Resumen Teórica 9 : Seguridad en Redes

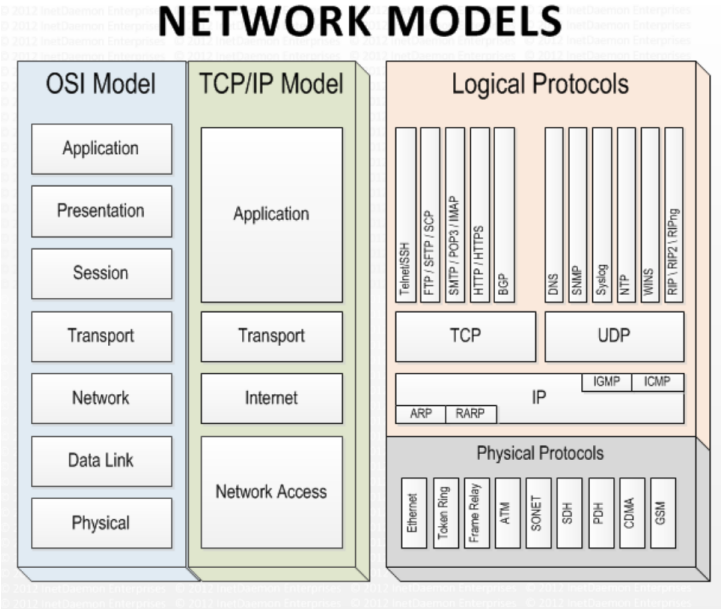
Tomás F. Melli

September 2025

Índice

1 Modelo TCP/IP

El modelo **TCP/IP** es un conjunto de protocolos organizados en capas que permite que computadoras y dispositivos se comuniquen en redes (ya visto en Redes). Un repaso no viene mal :

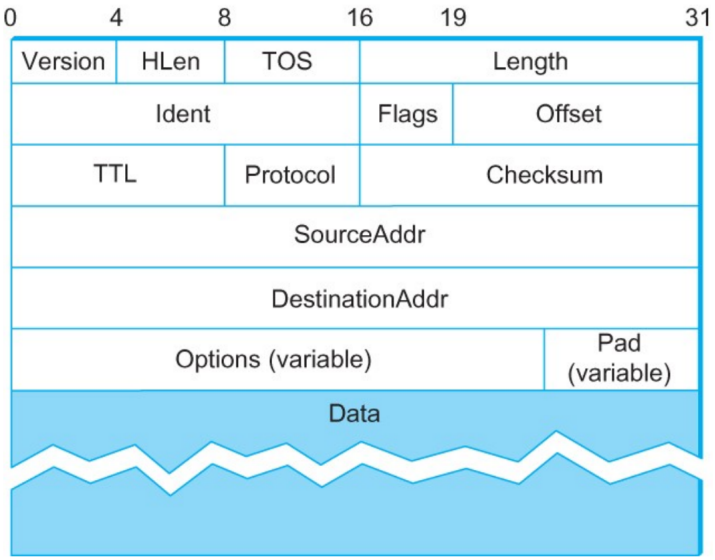


1. **Capa de Aplicación** : su función es brindar servicios a los usuarios y aplicaciones. Los protocolos utilizados son HTTP/HTTPS, SMTP, DNS, FTP, SSH, etc.
2. **Capa de Transporte** : Garantiza que los datos lleguen (TCP) o prioriza velocidad (UDP). Protocolos: TCP (confiable, orientado a conexión) y UDP (rápido, sin control de errores).
3. **Capa de Internet**: Direccionamiento y enrutamiento de paquetes. El protocolo principal es IP (IPv4 / IPv6).
4. **Capa de Acceso a Red (Enlace / Física)** : Entregar los datos físicamente en la red local. Los protocolos utilizados son : Ethernet, Wi-Fi, PPP, etc.

1.1 Datagrama IPv4

Un **datagrama IPv4** es la unidad básica de información que se transmite en la capa de Internet del stack TCP/IP. Es un *paquete de datos* compuesto por un (*header*) y un (*payload*).

Estructura

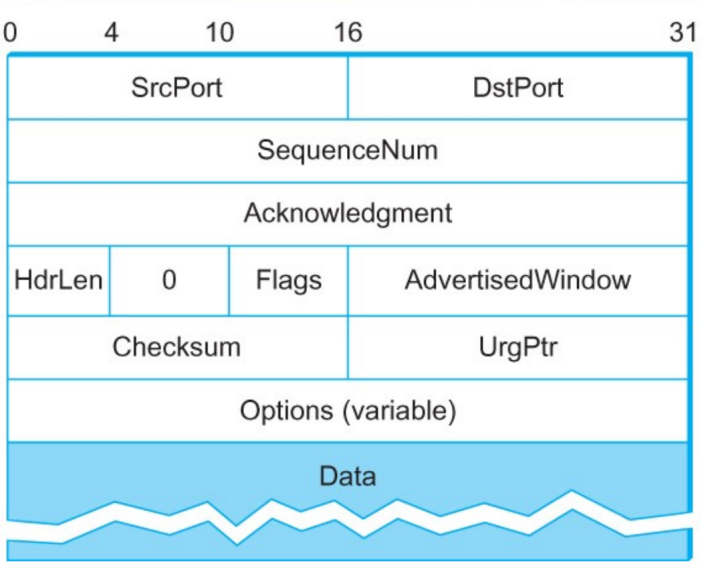


- **Version (4 bits):** indica el protocolo IP (4).
- **HLen (Internet Header Length):** tamaño del header.
- **Type of Service (TOS/DSCP/ECN):** prioridad y calidad de servicio.
- **Length:** tamaño total del datagrama (header + datos).
- **Identification, Flags, Fragment Offset:** utilizados en la **fragmentación**.
- **Time to Live (TTL):** cantidad máxima de saltos permitidos.
- **Protocol:** protocolo de la capa superior (ej. 6 = TCP, 17 = UDP, 1 = ICMP).
- **Header Checksum:** verificación de errores en el header.
- **Source Address:** dirección IP de origen.
- **Destination Address:** dirección IP de destino.
- **Options (opcional):** información de control o depuración.
- **Pad (opcional):** Para que el header siempre sea múltiplo de 32 bits, se agrega padding si es necesario.
- **Data (payload)** : lo que el datagrama transporta.

1.2 TCP (Transmission Control Protocol)

Protocolo de transporte orientado a conexión. Provee confiabilidad ya que garantiza la entrega de datos en orden y sin pérdidas. Implementa control de flujo para evitar saturar al receptor. Incluye mecanismos de control de congestión para adaptarse al estado de la red. Segmenta los datos en unidades más pequeñas llamadas segmentos TCP.

Estructura del paquete TCP



El header TCP tiene un tamaño mínimo de 20 bytes:

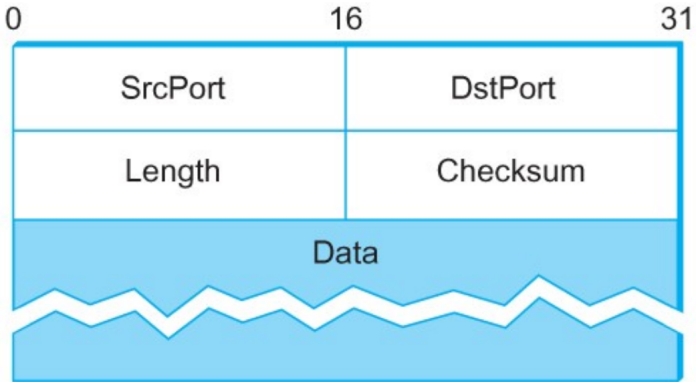
- **Source Port / Destination Port (16 bits cada uno):** identifican los procesos origen y destino.
- **Sequence Number (32 bits):** número de secuencia del primer byte de datos en el segmento.
- **Acknowledgment Number (32 bits):** confirma la recepción de datos previos.
- **Data Offset (4 bits):** tamaño de la cabecera.
- **Flags (6 bits principales):** URG, ACK, PSH, RST, SYN, FIN.
- **AdvertisedWindow (16 bits):** tamaño de ventana de recepción (control de flujo).

- **Checksum (16 bits):** verificación de errores en la cabecera y datos.
- **Urgent Pointer (16 bits):** datos urgentes.
- **Options (opcional) + Padding:** información adicional y relleno.
- **Data (payload):** datos de la aplicación.

1.3 UDP (User Datagram Protocol)

Protocolo de transporte no orientado a conexión. Muy sencillo, con header de solo 8 bytes. No provee confiabilidad: no hay control de flujo ni recuperación de errores. Alta velocidad, útil para aplicaciones que priorizan latencia sobre confiabilidad.

Estructura del datagrama UDP



El header UDP tiene un tamaño fijo de 8 bytes:

- **Source Port (16 bits):** puerto de origen.
- **Destination Port (16 bits):** puerto de destino.
- **Length (16 bits):** longitud total del datagrama (header + datos).
- **Checksum (16 bits):** verificación de errores.
- **Data (payload):** datos de la aplicación.

1.4 ICMP (Internet Control Message Protocol)

Protocolo de control usado en la capa de Internet. No transporta datos de usuario, sino mensajes de control y error. Se encapsula dentro de un datagrama IP (Protocol = 1). Herramienta más conocida: **Ping**, que utiliza mensajes ICMP Echo Request y Echo Reply.

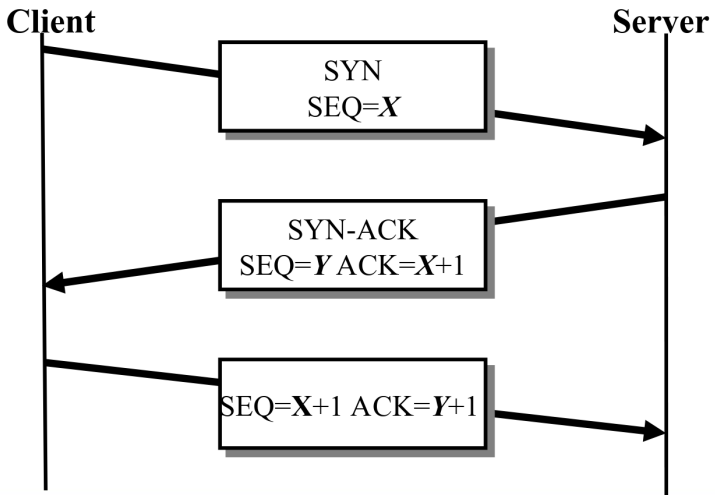
Estructura de un mensaje ICMP

El header ICMP tiene un tamaño mínimo de 8 bytes:

- **Type (8 bits):** tipo de mensaje (ejemplo: 0 = Echo Reply, 8 = Echo Request, 3 = Destination Unreachable).
- **Code (8 bits):** código específico según el tipo.
- **Checksum (16 bits):** verificación de errores.
- **Resto del header:** depende del tipo y código.
- **Data (payload):** información adicional, como parte del paquete original que causó el error.

