienen información son analizados al instante en busca de imágenes, si contienen indicios de imágenes entonces su conexión permanece viva y los datos se agregan, de lo contrario la conexión se marca para morir.

Nota: por conexión se entiende el par
(ip_origen:puerto_origen, ip_destino:puerto_destino)

```
un paquete entrante es detectado (PE de ahora en más)
SI PE corresponde a un paquete TCP ENTONCES
     se cuenta para estadísticas
     SI PE es un inicio de conexión ENTONCES
            se crea la conexión
     SINO
            SI PE pertenece a una conexión existente ENTONCES
                  DCPE = datos contenidos en PE
                  SI DCPE no es vacío y es el comienzo de una imagen ENTONCES
                        se agrega DCPE a los antiguos datos de esa conexión
                  SINO
                        se marca esa conexión como inservible y se la destina a morir
                  FIN SI
            FIN SI
            SI PE es un paquete de FIN de conexión ENTONCES
                  SI la conexión de PE era válida ENTONCES
                        Guardar todos los datos de la conexión en el disco y
                        mostrar la imagen en el mural
                        Eliminar la conexión y hacer un FLUSH de viejas conexiones
                  FIN SI
            FIN SI
     FIN SI
SINO
     se cuenta para estadísticas de acuerdo a su tipo
FIN SI
```

Unidades de soporte

Connections

Esta unidad se encarga de dar soporte al manejo de las conexiones. Implementa el tipo Tconn:

```
TConn = record
          ip_srcaddr : LongWord;  // ip origen
          ip_destaddr : LongWord; // ip destino
          src_portno : Word;      // puerto origen
          dst_portno : Word;      // puerto destino
          fin : byte;             // avisa que esta conexión no sirve más
          analyzed: boolean;      // flag de conexión ya analizada
          lastacc : TDateTime;    // ultimo tiempo de acceso
          ext : string;           // tipo de imagen que contiene
          data : TmemoryStream;   // la imagen relacionada a la conexión
        end;
```

con los siguientes métodos:

`function addConnection(sa : LongWord; sp : Word; da : LongWord; dp : Word) : integer;`
Se encarga de agregar una nueva conexión al pool de conexiónes existentes.

`procedure delConnection(i : integer);`
Elimina una conexión del pool.

`function searchConn(sa : LongWord; sp : Word; da : LongWord; dp : Word) : integer;`
Busca una conexión y retorna su puntero del pool de conexiones.

`procedure putData(i : integer; data : pointer; len : integer);`
Agrega información (datos de una imagen) a una conexión.

`function analyzedConn(i : integer) : boolean;`
Verifica si una conexión ya fue analizada.

`procedure finConn(i : integer);`
Marca la conexión para morir.

`function filterOld() : integer;`
Elimina conexiones viejas o marcadas para morir.

`function conn2disk(i : integer): Cardinal;`
Guarda los datos de una conexión en el disco.

`procedure setContentConn(i : integer; e : string);`
Setea el tipo de datos que contiene la conexión.

`function getConnection(i : integer) : Ptconn;`
Lee una conexión completa.

Mosaico

Se encarga de mostrar las imágenes en posiciones aleatorias del mural.

ImagesUnit

Es la encargada de reconocer si el contenido del primer paquete de datos de una conexión posee carga de utilidad, esto es, si los datos son una imagen GIF, JPG, bien por si solos o dentro de un encabezado HTTP.

```
function itWorth(var data : pointer; var len : integer; var isA : string)
: boolean;
```

es la principal y llama a isHTTP, isGIF, isJPG determinando si data es importante (evalúa con cortocircuito).

```
function isGIF(data : pointer; len : integer; var isA : string) :
boolean;
```

determina si *data* es el encabezado de una imagen GIF.

```
function isJpg(data : pointer; len : integer; var isA : string) :
boolean;
```

determina si *data* es el encabezado de una imagen JPG.

```
function isHTTP(var data : pointer; var len : integer; var isA :
string) : boolean;
```

determina si data es el comienzo de una imagen JPG dentro de un encabezado HTTP.

GifImage

Unidad open source desarrollada por Anders Melander, Filip Larsen y Reinier Sterkenburg.
Esta unidad provee el soporte para el manejo de imágenes GIF. Es de uso libre y el copyright pertenece a Anders Melander. No ha sido modificada en lo absoluto en Snipho.
Define el tipo de datos TGIFImage y de la cantidad de métodos que provee, en Snipho, solo usaremos el método LoadFromStream(I : MemoryStream) que se encarga decodificar un buffer de memoria como una imagen GIF, que luego sera mostrada en pantalla.

Winsock 2.0

Snipho utiliza funciones pertenecientes a Microsoft Winsock 2 para obtener las interfaces de red disponibles en el sistema, colocar la interface seleccionada en modo promiscuo y recibir los paquetes de información (entre las más importantes).

