EL SERVICIO FTP

Descargar PDF

PROTOCOLO FTP

INTRODUCCIÓN

El **File Transfer Protocol (FTP)** es un protocolo de la capa de aplicación diseñado para permitir la transferencia de archivos entre sistemas conectados a una red basada en TCP/IP. Su finalidad principal es proporcionar un mecanismo estándar que permita a un usuario acceder a archivos almacenados en un sistema remoto y realizar operaciones como listar directorios, descargar archivos o subir contenido.

FTP fue durante muchos años uno de los protocolos más utilizados en Internet, especialmente para la publicación de páginas web, la distribución de software y el intercambio de información en redes académicas y empresariales. Aunque en la actualidad ha sido sustituido en muchos entornos por alternativas más seguras, sigue siendo un protocolo fundamental desde el punto de vista formativo, ya que permite comprender claramente el funcionamiento de los servicios de red clásicos.

Desde una perspectiva didáctica, FTP resulta especialmente interesante porque hace explícitos muchos aspectos que en protocolos modernos quedan ocultos, como el uso de múltiples conexiones TCP, la negociación de puertos o el intercambio de comandos en texto plano.

CONTEXTO HISTÓRICO Y ESTANDARIZACIÓN

El protocolo FTP se desarrolló en los primeros años de ARPANET, en una época en la que la seguridad no era una preocupación prioritaria y las redes estaban formadas por un número reducido de equipos de confianza. Su especificación principal está recogida en el **RFC 959**, publicado en 1985, que define el funcionamiento básico del protocolo tal y como sigue utilizándose hoy en día.

A lo largo del tiempo se han publicado otros RFC que añaden extensiones o aclaran determinados comportamientos, pero el núcleo del protocolo apenas ha cambiado. Esto explica por qué FTP conserva decisiones de diseño que hoy pueden parecer poco adecuadas, como la ausencia de cifrado o el uso de múltiples conexiones.

FTP pertenece a la **capa de aplicación** del modelo TCP/IP y utiliza **TCP** como protocolo de transporte, lo que garantiza una comunicación fiable y orientada a conexión.

ARQUITECTURA Y PRINCIPIOS DE FUNCIONAMIENTO

FTP utiliza una arquitectura **cliente-servidor**. El cliente inicia la conexión y solicita servicios, mientras el servidor espera y responde a esas peticiones.

La clave de FTP es que emplea **dos canales de comunicación independientes**:

- **Canal de control**: Se usa para enviar comandos y recibir respuestas (por ejemplo, iniciar sesión, pedir un archivo, cambiar de carpeta, etc.). Este canal permanece abierto durante toda la sesión y normalmente utiliza el **puerto TCP 21** del servidor. La **comunicación es en texto plano (ASCII)**, lo que permite observar fácilmente el intercambio de comandos usando herramientas como `telnet`.
- **Canal de datos**: Se utiliza exclusivamente para transferir archivos o listados de directorios. La **comunicación puede ser en texto plano o binaria**, dependiendo del modo de transferencia. A diferencia del canal de control, esta conexión se abre solo cuando es necesario transferir información y se cierra al

terminar. El **puerto** usado para el canal de datos **varía según el modo de conexión (activo o pasivo)**, lo que puede complicar la configuración en redes con firewalls o NAT.

Esta separación entre canales permite que los comandos y las transferencias de archivos no se mezclen, facilitando la gestión y el control de la sesión FTP.

SESIÓN FTP

El proceso típico de una sesión FTP incluye las siguientes fases:

1. **Conexión inicial**: El cliente abre una conexión TCP al puerto 21 del servidor (canal de control) y recibe un mensaje de bienvenida.
2. **Autenticación**: El cliente envía su usuario (`USER`) y contraseña (`PASS`). El servidor valida las credenciales y permite el acceso.
3. **Comandos y transferencias**: El cliente envía comandos FTP por el canal de control. Estos comandos pueden incluir operaciones de gestión y control que no impliquen transferencia de datos o comandos que sí impliquen transferencia de datos (como `LIST` , `RETR` , `STOR`). Cuando se necesita transferir datos, se abre el canal de datos:
   - **Modo activo**: el servidor se conecta al cliente para transferir datos.
   - **Modo pasivo**: el cliente se conecta al servidor para transferir datos.

   Cuando la transferencia termina, se cierra el canal de datos.
4. **Finalización**: El cliente envía el comando `QUIT` y el servidor cierra la sesión y libera los recursos.

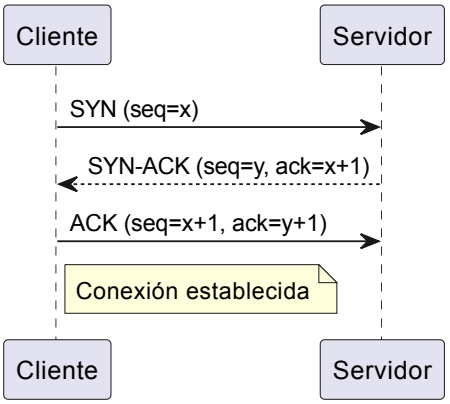
MECANISMOS DE ESTABLECIMIENTO Y CIERRE DE CONEXIONES TCP

Aunque FTP es un protocolo de aplicación, depende completamente de TCP para su funcionamiento. Comprender cómo TCP establece y cierra conexiones es fundamental para entender el flujo real de una sesión FTP, especialmente cuando algo falla o hay comportamientos inesperados en la red.