1.5 Three-way handshake (TCP)

El **three-way handshake** es el proceso de establecimiento de conexión en TCP. Permite que el cliente y el servidor sincronicen sus números de secuencia iniciales y acuerden abrir una conexión confiable antes de transferir datos.



Pasos del Handshake

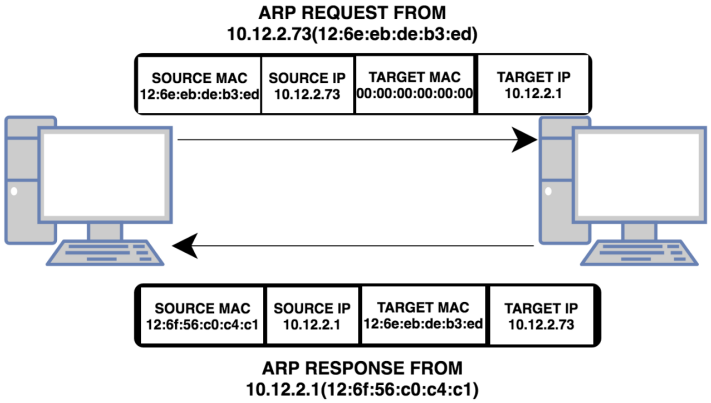
1. **SYN (Synchronize)**
   - El cliente envía un segmento TCP con el flag **SYN = 1**.
   - Incluye un número de secuencia inicial (*ISN*) llamado x.
   - Estado del cliente: **SYN-SENT**.
2. **SYN-ACK (Synchronize + Acknowledge)**
   - El servidor responde con **SYN = 1 y ACK = 1**.
   - **Seq = y** (ISN del servidor).
   - **ACK = x + 1** (confirma recepción del SYN del cliente).
   - Estado del servidor: **SYN-RECEIVED**.
3. **ACK (Acknowledgment)**
   - El cliente responde con **ACK = y + 1**.
   - **Seq = x + 1** (siguiente byte a enviar).
   - Ambos estados pasan a **ESTABLISHED** y la conexión está lista para transferir datos.

El Three-Way Handshake garantiza que TCP establezca una conexión confiable y sincronizada antes de que cualquier dato de la aplicación se transfiera, mediante tres pasos precisos con secuencias y ACK.

2 ARP (Address Resolution Protocol)

El **ARP** es un protocolo definido en la RFC 826 que se utiliza en redes IPv4 para resolver direcciones IP en direcciones MAC (Media Access Control : Se utiliza para identificar de manera única a cada interfaz de red dentro de una LAN. Normalmente viene grabada en el hardware (NIC, tarjeta de red) y tiene formato de 48 bits, representados en hexadecimal. Son **únicas**). Dado que los dispositivos en una LAN se comunican mediante direcciones físicas (MAC), pero los protocolos y aplicaciones usan direcciones lógicas (IP), ARP actúa como un traductor entre ambos tipos de direcciones.

Funcionamiento básico



1. **Petición ARP (ARP Request)**
   - Un host necesita enviar un paquete a una IP determinada en la misma red local.
   - Comprueba su **tabla ARP** para ver si ya conoce la MAC correspondiente.
   - Si no la conoce, envía un **broadcast ARP Request** preguntando: *"¿Quién tiene la IP X.X.X.X?"*
2. **Respuesta ARP (ARP Reply)**
   - El host que posee la IP solicitada responde con un **unicast ARP Reply**, indicando su dirección MAC.
   - El remitente actualiza su **tabla ARP** con el nuevo mapeo IP → MAC (cache temporal).

Tabla ARP

- Cada equipo mantiene una **tabla ARP** o **cache ARP** para guardar resultados recientes.
- Esta cache es temporal y los registros suelen expirar después de cierto tiempo, dependiendo del sistema operativo.

3 Sniffers (Husmeadores de Paquetes)

Un **sniffer** es un programa de software o un dispositivo de hardware que permite capturar, registrar y analizar el tráfico que circula por una red digital.

Mientras los datos viajan por la red, el sniffer puede:

- Capturar cada paquete.
- Decodificar y analizar su contenido según estándares, protocolos y reglas definidas.

Modo de operación

Los sniffers suelen funcionar en **modo promiscuo**, lo que significa que:

- Escuchan **todo el tráfico** que pasa por el medio de comunicación.
- No se limitan únicamente a los paquetes destinados al equipo donde se ejecuta la herramienta.

Dependiendo de la **infraestructura de la red** (switch, hub, Wi-Fi, VLAN, etc.), un sniffer puede:

- Capturar **todo el tráfico** de la red.
- Capturar **solo una parte del tráfico** desde un equipo específico.

Usos de los sniffers

Los sniffers pueden ser utilizados tanto para **administración de redes** como para **ataques o espionaje**:

- **Administración y monitoreo de red:**
  - Analizar problemas de rendimiento o errores en la red.
  - Monitorear el uso de la red y generar estadísticas.
  - Hacer ingeniería inversa de protocolos de red para depuración o desarrollo.
- **Seguridad y ataques:**
  - Detectar intentos de intrusión o actividades sospechosas.
  - Obtener información sensible de usuarios, como contraseñas o datos de sesión.
  - Recopilar información para planificar ataques posteriores.

Ejemplos de sniffers

- **Tcpdump:** herramienta de línea de comandos para captura de paquetes en redes Unix/Linux.
- **Wireshark (anteriormente Ethereal):** sniffer gráfico con análisis detallado de protocolos.
- **ngrep:** permite filtrar paquetes según expresiones regulares.
- **dsniff:** conjunto de herramientas para sniffing y auditoría de redes.
- **Kismet:** sniffer especializado en redes inalámbricas (Wi-Fi).

3.1 Ejemplo : Establecimiento de la conexión (tcpdump)

```
# tcpdump -n -i eth0 -S port 9 (servicio discard)
16:38:21.644505 IP 192.168.0.1.1912 > 192.168.0.99.9: S 3617094593:3617094593(0)
    win 65535 <mss 1460,nop,nop,sackOK>
16:38:21.644603 IP 192.168.0.99.9 > 192.168.0.1.1912: S 3448904776:3448904776(0)
    ack 3617094594 win 5840 <mss 1460>
16:38:21.644720 IP 192.168.0.1.1912 > 192.168.0.99.9: . ack 3448904777 win 65535
```

Línea 1

- **Timestamp:** 16:38:21.644505, indica cuándo se capturó el paquete.
- **IP 192.168.0.1.1912 ¿ 192.168.0.99.9:** paquete de origen hacia destino.
- **S:** SYN, inicio de conexión TCP.
- **3617094593:3617094593(0):** números de secuencia, sin datos aún.
- **win 65535:** ventana de recepción máxima.
- **¡mss 1460,nop,nop,sackOK¿:** opciones TCP (Maximum Segment Size, padding, soporte SACK).

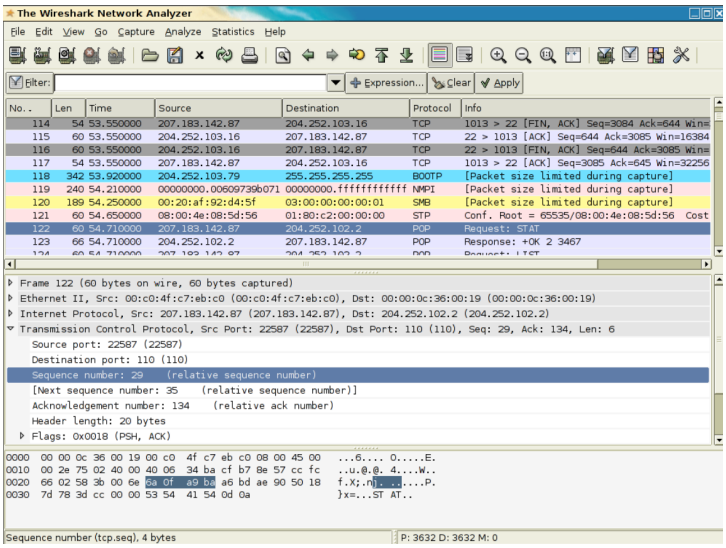
Línea 2

- Respuesta del servidor al cliente.
- **S:** SYN para sincronizar número de secuencia del servidor.
- **ack 3617094594:** confirma la recepción del SYN del cliente.
- **win 5840:** tamaño de ventana del servidor.
- **¡mss 1460¿:** tamaño máximo de segmento aceptado por el servidor.

Línea 3

- Respuesta final del cliente completando el **three-way handshake**.
- **. :** paquete ACK.
- **ack 3448904777:** confirma recepción del SYN del servidor.
- **win 65535:** ventana de recepción del cliente.

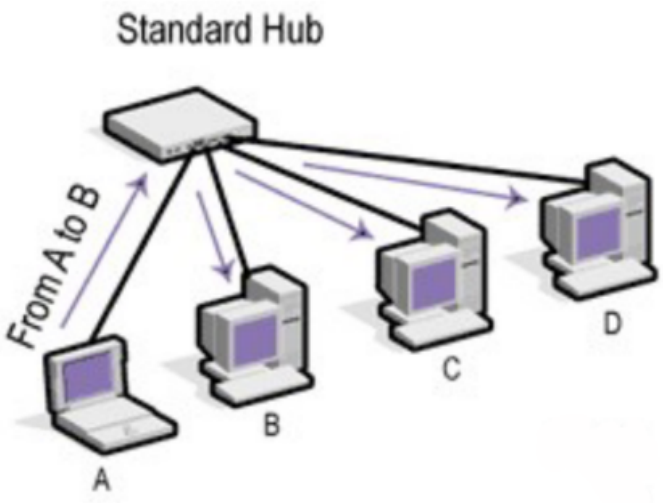
3.2 Ejemplo : Wireshark



4 Hub vs Switch

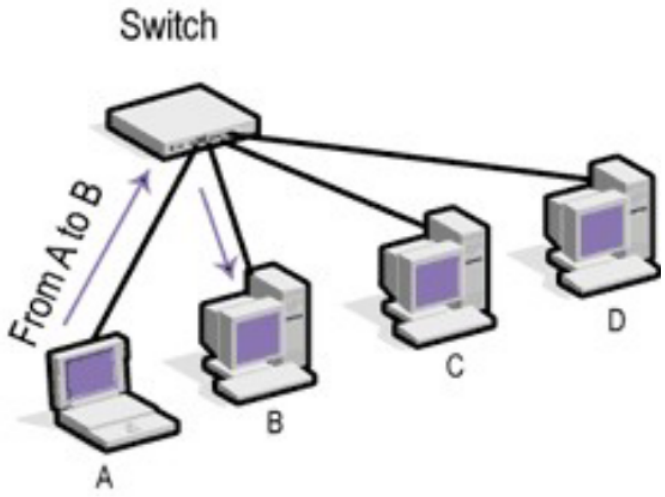
En las redes LAN, los dispositivos de interconexión permiten que múltiples equipos se comuniquen entre sí. Entre estos dispositivos, los más comunes son los **hubs** y los **switches**. Aunque ambos cumplen la función de conectar dispositivos, su manera de manejar el tráfico de red y su eficiencia son muy diferentes.