Lista de interfaces

función `WSAIoCtl` subfuncion `SIO_GET_INTERFACE_LIST`.

Devuelve un arreglo de tipo

```
INTERFACE_INFO = packed record
    iiFlags : u_long;
    iiAddress : sockaddr_gen;
    iiBroadcastAddress : sockaddr_gen;
    iiNetmask : sockaddr_gen;
```

con tantas componentes como interfaces haya.

Modo Promiscuo

función `WSAIoctl` subfuncion `SIO_RCVALL`.

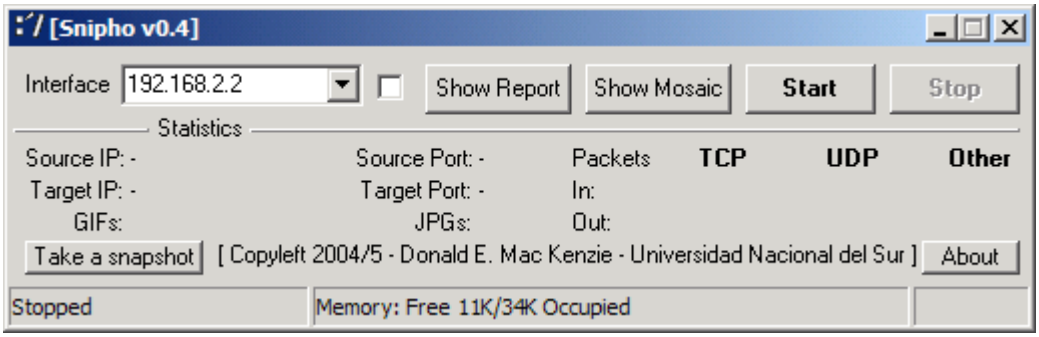
Le permite al socket activo recibir todos los paquetes que circulen por la red. Esta función solo esta disponible en Windows 2000 o superior. Necesita privilegios de administrador para poder ser usada.

WMASyncSelect

Esta función es notificada cada vez que un nuevo paquete de datos es leído. Desde aquí es en donde se llama al analizador de contenidos.

Para mayores referencias sobre Winsock 2, favor de visitar:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/winsock/windows_sockets_start_page_2.asp

Forma de uso



1- Ejecutar el archivo *snipho.exe*.
2- Seleccionar la interface de red a usar desde el *drop-down-box interface*.
3- Presionar *Start* para comenzar el *sniffing*.
4- Sentarse y esperar/navegar.
5- Presionar Stop para detenerlo.

Mientras el programa esta actuando se puede visualizar el mosaico con las imágenes que han sido vistas circulando la red haciendo click sobre el botón *Show Mosaic*.
El botón *Show Report* se encarga de desplegar una nueva ventana mostrando el *log* de operaciones, entre estas: conexiones establecidas, eliminadas, agregado de datos a una cierta conexión, imagen recibida completamente, etc.
Haciendo click sobre *Take a snapshot* toma una captura del mosaico y la almacena en el directorio actual con el nombre "SmosXXX.jpg" siendo XXX un entero positivo.
El botón *About* muestra un una mini ventana con los datos del desarrollador.

Importante:
Snipho no funcionará en sistemas operativos Microsoft Windows inferiores a la versión 2000, además requiere ser ejecutado con privilegios de administrador.

Pruebas reales

Referencias

- AbdelallahElhadj, H., Khelalfa, H., & Kortebi, H. (2002). An Experimental Sniffer Detector: SnifferWall. Basic Software Laboratory, CERIST.
- Hawes, Christopher., & Naghibi, Farzaneh. (2002). CSIC 471 Security Project: Detecting Sniffers. St. Francis Xavier University.
- Stevens, Richard. (2003). TCP/IP Illustrated, Volume 1: The Protocols. Addison Wesley.
- RFCs:
  - 791 (Internet Protocol).
  - 793 (Transmission Control Protocol).
  - 1122 (Communication Layers).
  - 1945 (HTTP Protocol 1.0).
  - 2068 (HTTP Protocol 1.1).
- www.borland.com/
- www.wikipedia.org/
- www.msdn.com/
- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/winsock/windows_sockets_start_page_2.asp

Advertencia: nunca usar Snipho con el fin de invadir la privacidad de las personas sin consentimiento previo.

Comentario general

La documentación fue escrita usando OpenOffice 2.4.0.

Las imágenes fueron creadas usando Inkscape 0.46.

Los PDFs fueron leídos usando Evince Document Viewer 2.22.1.1.

Las páginas web fueron visitadas con Opera 9.50 Beta 2 y Firefox 3.0b5.

El sistema operativo usado para ejecutar todas las aplicaciones fue Ubuntu 8.04 – Kernel 2.6.24-16.

El autor de ésta documentación esta a favor del movimiento Open Source.

Este documento se encuentra bajo la licencia *Creative Commons.*

```
// EOF! (yay!)
```