ESTABLECIMIENTO DE CONEXIÓN: THREE-WAY HANDSHAKE

Cuando un cliente FTP se conecta al servidor, TCP realiza un proceso llamado three-way handshake (apretón de manos de tres vías) para garantizar que ambos extremos están preparados para comunicarse de forma confiable.

- **SYN**: El cliente envía un segmento TCP con el flag SYN activado, indicando su intención de iniciar una conexión. Este segmento incluye un número de secuencia inicial que el cliente ha elegido.
- **SYN-ACK**: El servidor recibe el SYN y responde con un segmento que tiene tanto el flag SYN como el flag ACK activados. Esto confirma que ha recibido el intento de conexión del cliente y a su vez propone su propio número de secuencia inicial.
- **ACK**: El cliente recibe el SYN-ACK y responde con un segmento ACK, confirmando que ha recibido la respuesta del servidor. A partir de este momento, la conexión está completamente establecida y ambos lados pueden comenzar a intercambiar datos.



En el contexto de FTP, este three-way handshake ocurre cada vez que el cliente intenta conectarse al puerto 21 del servidor. Si el servidor no responde en tiempo razonable, el cliente asume que el servidor está inactivo o el puerto no está disponible. Si el servidor rechaza (envía un RST), significa que no hay servicio FTP escuchando en ese puerto.

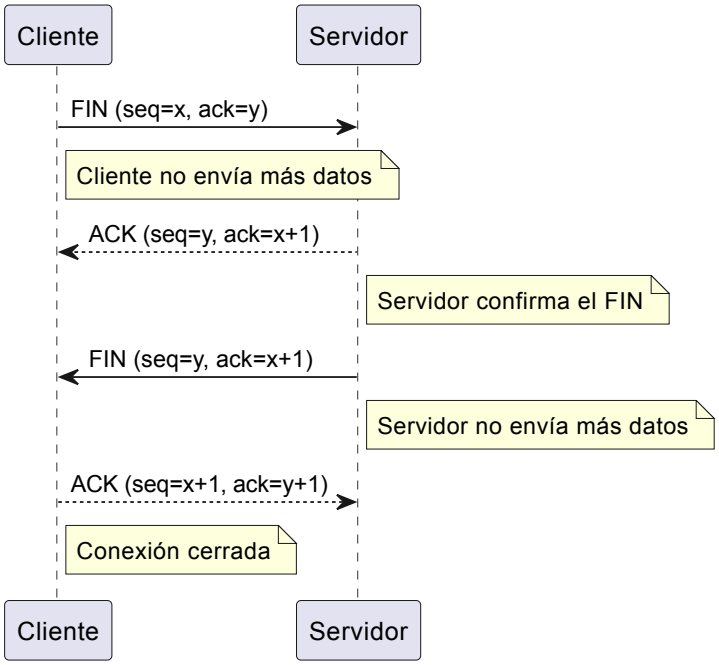
CIERRE DE CONEXIÓN: FOUR-WAY Y TWO-WAY HANDSHAKE

El cierre de una conexión TCP es un proceso más complejo que tiene variantes según la situación. A diferencia del establecimiento (que es relativamente rígido), el cierre puede haber sido iniciado por cualquiera de los dos lados y puede ser ordenado o abrupto.

Cierre ordenado (Four-Way Handshake)

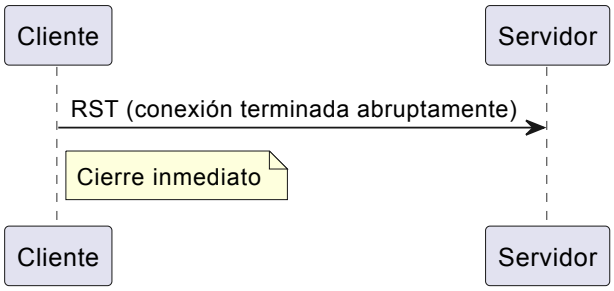
Cuando una conexión se cierra de forma ordenada, ambos lados desean informar al otro que no van a enviar más datos. Esto ocurre generalmente cuando un usuario ejecuta el comando FTP `QUIT` o cuando la aplicación termina normalmente.

- **FIN**: Un lado (generalmente el cliente) envía un segmento TCP con el flag FIN activado, indicando que ha terminado de enviar datos y que desea cerrar la conexión.
- **ACK**: El otro lado (el servidor) recibe el FIN y responde inmediatamente con un ACK, confirmando que ha entendido la solicitud de cierre. Sin embargo, el servidor podría tener más datos para enviar.
- **FIN**: Cuando el servidor también ha terminado de enviar datos, envía su propio segmento FIN al cliente.
- **ACK**: El cliente recibe el FIN del servidor y responde con un ACK final. A partir de este momento, la conexión está completamente cerrada.