- **Hub:** Un hub es un dispositivo simple que opera en la capa física (Capa 1) del modelo OSI. Cuando recibe un paquete en uno de sus puertos, lo retransmite a todos los demás puertos sin distinguir el destino, lo que provoca frecuentes colisiones y reduce la eficiencia de la red. No mantiene información sobre las direcciones MAC de los dispositivos conectados, por lo que su funcionamiento es básico y limitado.



- **Switch:** Un switch opera en la capa de enlace de datos (Capa 2) y es mucho más inteligente. Utiliza una tabla de direcciones MAC para enviar cada paquete únicamente al puerto correspondiente al dispositivo de destino. Esto reduce

las colisiones y mejora significativamente la eficiencia de la red. Además, los switches modernos permiten segmentación de la red mediante VLANs y ofrecen un control avanzado del tráfico, haciendo que la comunicación entre dispositivos sea más rápida y segura.



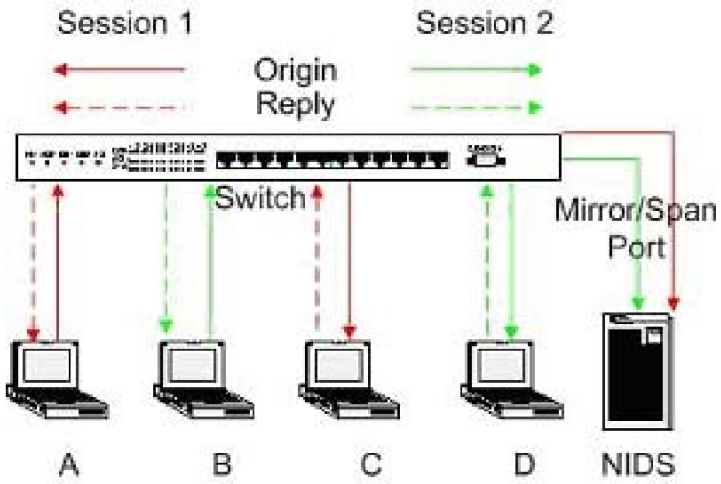
4.1 Monitoreo se de redes switcheadas

4.1.1 Port Mirroring (SPAN)

En las redes modernas, la mayoría de los switches con funcionalidades de administración ofrecen herramientas para monitorear el tráfico de la red de manera eficiente. Una de las técnicas más utilizadas es la configuración de un puerto de monitoreo, también conocido como port mirroring o Switch Port Analyzer (SPAN).

Cuando se habilita esta funcionalidad, el switch copia todo el tráfico que entra y sale de uno o varios puertos seleccionados hacia el puerto de monitoreo. De esta manera, un analista o una herramienta de sniffing conectada a ese puerto puede observar, capturar y analizar el tráfico sin interferir en la operación normal de la red. Esto permite detectar problemas, analizar el rendimiento, identificar patrones de uso y, si se utiliza con fines de seguridad, supervisar posibles intrusiones o actividades sospechosas.

El monitoreo mediante port mirroring es especialmente útil en redes switcheadas, ya que, a diferencia de los hubs, los switches envían los paquetes solo al puerto de destino, lo que normalmente impediría que un sniffer observe todo el tráfico. Con un puerto SPAN, es posible superar esta limitación y obtener una vista completa del tráfico relevante sin afectar el funcionamiento de la red.



4.1.2 Network Tap

n Network TAP (Test Access Point) es un dispositivo físico utilizado para monitorear el tráfico de red de manera pasiva y no intrusiva. Se inserta directamente en un enlace de comunicación entre dos dispositivos, copiando todo el tráfico que pasa por ese enlace hacia uno o varios puertos de monitoreo sin interferir en la operación normal de la red.

A diferencia del port mirroring (SPAN), que depende de la funcionalidad interna de un switch y puede generar cierta carga adicional, un Network TAP garantiza la captura completa y confiable de todo el tráfico sin afectar el rendimiento del

enlace ni de los dispositivos conectados. Esto lo hace especialmente útil para aplicaciones de seguridad, análisis de tráfico en tiempo real, auditorías y pruebas de rendimiento.

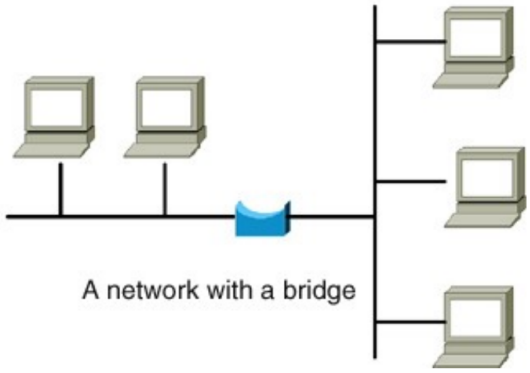
Los TAPs pueden ser de diferentes tipos, incluyendo:

- TAPs de cobre o fibra óptica, según el medio de transmisión.
- TAPs de unidireccional o bidireccional, dependiendo de si capturan el tráfico en un solo sentido o en ambos sentidos simultáneamente.
- Algunos modelos incluyen filtrado o agregación de tráfico para optimizar la captura hacia herramientas de monitoreo.

4.1.3 Dispositivos Inline

Los dispositivos inline son sistemas de monitoreo o seguridad que se colocan directamente en la ruta del tráfico de red, de manera que todo el tráfico entre dos segmentos de red debe pasar a través de ellos. Esto permite inspeccionar, filtrar o modificar los paquetes en tiempo real antes de que alcancen su destino.

Una forma común de implementar un dispositivo inline es utilizar una estación de trabajo con dos tarjetas de red, configuradas como un bridge transparente. En este modo, la estación actúa como un puente entre dos segmentos de red, dejando pasar el tráfico sin necesidad de asignar direcciones IP a las interfaces de red involucradas. Esto asegura que la comunicación de los equipos conectados no se vea afectada por la presencia del dispositivo.



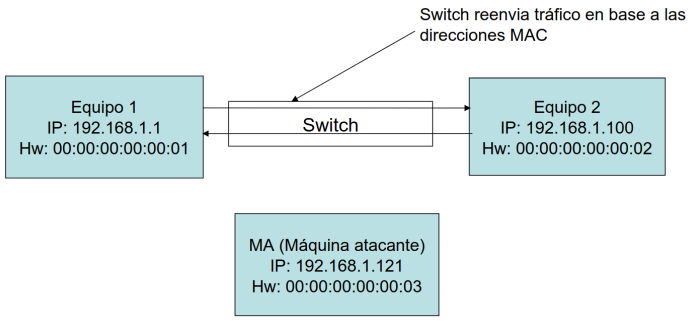
Además, para poder administrar y configurar el dispositivo de manera remota, suele agregarse una tercera interfaz con dirección IP, separada del bridge principal. Esta interfaz permite acceder a la estación para tareas de configuración, monitoreo, actualización de reglas o análisis del tráfico sin interrumpir el flujo normal de datos.

Los dispositivos inline son ampliamente utilizados en soluciones de seguridad de red, como firewalls, sistemas de prevención de intrusiones (IPS) y proxies transparentes, porque permiten intervenir en el tráfico en tiempo real mientras se mantiene la transparencia para los equipos de la red.

5 ARP Spoofing

El **ARP spoofing** (también llamado **ARP poisoning**) es un ataque en redes locales en el que un atacante manipula las tablas ARP de los equipos para interceptar, redirigir o modificar el tráfico que circula entre ellos. Este ataque se aprovecha de la falta de autenticación del protocolo ARP, que permite asociar cualquier dirección IP con cualquier dirección MAC.

Ejemplo



Antes del ataque……..

Consideremos una red con los siguientes equipos y un switch:

- **Equipo 1 (E1):** IP 192.168.1.1, MAC 00:00:00:00:00:01
- **Equipo 2 (E2):** IP 192.168.1.100, MAC 00:00:00:00:00:02
- **Máquina atacante (MA):** IP 192.168.1.121, MAC 00:00:00:00:00:03
- **Switch:** reenvía tráfico basado en direcciones MAC

Antes del ataque, los equipos se comunican normalmente entre sí. Las tablas ARP y MAC relevantes son las siguientes:

Tablas ARP antes del ataque

E1	
192.168.1.100	00:00:00:00:00:02
192.168.1.123	00:00:00:00:00:14
E2	
192.168.1.1	00:00:00:00:00:01
192.168.1.123	00:00:00:00:00:14

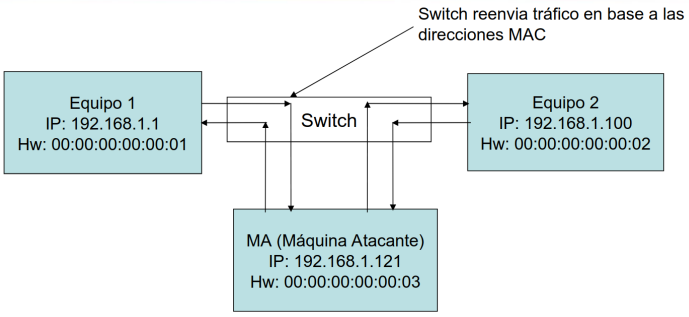
El switch funciona a nivel de direcciones MAC, reenviando los paquetes hacia la máquina correcta según su dirección física. Por lo tanto, E1 y E2 pueden intercambiar datos directamente sin problemas.

Ataque

Durante un ARP spoofing, la máquina atacante envía paquetes ARP falsos a E1 y E2, haciendo que cada uno de ellos asocie la IP del otro equipo con la MAC de la atacante (00:00:00:00:00:03). Como resultado, el switch reenvía todo el tráfico destinado a E1 o E2 hacia la máquina atacante, permitiéndole interceptar, modificar o registrar la comunicación entre ambos equipos.

Tablas ARP después del ataque

E1	
192.168.1.100	00:00:00:00:00:03
192.168.1.123	00:00:00:00:00:14
E2	
192.168.1.1	00:00:00:00:00:03
192.168.1.123	00:00:00:00:00:14



Después del ataque……..

Ahora, la dirección MAC 00:00:00:00:00:03 corresponde a la máquina atacante, lo que significa que todo el tráfico entre E1 y E2 pasa primero por MA antes de continuar hacia su destino.

IP forwarding

Para que la comunicación entre los equipos comprometidos siga funcionando normalmente, la máquina atacante debe ser capaz de reenviar los paquetes recibidos hacia su destino final. Esto se logra habilitando el IP forwarding (reenvío de IP) en el sistema operativo de la atacante.

Si no se habilita el IP forwarding:

- Los paquetes que la atacante recibe de E1 destinados a E2, o viceversa, no serán reenviados.
- Como resultado, la comunicación entre las máquinas atacadas se rompe y los usuarios pueden notar interrupciones o fallos en la red.
- La atacante seguiría viendo el tráfico, pero no permitiría que llegue a su destino, lo que podría delatar el ataque.

6 IP Spoofing

El **IP spoofing** es una técnica en la que un atacante envía paquetes IP con una **dirección de origen falsificada**, haciendo que el host receptor confíe en que el paquete proviene de una máquina legítima. Este ataque se basa en el hecho de que muchos sistemas y aplicaciones **asumen que la dirección IP de origen es verdadera** y no realizan validaciones adicionales.

Motivos para falsificar la IP de origen

Un atacante puede querer modificar la dirección IP de los paquetes por varias razones:

- **Esconder el origen del ataque:** ocultando la dirección IP real del atacante, es más difícil rastrear la fuente de un ataque.
- **Secuestrar una sesión abierta:** si un atacante conoce una sesión legítima entre dos hosts, puede enviar paquetes falsificados para insertarse en esa comunicación.
- **Explotar confianza basada en IP:** algunas aplicaciones confían en la dirección IP de origen para autenticar usuarios o permitir el acceso a recursos. Falsificar la IP puede permitir al atacante evadir controles de seguridad.

Cómo se hace?

El proceso de IP spoofing consiste en **crear paquetes con la dirección IP de origen falsificada**, de manera que el receptor crea que el paquete proviene de un host confiable. Estos paquetes pueden ser enviados para:

- Atacar directamente un host o una red.
- Insertarse en una comunicación existente (MITM parcial).
- Aprovechar vulnerabilidades de servicios que confían en la dirección IP de origen.

7 Ataques de Denegación de Servicio (DoS) basados en IP Spoofing

Los ataques de **denegación de servicio (DoS)** buscan interrumpir el funcionamiento normal de un servicio o sistema, haciendo que deje de estar disponible para usuarios legítimos. Muchos de estos ataques se combinan con **IP spoofing**, donde el atacante falsifica la dirección IP de origen de los paquetes, dificultando la detección y el rastreo de la fuente del ataque. Entre los más relevantes se encuentran el **TCP RST**, el **ICMP contra TCP** y el **SYN Flooding**.

7.1 Ataque TCP RST

El ataque **TCP RST** consiste en enviar un paquete TCP con el **flag RST (reset)** activado hacia una conexión legítima. Si la víctima acepta estos paquetes como válidos, la conexión activa se cierra abruptamente. Para realizar este ataque de forma efectiva, se puede **spoofear la dirección IP de origen** del paquete, de manera que la víctima crea que proviene de un host legítimo.

7.1.1 Importancia de conocer SRC_IP, SRC_PORT, DST_IP y DST_PORT

Para que el paquete RST sea aceptado, el atacante necesita conocer ciertos parámetros de la conexión:

- SRC_IP, SRC_PORT, DST_IP, DST_PORT
- Número de secuencia dentro de la ventana de uso de la conexión TCP

Estos cuatro parámetros forman el **cuarteto de sockets TCP** (*4-tuple*) que identifica de manera única una conexión TCP entre dos hosts.

- **SRC_IP (dirección IP de origen):** Identifica la máquina remota. Si no coincide, el paquete es descartado.
- **SRC_PORT (puerto de origen):** El puerto de la aplicación remota. TCP usa este valor para diferenciar conexiones.
- **DST_IP (dirección IP de destino):** La IP de la víctima que debería recibir el RST.
- **DST_PORT (puerto de destino):** El puerto local de la víctima que participa en la conexión.

Si alguno de estos valores es incorrecto, el paquete será ignorado y la conexión permanecerá activa. Por eso, es fundamental obtenerlos correctamente, ya sea:

- **Sniffeando el tráfico:** Capturando paquetes en la red para leer directamente todos los valores.
- **Adivinando los parámetros:** Intentando predecir los valores, especialmente los números de secuencia y puertos efímeros, aunque es menos confiable.

Este ataque permite **interrumpir sesiones de usuarios o servicios críticos** sin saturar la red.
Ref: Ignoring the Great Firewall of China, Robert Clayton

7.2 Ataque ICMP contra TCP

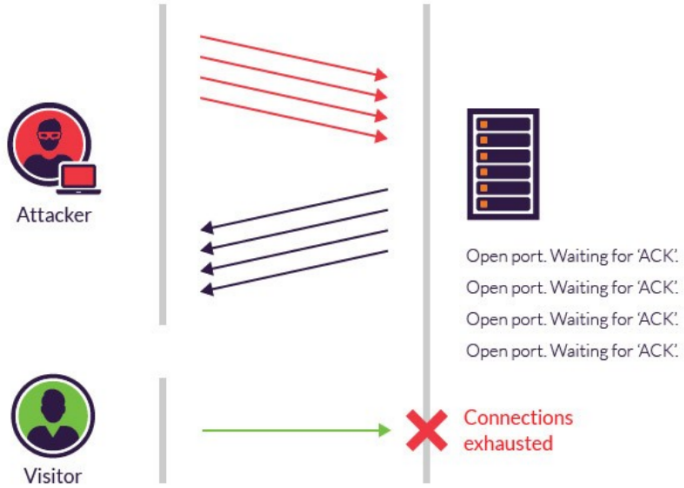
En este ataque, el atacante utiliza paquetes **ICMP** para interrumpir conexiones TCP activas. Existen variantes que permiten **finalizar la conexión sin necesidad de conocer los números de secuencia**, lo que facilita la ejecución del ataque.
Los tipos de mensajes ICMP involucrados incluyen:

- **Protocol Unreachable:** Indica que el protocolo especificado en el encabezado IP del paquete recibido no está soportado por el host receptor. Puede engañar a la víctima para que piense que la conexión TCP es inválida y cerrarla.
- **Port Unreachable:** Señala que el puerto de destino no está abierto o no tiene un servicio escuchando. Si se falsifica este mensaje, la víctima TCP puede cerrar la sesión de manera inmediata.
- **Fragmentation Needed with DF set:** Indica que un paquete IP no puede ser fragmentado debido al flag DF (Don't Fragment). La víctima puede ajustar el tamaño de segmento o incluso cerrar la conexión, interrumpiendo o ralentizando la comunicación.

Al igual que en TCP RST, la dirección IP de origen puede ser **spoofeada**, haciendo que la víctima perciba los paquetes ICMP como provenientes de un host confiable.
Ref: "ICMP attacks against TCP" RFC5927

7.3 Ataque de SYN Flooding

El **SYN Flooding** es uno de los ataques DoS más conocidos contra servicios TCP. Consiste en enviar **una gran cantidad de solicitudes SYN** a un servidor sin completar la fase de handshake. Al usar **IP spoofing**, cada paquete SYN parece provenir de diferentes hosts, lo que **agota los recursos del servidor**, ya que mantiene conexiones semiabiertas esperando la respuesta ACK que nunca llega. Este tipo de ataque puede saturar la pila TCP del servidor y hacer que **nuevas conexiones legítimas sean rechazadas**.



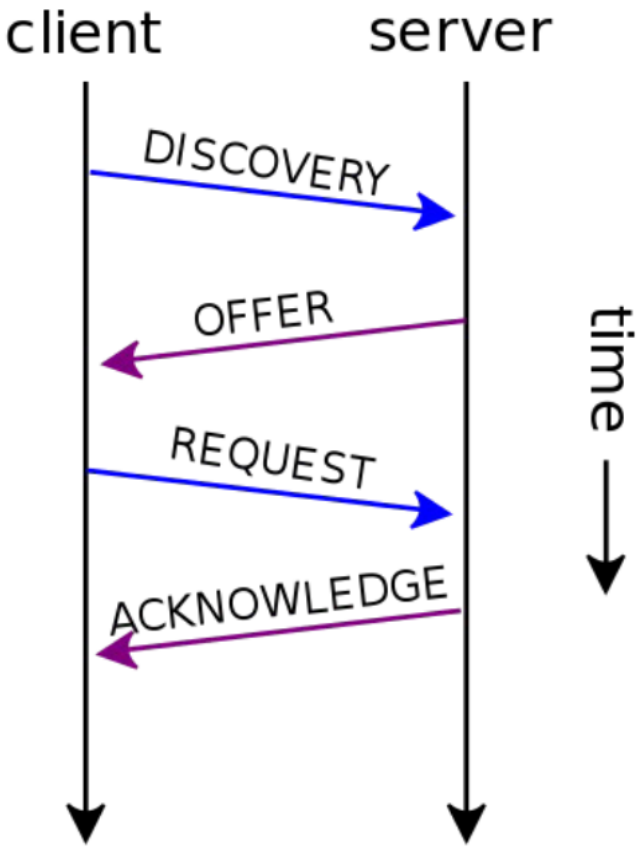
Ref: "TCP SYN Flooding Attacks and Common Mitigations" RFC4987

8 DHCP (Dynamic Host Configuration Protocol)

DHCP (**Dynamic Host Configuration Protocol**) es un protocolo de red que se utiliza para **asignar automáticamente direcciones IP y otros parámetros de configuración de red** a los dispositivos (hosts) dentro de una red. Su función principal es simplificar la administración de la red, evitando la necesidad de configurar manualmente la IP, máscara de subred, puerta de enlace, servidores DNS, entre otros.

8.1 Funcionamiento básico

Cuando un dispositivo se conecta a la red y necesita una dirección IP, el proceso DHCP típicamente sigue estos pasos, conocido como **DORA**:



1. **Discover (Descubrimiento):** El host envía un mensaje de broadcast buscando servidores DHCP disponibles.
2. **Offer (Oferta):** Un servidor DHCP responde ofreciendo una dirección IP y otros parámetros de configuración.
3. **Request (Solicitud):** El host solicita formalmente la dirección IP ofrecida.
4. **Acknowledgment (Confirmación):** El servidor DHCP confirma la asignación y establece la duración del *lease* (tiempo que el host puede usar esa IP).

9 Protocolos de aplicación

En el modelo TCP/IP, los **protocolos de aplicación** operan en la **capa más alta**, encargándose de la **interacción directa con los usuarios y aplicaciones**. Mientras que protocolos de capas inferiores (como IP o TCP) se encargan de enrutar, entregar y asegurar los datos, los protocolos de aplicación definen **cómo se intercambian los datos entre programas** en diferentes hosts.

Estos protocolos proporcionan servicios específicos, por ejemplo:

- **Acceso remoto:** Telnet, SSH, rlogin
- **Transferencia de archivos:** FTP, TFTP
- **Correo electrónico:** SMTP, POP3, IMAP
- **Navegación web:** HTTP, HTTPS
- **Servicios de red y administración:** DNS, SNMP, NTP

Aspectos clave

1. **Orientados al usuario:** Permiten que las aplicaciones interactúen entre sí o con usuarios remotos.
2. **Dependientes de la capa de transporte:** La mayoría se basan en TCP para confiabilidad o UDP para velocidad.
3. **Estandarizados:** Existen normas y RFC que definen su funcionamiento para garantizar interoperabilidad entre diferentes sistemas.

En términos de seguridad, los protocolos de aplicación son especialmente **sujetos a vulnerabilidades**, ya que son la interfaz directa con los usuarios. Por eso, conocer su funcionamiento es fundamental para proteger la información y controlar accesos dentro de una red.

Vamos a ver los siguientes

9.1 Comandos `r`

Hace un tiempo, cuando las redes comenzaron a expandirse, surgió la necesidad de acceder a equipos remotos de forma rápida y sencilla, sin tener que pasar por el mecanismo tradicional de ingresar un usuario y una contraseña en cada conexión. Para responder a esta necesidad se crearon los denominados **comandos r** (remote commands), que ofrecían utilidades específicas para ejecutar tareas sobre máquinas remotas. Entre ellos se destacan:

- **rcp (Remote Copy Protocol):** permite realizar copias de archivos entre sistemas remotos. Utiliza el puerto `TCP/514`.
- **rlogin (Remote Login):** posibilita acceder a una sesión interactiva en un sistema remoto, similar a iniciar sesión localmente. Utiliza el puerto `TCP/513`.
- **rsh (Remote Shell):** brinda acceso a una shell remota para ejecutar comandos directamente en otro sistema. Utiliza el puerto `TCP/514`.
- **rwho (Remote Who):** consulta qué usuarios están actualmente conectados en un equipo remoto. Utiliza el puerto `UDP/513`.

Estos comandos facilitaron considerablemente la administración de sistemas distribuidos en entornos UNIX, ya que reducían la fricción en la interacción con múltiples servidores.

Autenticación

El mecanismo de autenticación de los comandos `r` se basaba en archivos de configuración que definían relaciones de confianza entre equipos y usuarios:

- **/etc/hosts.equiv:** archivo a nivel de sistema que especifica qué máquinas y usuarios remotos son considerados confiables. Si un host aparece en este archivo, sus usuarios pueden acceder sin necesidad de proporcionar contraseña.
- **.rhosts:** archivo a nivel de usuario, ubicado en el directorio home de cada cuenta. Permite que un usuario configure qué equipos remotos y cuentas se consideran de confianza para acceder a su sesión.

Estos mecanismos **eludían los controles estándar basados en usuario y contraseña**, otorgando acceso automático a los hosts definidos como confiables.

Si bien en su momento ofrecieron practicidad, hoy en día representan un **alto riesgo de seguridad**, ya que la confianza podía ser explotada fácilmente en caso de que un atacante comprometiera un host o lograra falsificar su identidad en la red.

9.2 Identd (TCP/113)

El servicio **Identd**, definido en la RFC 1413 y también conocido como **auth**, fue diseñado para proporcionar un mecanismo sencillo de identificación de usuarios en conexiones TCP. Su función principal es devolver el nombre del usuario que originó una conexión en un sistema remoto, de manera que la otra parte pueda verificar con quién está interactuando.

Cuando un usuario o programa en la computadora **A** realiza una consulta **Ident** hacia la computadora **B**, la petición no es general, sino que está limitada a las conexiones establecidas entre ambos hosts. El cliente debe especificar los números de puertos involucrados en la conexión (el puerto local y el remoto). A partir de esa información, el servidor **Identd** en **B** busca la conexión en su tabla de sockets y responde con una cadena que contiene el nombre del usuario propietario de dicho proceso.

Este mecanismo fue útil en su momento para aplicaciones como servidores de correo, sistemas de mensajería o servicios de red que deseaban registrar el nombre del usuario asociado a una conexión. Sin embargo, **Identd no provee garantías fuertes de autenticidad ni de seguridad**, ya que depende completamente de la veracidad de la información que devuelva el servidor remoto, el cual puede ser manipulado o configurado de manera engañosa.

9.3 Telnet (TCP/23)

Telnet es un protocolo de acceso remoto que permite iniciar sesión y controlar un sistema remoto mediante una sesión interactiva tipo consola. Opera sobre el puerto **TCP/23** y fue uno de los mecanismos estándar en entornos UNIX y en equipos de red para administración remota.

Características principales

- Permite el **login remoto** mediante un prompt interactivo similar al de una terminal local.
- El **tráfico viaja en claro** (sin cifrado): tanto credenciales como comandos se transmiten en texto plano por la red.
- Utiliza el esquema clásico de **usuario y contraseña** para autenticar al usuario.
- Fue **muy utilizado** para la administración de routers, switches, servidores y dispositivos embebidos debido a su simplicidad.
- Es vulnerable a ataques como **Man-in-the-Middle (MitM)**, **sniffing** de credenciales y **Session Hijacking**.

Riesgos

- Un atacante que pueda interceptar el tráfico puede capturar credenciales y comandos transmitidos en claro.
- La ausencia de mecanismos de integridad permite modificar comandos en tránsito.
- Las sesiones Telnet pueden ser secuestradas o reproducidas, otorgando privilegios de la sesión legítima.
- Muchos dispositivos legacy aún exponen Telnet con credenciales débiles o por defecto.

9.4 SSH (TCP/22)

Secure Shell (SSH) es un protocolo de comunicaciones seguras que opera por defecto sobre **TCP puerto 22**. Está diseñado para reemplazar protocolos inseguros de acceso remoto (como Telnet, rsh, rlogin) y para ofrecer un canal cifrado y autenticado entre un cliente y un servidor. SSH permite, entre otras cosas, iniciar sesión remota (login interactivo), transferir archivos (SCP, SFTP) y realizar *tunneling* (reenvío de puertos) de conexiones arbitrarias.

Propiedades de seguridad

SSH proporciona las tres garantías básicas de la seguridad en la comunicación:

- **Confidencialidad:** cifra todo el tráfico entre cliente y servidor (incluyendo credenciales y comandos), evitando que un tercero pueda leer el contenido de la sesión.
- **Integridad:** detecta y previene modificaciones en tránsito mediante MACs y algoritmos autenticados, evitando que los paquetes sean alterados sin detección.
- **Autenticación:** permite verificar la identidad del servidor y del cliente. La autenticación del servidor protege contra suplantación, y la del cliente evita accesos no autorizados.

Tipos de ataques contra los que protege

- **Man-in-the-Middle (MitM) / Spoofing:** la autenticación del servidor previene que un atacante se haga pasar por el destino legítimo.
- **Session hijacking:** la sesión está cifrada y autenticada; no puede secuestrarse sin las claves adecuadas.
- **Sniffing:** todo el contenido, incluidas credenciales y datos, viaja cifrado.
- **Data modification:** los mecanismos de integridad evitan modificaciones en los datos en tránsito.

Autenticación de cliente

SSH soporta varios métodos de autenticación:

- **Usuario y contraseña:** sencillo pero menos seguro, vulnerable a ataques de fuerza bruta.
- **Par de claves públicas/privadas:** método recomendado; la clave pública se copia al servidor y la privada permanece protegida en el cliente.
- **Agente SSH (`ssh-agent`):** permite usar claves sin exponer la clave privada en cada conexión.
- **Autenticaciones adicionales:** certificados SSH, multifactor o integración con sistemas como Kerberos/LDAP.

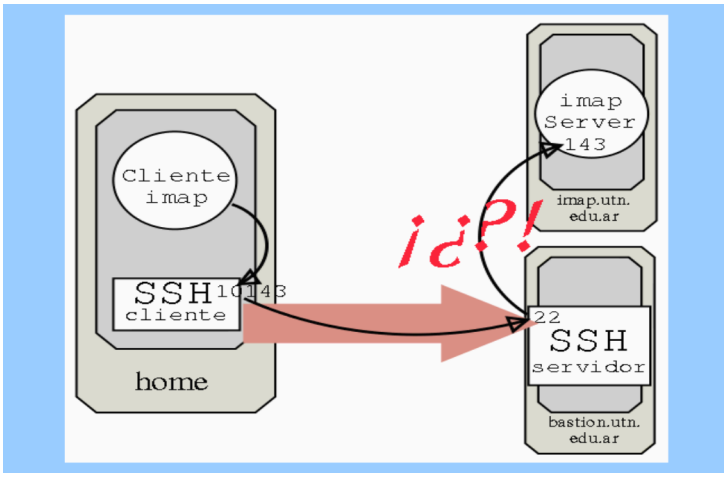
Transferencia de archivos y servicios relacionados

- **SCP / SFTP:** permiten transferencia de archivos cifrada y autenticada.
- **Túneles y reenvío de puertos:** SSH puede redirigir puertos locales o remotos y establecer túneles dinámicos tipo SOCKS.

9.4.1 Túneles SSH — ejemplo local hacia otro servidor

Un uso común es crear un túnel local que conecte un puerto en la máquina cliente con un servicio remoto accesible desde el servidor SSH (bastion/jump host). Ejemplo:

```
ssh -L 10143:imap.utn.edu.ar:143 user@bastion.utn.edu.ar
```



Explicación:

- `-L 10143:imap.utn.edu.ar:143` indica abrir el puerto local 10143; todo lo que se conecte a `localhost:10143` será enviado cifrado al servidor `bastion.utn.edu.ar`, que a su vez se conectará a `imap.utn.edu.ar:143`.
- Después de establecer el túnel, un cliente IMAP (sería justamente el programa de correo) local puede conectarse a `localhost:10143` y comunicarse con el servidor remoto de forma segura.

Otros flags útiles:

- `-R`: reenvío de puerto remoto.
- `-D`: túnel dinámico (proxy SOCKS).
- `-N`: no ejecutar comandos remotos (solo túnel).
- `-f`: enviar a background después de autenticarse.

9.5 TFTP (UDP/69)

El **Trivial File Transfer Protocol (TFTP)** es un protocolo de transferencia de archivos extremadamente simple que funciona sobre **UDP puerto 69**. Su diseño minimalista buscó ofrecer una alternativa ligera a FTP para entornos donde la complejidad o los requerimientos de recursos debían ser mínimos.

Características principales

- Protocolo muy básico de transferencia de archivos: no soporta comandos avanzados ni mecanismos de autenticación.
- Utiliza **UDP**, sin control de sesión ni retransmisiones integradas como en TCP; depende del propio protocolo para manejar confirmaciones y reenvíos.
- No requiere **usuario ni contraseña**; cualquier cliente puede solicitar o enviar un archivo al servidor, siempre que tenga permisos en la configuración.
- Normalmente opera con archivos pequeños o medianos debido a sus limitaciones en velocidad, confiabilidad y seguridad.

Usos comunes

- **Arranque de estaciones de trabajo sin disco rígido:** estaciones o terminales "diskless" descargaban el sistema operativo o el cargador de arranque desde un servidor TFTP.
- **Transferencia de configuraciones de dispositivos de red:** routers, switches y firewalls suelen permitir copiar configuraciones y firmware mediante TFTP.
- **PXE (Preboot Execution Environment):** los entornos de arranque por red utilizan TFTP para descargar la imagen inicial (bootloader) desde el servidor al cliente.
- También puede ser usado para **copiar archivos en equipos comprometidos**, ya que no requiere autenticación ni cifrado.

Seguridad y limitaciones

- No ofrece confidencialidad: los archivos se transfieren en claro.
- No provee autenticación: cualquiera con acceso a la red puede intentar leer o escribir archivos en el servidor.
- Carece de controles de acceso robustos, salvo limitaciones de permisos en el servidor.
- Su dependencia de UDP lo hace más vulnerable a pérdidas de paquetes, y no es adecuado para archivos grandes ni entornos con alta latencia.

9.6 HTTP (TCP/80)

El **HyperText Transfer Protocol (HTTP)** es un protocolo de aplicación diseñado para **sistemas de información hipermediales, distribuidos y colaborativos**. Es el lenguaje que utilizan los clientes y servidores web para comunicarse entre sí.

HTTP es un protocolo **simple, basado en texto**, que **no mantiene estado**. Cada solicitud que un cliente envía al servidor es independiente de solicitudes anteriores, salvo que se utilicen mecanismos adicionales como cookies, sesiones o tokens.

Evolución

- **HTTP/1.1:** texto plano, sin multiplexación, ampliamente usado.
- **HTTP/2:** protocolo binario; encabezados comprimidos, multiplexación de múltiples transmisiones en la misma conexión.
- **HTTP/3:** basado en **QUIC** sobre UDP; mejora tiempos de conexión, reduce latencia y maneja mejor la pérdida de paquetes. Más información en HTTP/3 Explained.

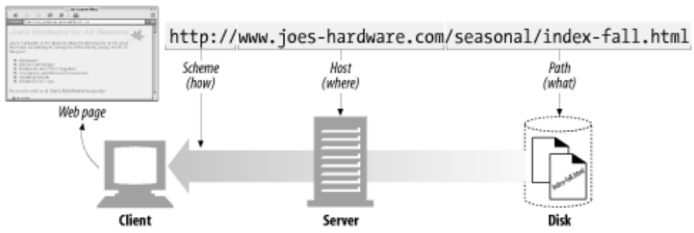
9.6.1 Protocolo cliente-servidor

HTTP sigue un modelo de **Request/Response**:

- El **cliente** envía una solicitud (*request*) al servidor, indicando un método HTTP, un URI y la versión del protocolo.
- La solicitud puede incluir encabezados (*headers*) y un cuerpo (*body*) con datos adicionales.
- El **servidor** responde con un código de estado, encabezados y, opcionalmente, un cuerpo con el contenido solicitado.

9.7 URI y URL

- **URI:** forma estándar de localizar un recurso en Internet.
- **URL:** concepto informal asociado a esquemas populares como HTTP, FTP, MAILTO; es una forma concreta de escribir un URI, aunque el estándar técnico moderno utiliza URI. Ejemplo:



9.7.1 Métodos HTTP

- **GET:** solicita la entidad identificada por el URI.
- **HEAD:** igual que GET, pero el servidor solo devuelve los encabezados.
- **POST:** envía datos al servidor para que los procese o almacene, subordinados al URI.
- **PUT:** solicita al servidor almacenar la entidad enviada en el URI indicado.
- **DELETE:** solicita al servidor eliminar el recurso en el URI indicado.
- **OPTIONS:** pide información sobre los métodos y mecanismos de comunicación disponibles.
- **TRACE:** el servidor devuelve la solicitud tal como la recibió, útil para debugging.

9.7.2 Ejemplo de HTTP Request

```
GET http://www.ejemplo.edu.ar HTTP/1.1
Host: www.ejemplo.edu.ar
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.1) Gecko/2008070208 Firefox/3.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
Content-length: 0
```

- **GET http://www.ejemplo.edu.ar HTTP/1.1** Método GET, URI del recurso y versión HTTP.
- **Host: www.ejemplo.edu.ar** Host al que se dirige la solicitud (obligatorio en HTTP/1.1).

- **User-Agent: Mozilla/5.0 (...)** Identifica el cliente o navegador que hace la solicitud.
- **Accept: text/html,application/xhtml+xml,...** Tipos de contenido que el cliente puede procesar.
- **Accept-Language: en-us,en;q=0.5** Idiomas preferidos para la respuesta.
- **Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7** Codificaciones de caracteres aceptadas.
- **Keep-Alive: 300** Solicita mantener la conexión abierta durante 300 segundos para futuras solicitudes.
- **Proxy-Connection: keep-alive** Similar a Keep-Alive, pero para conexiones a través de un proxy.
- **Content-length: 0** Longitud del cuerpo de la solicitud; 0 en GET ya que no se envía contenido.

9.7.3 Ejemplo de HTTP Response

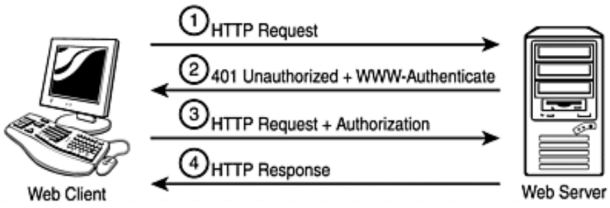
```
HTTP/1.1 200 OK
Date: Tue, 16 Sep 2008 20:53:07 GMT
Server: Apache/1.3.32 (Unix) PHP/5.2.4 Chili!Soft-ASP/3.6.2 mod_ssl/2.8.21 OpenSSL/0.9.7
Last-Modified: Fri, 22 Sep 2006 15:54:45 GMT
ETag: "b1c03a-523c-45140745"
Accept-Ranges: bytes
Content-Length: 21052
Content-Type: text/html

<HTML><HEAD><TITLE>Web interno de la universidad</TITLE></HEAD>
<BODY bgcolor="#CCCC66">
<p><img src="arcert.jpg" width="400" height="40"> </p>
<font face="Verdana, Arial, Helvetica, sans-serif">
<p><b>Web Interno de la universidad</b></p>
```

- **HTTP/1.1 200 OK** Versión HTTP, código de estado (200 = éxito) y mensaje de razón.
- **Date: Tue, 16 Sep 2008 20:53:07 GMT** Fecha y hora en que el servidor generó la respuesta.
- **Server: Apache/1.3.32 (Unix) PHP/...** Información sobre el servidor web y el software usado.
- **Last-Modified: Fri, 22 Sep 2006 15:54:45 GMT** Fecha de última modificación del recurso.
- **ETag: "b1c03a-523c-45140745"** Identificador único del recurso para cache y control de versiones.
- **Accept-Ranges: bytes** Indica que el servidor soporta solicitudes parciales (rango de bytes).
- **Content-Length: 21052** Longitud del cuerpo de la respuesta en bytes.
- **Content-Type: text/html** Tipo de contenido devuelto (HTML en este caso).
- **Cuerpo de la respuesta (HTML)** Contenido real del recurso solicitado, por ejemplo la página web.

9.7.4 HTTP Basic Authentication

HTTP Basic Authentication es un mecanismo de autenticación definido en RFC 2617. Es uno de los métodos más simples que HTTP ofrece para autenticar clientes ante un servidor.

Funcionamiento



- Cuando un cliente solicita un recurso protegido, el servidor responde con un código **401 Unauthorized** y un encabezado:

```
WWW-Authenticate: Basic realm="nombre_del_realm"
```

- El cliente envía nuevamente la solicitud incluyendo el encabezado **Authorization**:

```
Authorization: Basic <credenciales>
```

  donde `<credenciales>` es la combinación de usuario y contraseña codificada en Base64.

- El servidor decodifica la cadena Base64, valida el usuario y la contraseña, y concede o deniega el acceso al recurso.

Características

- **Simplicidad:** fácil de implementar y compatible con todos los navegadores y clientes HTTP.
- **Sin cifrado:** las credenciales viajan codificadas en Base64, pero no cifradas, por lo que pueden ser interceptadas en conexiones HTTP no seguras.
- **Uso recomendado solo sobre HTTPS:** debido a la falta de cifrado, Basic Auth debería usarse únicamente sobre conexiones seguras (HTTPS/TLS).
- **Realm:** permite al servidor definir un área de protección, ayudando al cliente a identificar qué credenciales usar.

9.7.5 HTTPS (HTTP sobre TLS)

HTTPS es la versión segura de HTTP, que combina el protocolo de aplicación HTTP con **TLS (Transport Layer Security)** para proteger la comunicación entre cliente y servidor.

Funcionamiento

- HTTPS cifra toda la información que se intercambia entre el cliente y el servidor, incluyendo solicitudes, respuestas y credenciales de acceso.
- Permite establecer un **canal seguro** que impide que atacantes intercepten o modifiquen los datos en tránsito, protegiendo contra **sniffing**, **Man-in-the-Middle** y modificaciones de contenido.
- Además del cifrado, HTTPS permite **autenticar al servidor**, garantizando que el cliente se comunica con el servidor legítimo. En algunos casos, también puede autenticarse el cliente mediante certificados.

Características principales

- Comunicación cifrada: todo el tráfico viaja protegido por TLS.
- Autenticación de servidor: el cliente verifica la identidad del servidor mediante certificados digitales.
- Integridad de datos: garantiza que los datos no hayan sido alterados en tránsito.
- Puerto estándar: TCP/443.
- Recomendado para toda comunicación sensible en la web, incluyendo login, pagos, transferencia de datos personales, etc.

9.8 Correo Electrónico

El **correo electrónico** es uno de los servicios más antiguos y ampliamente utilizados en Internet. Su funcionamiento se basa en **protocolos estándar** que permiten enviar, recibir y almacenar mensajes entre usuarios y servidores.

9.8.1 Protocolos principales

- **SMTP (TCP/25):** utilizado para el envío de correos entre clientes y servidores y entre servidores.
- **POP3 (TCP/110):** utilizado para la recepción de correos; descarga los mensajes al cliente y, por defecto, los elimina del servidor.
- **IMAP4 (TCP/143):** permite acceder a los mensajes directamente en el servidor, manteniendo la sincronización entre múltiples dispositivos.

Estos protocolos también pueden operar sobre canales cifrados mediante SSL/TLS:

- SMTP sobre SSL/TLS: puerto 465
- POP3 sobre SSL/TLS: puerto 995
- IMAP4 sobre SSL/TLS: puerto 993

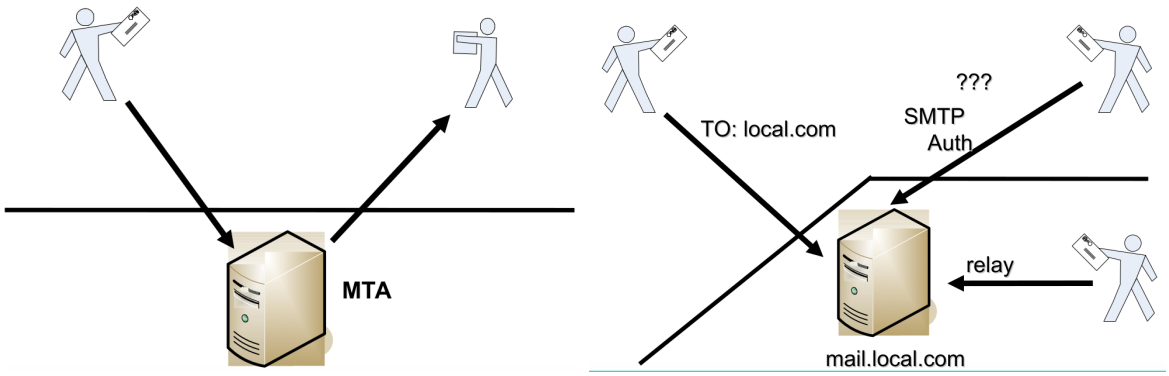
Servidores de correo comunes

- Microsoft Exchange
- Postfix
- Exim
- Sendmail
- Qmail

9.8.2 Ejemplo de correo electrónico

```
Return-Path: <xxx@xxx.com.ar>
Delivered-To: rbaader@ejemplo.edu.ar
Received: from unknown (HELO me) (200.x.x.x)
    by mail.ejemplo.edu.ar with SMTP; 29 Oct 2004 14:00:16 -0000
Received: from ecosport.xxx.com.ar ([127.0.0.1])
    by localhost (chatarra [127.0.0.1]) (amavisd-new, port 10024)
    with ESMTP id 28182-02 for <rbaader@ejemplo.edu.ar>
Received: from of123c (of123_1-T_rbaader.xxx.com.ar [10.1.1.51])
    by chatarra.xxx.com.ar (Postfix) with SMTP id C2AA5204F6A for
    <rbaader@ejemplo.edu.ar>
From: "Fernando X" <xxx@xxx.com.ar>
To: "Rodolfo Baader" <rbaader@ejemplo.edu.ar>
Subject: Re: Un favor!
```

9.8.3 Problemas de seguridad: Open Relay



Un **Open Relay** es un servidor de correo (MTA, Mail Transfer Agent) que permite enviar correos electrónicos desde cualquier dirección, **sin autenticar al remitente**, y hacia cualquier destino.

- No verifica si quien envía el correo tiene permiso para usar el servidor.
- Cualquier persona en Internet podría usarlo para enviar emails, incluidos **spammers**.
- Es considerado una **vulnerabilidad grave**, porque tu servidor puede ser usado para enviar correo masivo no deseado y luego ser listado en **blacklists**.

9.8.4 Cómo funciona

Supongamos un ejemplo:

```
TO: local.com
SMTP
Auth
relay
mail.local.com
```

- Un usuario externo se conecta al servidor `mail.local.com`.
- Si el servidor **no requiere autenticación**, permite que este usuario envíe correos hacia `local.com` o cualquier otro dominio.
- Esto convierte al servidor en un **Open Relay**, vulnerable a abusos.

9.8.5 SMTP-AUTH (RFC 2554)

SMTP-AUTH permite que los clientes se autentiquen ante el servidor antes de enviar correo.

Ejemplo de autenticación LOGIN

El método **LOGIN** es el más simple de autenticación SMTP. El cliente envía el usuario y la contraseña codificados en Base64.

- Base64 no cifra la información, solo la codifica.
- Si se usa sobre una conexión TCP sin TLS/SSL, un atacante podría capturar las credenciales.
- Se recomienda usar LOGIN solo sobre TLS/SSL (por ejemplo, STARTTLS o puerto 465).

Flujo resumido

1. Servidor solicita "Username:" codificado en Base64.
2. Cliente envía el usuario en Base64.
3. Servidor solicita "Password:" codificado en Base64.
4. Cliente envía la contraseña en Base64.
5. Servidor valida y acepta la autenticación.

```
220-mail.xxxxxxxx.com ESMTP Exim 4.34 #1 Wed, 23 Jun 2004 17:35:13 -0700
EHLO mail.myserver.com
250-mail.xxxxxxxx.com Hello mail.myserver.com [192.168.0.156]
250-SIZE 52428800
250-PIPELINING
250-AUTH PLAIN LOGIN
250-STARTTLS
250 HELP
AUTH LOGIN
334 VXNlcm5hbWU6
bXl1c2VybmFtZQ==
334 UGFzc3dvcmQ6
bXlwYXNzd29yZA==
235 Authentication succeeded
```

- **220-mail.xxxxxxxx.com ESMTP Exim 4.34 ...**: Código de estado 220 indica que el servidor SMTP está listo; se muestra el nombre del servidor, versión de Exim y la marca de tiempo.
- **EHLO mail.myserver.com**: Comando del cliente para iniciar la sesión ESMTP; identifica al cliente.
- **250-mail.xxxxxxxx.com Hello ...**: Respuesta del servidor aceptando EHLO y saludando al cliente.
- **250-SIZE 52428800**: Tamaño máximo de mensaje que acepta el servidor (aprox. 50 MB).
- **250-PIPELINING**: Indica que el servidor soporta pipelining (envío de múltiples comandos sin esperar respuesta intermedia).

- **250-AUTH PLAIN LOGIN**: Métodos de autenticación soportados por el servidor (PLAIN y LOGIN).
- **250-STARTTLS**: Indica que se puede usar STARTTLS para cifrar la sesión.
- **250 HELP**: Comando de ayuda disponible.
- **AUTH LOGIN**: El cliente solicita autenticarse usando el método LOGIN (usuario y contraseña codificados en Base64).
- **334 VXNlcm5hbWU6**: Servidor solicita el nombre de usuario; la cadena es Base64 de "Username:".
- **bXl1c2VybmFtZQ==**: Cliente envía el usuario codificado en Base64 (ej. "myusername").
- **334 UGFzc3dvcmQ6**: Servidor solicita la contraseña; la cadena es Base64 de "Password:".
- **bXlwYXNzd29yZA==**: Cliente envía la contraseña codificada en Base64 (ej. "mypassword").
- **235 Authentication succeeded**: Servidor confirma que la autenticación fue exitosa y permite enviar correos.

Ejemplo de autenticación CRAM-MD5

El método **CRAM-MD5** (Challenge-Response Authentication Mechanism) utiliza la función de hash MD5 para autenticar al usuario sin enviar la contraseña en texto plano.

- El servidor envía un **challenge** (cadena aleatoria).
- El cliente combina su contraseña con el challenge y calcula un hash HMAC-MD5.
- El cliente envía solo el hash al servidor, nunca la contraseña.
- El servidor realiza la misma operación y valida el hash; si coincide, la autenticación es exitosa.

```
S: 220 smtp.example.com ESMTP server ready
C: EHLO jgm.example.com
S: 250-smtp.example.com
S: 250 AUTH CRAM-MD5 DIGEST-MD5
C: AUTH FOOBAR
S: 504 Unrecognized authentication type.
C: AUTH CRAM-MD5
S: 334 PENCeUxFREJoU0NnbmhNWitOMjNGNndAZWx3b29kL.mubm9zb2Z0LmNvbT4=
C: h
S: 235 Authentication successful
```

- **S: 220 smtp.example.com ...**: Servidor listo, código 220, nombre del servidor y mensaje de bienvenida.
- **C: EHLO jgm.example.com**: Cliente inicia sesión ESMTP, identificándose como jgm.example.com.
- **S: 250-smtp.example.com**: Servidor acepta EHLO y saluda al cliente.
- **S: 250 AUTH CRAM-MD5 DIGEST-MD5**: Métodos de autenticación soportados por el servidor.
- **C: AUTH FOOBAR**: Cliente intenta autenticarse con un método inválido.
- **S: 504 Unrecognized authentication type.**: Servidor rechaza el método no reconocido.
- **C: AUTH CRAM-MD5**: Cliente solicita autenticarse correctamente con CRAM-MD5.
- **S: 334 PENCeUxFREJo...**: Servidor envía challenge Base64 que el cliente debe usar para calcular HMAC-MD5.
- **C: h**: Cliente responde con el hash HMAC-MD5 calculado.
- **S: 235 Authentication successful.**: Servidor confirma que la autenticación fue exitosa.

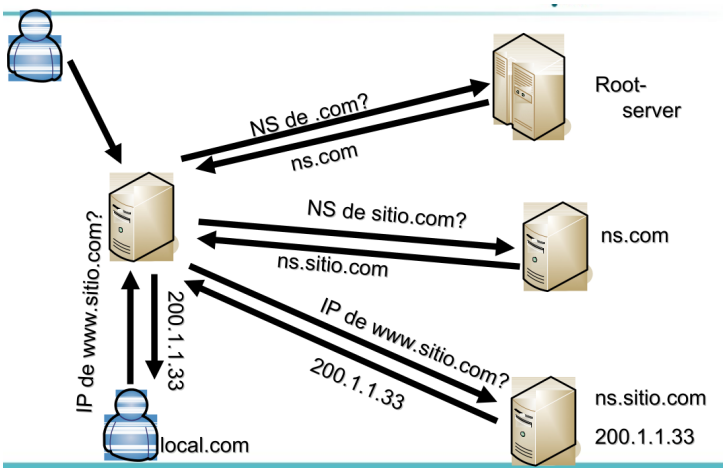
9.9 DNS (Domain Name System)

El **Domain Name System (DNS)** es un sistema fundamental en Internet que permite resolver nombres de dominio en direcciones IP y viceversa. Funciona sobre **UDP/53** y, en ciertos casos, sobre **TCP/53** (como en transferencias de zona o respuestas grandes).

9.9.1 Características principales

- Es un esquema **jerárquico** de resolución de nombres, distribuido en múltiples niveles:
  - **Root servers**: servidores raíz que conocen los servidores de los dominios de primer nivel (TLD, como .com, .org, .edu).
  - **Servidores TLD**: manejan dominios de nivel superior y redirigen a los servidores autoritativos de cada dominio.
  - **Servidores autoritativos**: contienen los registros finales de un dominio, como direcciones IP de hosts y subdominios.
- Existen **servidores primarios y secundarios** para cada dominio, que se comunican entre sí para mantener consistencia de los registros (transferencia de zona).
- Permite resolver nombres de manera **recursiva** o **iterativa** según la configuración del servidor y el tipo de consulta.

9.9.2 Ejemplo de resolución de nombre



Supongamos que se desea conocer la IP de `www.sitio.com`:

- Se consulta al servidor raíz, que indica el servidor TLD para `.com`.
- El servidor TLD redirige al servidor autoritativo de `sitio.com`.
- El servidor autoritativo devuelve la IP final: `200.1.1.33`.

Gráficamente, se tiene la jerarquía:

- Root server → Servidor `.com` → Servidor `sitio.com` → IP de `www.sitio.com`

9.9.3 Seguridad en DNS

El DNS, por ser crítico en la infraestructura de Internet, requiere medidas de seguridad para evitar ataques y filtrado de información:

- **No permitir consultas recursivas** a servidores públicos que puedan ser usados en ataques DDoS.
- **Restringir la transferencia de zonas** (AXFR) solo a servidores secundarios autorizados.
- **Cuidar los nombres de los hosts**, evitando revelar información sensible en los registros DNS.
- Mantener actualizados los servidores y registros DNS para prevenir vulnerabilidades conocidas.
- Implementar mecanismos de privacidad como:
  - **DNS over TLS (DoT)**
  - **DNS over HTTPS (DoH)**

  para cifrar consultas y evitar que un observador pueda rastrear las solicitudes de nombres de dominio.

9.10 SNMP (Simple Network Management Protocol)

El **Simple Network Management Protocol (SNMP)** es un protocolo de aplicación diseñado para la **gestión y monitoreo de dispositivos de red**, como routers, switches, impresoras y servidores. Funciona principalmente sobre **UDP**, usando los puertos **161** (agentes) y **162** (traps).

9.10.1 Versiones de SNMP

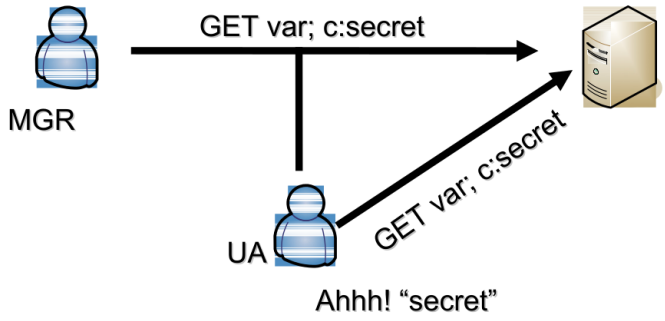
- **v1**: Primera versión; muy básica y sin mecanismos de seguridad robustos.
- **v2**: Añade mejoras en rendimiento y tipos de datos, pero la seguridad sigue siendo limitada.
- **v3**: Introduce seguridad real, con autenticación y cifrado; es la única versión adecuada para ambientes no asegurados.

9.10.2 Operaciones básicas de SNMP

SNMP permite principalmente **consultar y modificar variables** en dispositivos de red mediante **MIBs (Management Information Bases)**.

GET

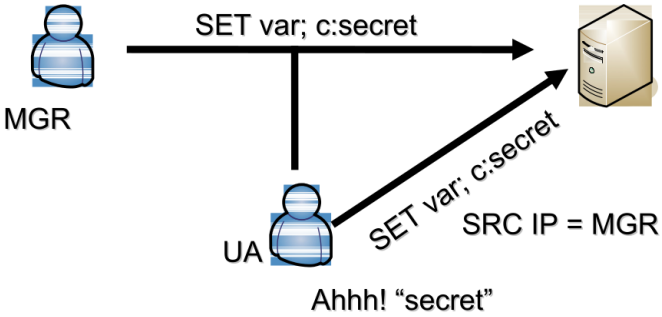
Permite al **gestor (manager)** solicitar el valor de una variable específica de un agente SNMP.



En SNMP v1/v2, la comunidad (‘c:secret‘) actúa como “contraseña” simple, transmitida en texto claro.

SET

Permite al **gestor (manager)** modificar el valor de una variable en el agente.



Al igual que en GET, la comunidad se transmite sin cifrado en v1/v2, lo que representa un riesgo de seguridad. La dirección IP de origen generalmente se valida contra la comunidad permitida (‘SRC IP = MGR‘).

9.10.3 Seguridad en SNMP

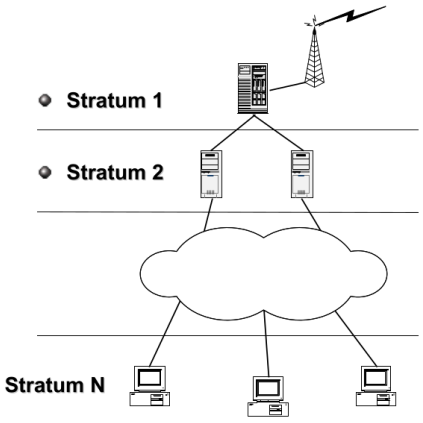
- Las versiones v1 y v2 son **inseguras** en entornos abiertos, ya que las credenciales (comunidades) viajan en texto claro.
- SNMP v3 agrega:
  - **Autenticación** (MD5 o SHA)
  - **Cifrado de datos** (DES, AES)
  - **Control de acceso granular**
- Siempre que sea posible, se recomienda usar **SNMP v3** para proteger información de gestión crítica.

9.11 NTP (Network Time Protocol)

El **Network Time Protocol (NTP)** es un protocolo de aplicación diseñado para **sincronizar los relojes de los sistemas de computación** a través de redes de datos, incluyendo Internet. Funciona sobre **UDP**, usando el puerto **123**.

9.11.1 Características principales

- NTP permite que equipos en una red mantengan **tiempo preciso y consistente**, crucial para registros de eventos, seguridad, transacciones financieras, sistemas distribuidos y auditorías.
- Sitio oficial: http://www.ntp.org
- Funciona mediante un esquema jerárquico de servidores, organizados por **stratum**:



  - **Stratum 1**: servidores de referencia directa, conectados a relojes atómicos o GPS, considerados fuentes de tiempo precisas.
  - **Stratum 2**: servidores que se sincronizan con servidores stratum 1, distribuyendo la hora a otros dispositivos.
  - **Stratum N**: servidores o clientes sincronizados a niveles superiores, formando una jerarquía en cascada.
- Proporciona mecanismos de corrección de desviaciones de reloj y compensación de retrasos de red, permitiendo mantener la hora precisa incluso en redes con latencias variables.

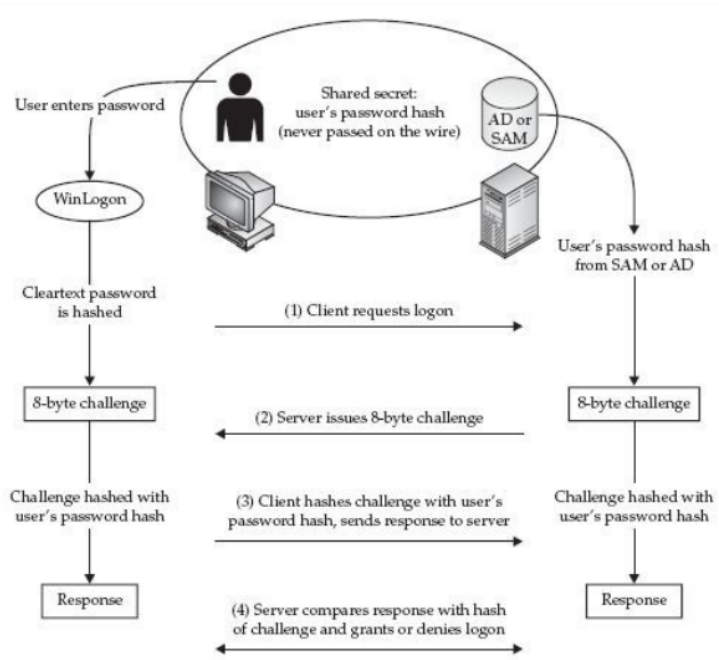
9.11.2 Seguridad

- La versión básica de NTP (v3/v4) incluye **autenticación opcional** basada en claves compartidas.
- Sin seguridad, los servidores NTP pueden ser utilizados para ataques, como **reflejo/amplificación DDoS**.
- Se recomienda utilizar configuraciones seguras, filtros de acceso, y versiones actualizadas del protocolo.

9.12 Autenticación en Windows vía Red: MSV1_0

En entornos Windows, el acceso remoto a recursos de red requiere mecanismos de autenticación que permitan validar la identidad del usuario antes de conceder permisos. Uno de estos mecanismos es **MSV1_0**, utilizado en sistemas anteriores a Kerberos o en compatibilidad con protocolos preexistentes.

9.12.1 MSV1_0: Mecanismo pre-Kerberos



- MSV1_0 es un **módulo de seguridad de Windows (Security Support Provider, SSP)** que implementa autenticación basada en el **protocolo NTLM (NT LAN Manager)**.
- Permite a los clientes autenticarse en servidores Windows mediante desafíos de contraseña, sin enviar la contraseña en texto claro a través de la red.
- Flujo típico de autenticación NTLM vía MSV1_0:
  1. El cliente solicita acceso a un recurso de red.
  2. El servidor envía un **challenge** (valor aleatorio).
  3. El cliente genera una respuesta usando la contraseña del usuario y el challenge.
  4. El servidor valida la respuesta contra la base de datos local o un controlador de dominio.
  5. Si coincide, se concede acceso; de lo contrario, se rechaza la autenticación.
- Este mecanismo es llamado **pre-Kerberos** porque en versiones modernas de Windows se utiliza **Kerberos** como protocolo principal, más seguro y basado en tickets.
- MSV1_0 sigue disponible por compatibilidad con sistemas antiguos, aplicaciones legacy o entornos mixtos donde Kerberos no es aplicable.

9.12.2 Seguridad

- NTLM es vulnerable a ataques de **replay**, **pass-the-hash** y ciertos ataques de fuerza bruta si las contraseñas son débiles.
- Se recomienda, siempre que sea posible, migrar a **Kerberos** y deshabilitar MSV1_0 en entornos modernos.
- Para proteger la autenticación NTLM, se puede usar **NTLMv2**, que mejora la seguridad frente a ataques de interceptación y de repetición.