Cierre abrupto (Two-Way o Reset)

En ocasiones, una conexión se cierra de forma abrupta sin pasar por el proceso ordenado. Esto puede ocurrir por varias razones: un cliente desconecta abruptamente la red, se agota el tiempo de espera, o hay un error grave. En estos casos, uno de los lados envía un segmento con el flag RST (reset) activado, lo que indica al otro lado que la conexión debe cerrarse inmediatamente sin intercambiar más mensajes.



En FTP, el four-way handshake ocurre cuando el usuario ejecuta normalmente QUIT, ordenando al cliente que cierre la conexión de forma ordenada. El two-way (RST) puede ocurrir si el cliente se desconecta abruptamente, la red se interrumpe, o hay un timeout.

USUARIOS FTP

Cuando hablamos de usuarios en FTP, en realidad estamos hablando de **identidades** que el servidor necesita reconocer para saber quién se está conectando y qué puede hacer. Es importante entender que el protocolo FTP en sí no define cómo gestionar usuarios —eso depende del sistema operativo y del servidor específico—. Lo que sí define FTP es **cómo una identidad se identifica ante el servidor** (mediante los comandos `USER` y `PASS`) y las restricciones que se pueden aplicar.

Para entender mejor los usuarios FTP, es útil pensar en dos dimensiones diferentes: **cómo se autentican** (es decir, cómo prueban quiénes son) y **qué representan realmente** en el sistema.

AUTENTICACIÓN: ACCESO ANÓNIMO VS. ACCESO AUTENTICADO

En términos de autenticación, FTP te ofrece dos opciones muy diferentes.

El usuario anónimo es la opción más abierta. Imagina que quieres compartir unos archivos de forma pública —por ejemplo, una versión de software o documentación que anyone puede acceder—. Para esto, configuras un usuario especial que se llama `anonymous` (o a veces `ftp`). Cualquiera puede conectarse con este usuario sin necesidad de credenciales reales. Si te pregunta por contraseña, puedes poner "anonymous" o incluso una dirección de correo. El servidor no verifica realmente quién eres; solo asume que eres el mismo usuario anónimo que los demás.

Por supuesto, el acceso anónimo tiene limitaciones de seguridad: no sabes realmente quién está accediendo, es difícil rastrear acciones individuales, y el nivel de control es muy limitado. Generalmente se restringe a lectura únicamente, aunque algunos servidores permiten un directorio especial para subidas (típicamente llamado `incoming`). Este modelo era común en los primeros tiempos de Internet para distribuir software o documentación pública, pero hoy en día se usa cada vez menos.

El usuario autenticado es lo opuesto: cada persona que se conecta usa su propio usuario y contraseña. Esto permite al servidor saber exactamente quién accede, cuándo lo hace y qué hace. Es mucho más seguro y trazable, y es el modelo que se usa en la mayoría de los entornos profesionales hoy en día. Cada usuario puede tener sus propios permisos específicos, límites de espacio, o incluso horarios de acceso.

REPRESENTACIÓN: USUARIOS DEL SISTEMA VS. USUARIOS VIRTUALES

Ahora bien, independientemente de cómo se autentique un usuario, **qué representa realmente ese usuario en tu servidor** es otra cuestión importante.

Un usuario del sistema es alguien que realmente existe como cuenta en tu sistema operativo. Cuando un usuario se conecta al FTP como "jose", esa cuenta también existe en Linux o Windows —aparece en el sistema de archivos, tiene un directorio de inicio, permisos específicos del SO, y puede usarse para otros servicios—. El servidor FTP simplemente delega la autenticación al sistema operativo. Esto es simple de entender y funciona bien en entornos pequeños o en configuraciones tradicionales.

Sin embargo, este modelo presenta un problema importante desde el punto de vista de la seguridad: si alguien compromete la cuenta FTP del usuario, también compromete potencialmente la cuenta del sistema operativo. Además, si tienes muchos usuarios FTP, terminas con muchas cuentas de sistema, lo que puede complicar la gestión y aumentar la superficie de ataque del servidor.

Un usuario virtual es diferente: existe **solo dentro del FTP**. No hay una cuenta de sistema correspondiente. El servidor FTP mantiene su propia base de datos de usuarios y contraseñas (que puede estar en un archivo, en una base de datos, o en un servidor LDAP), y gestiona internamente qué archivos puede ver cada uno y con qué permisos. Esto ofrece varias ventajas: mayor seguridad (un problema en FTP no afecta el resto del sistema), mejor escalabilidad (puedes tener cientos de usuarios virtuales sin saturar el sistema operativo), y mayor flexibilidad (puedes crear usuarios con políticas muy específicas sin que sean cuentas reales del SO).

Por eso, los usuarios virtuales son el modelo preferido en servidores de alojamiento (hosting), plataformas multiusuario, e infraestructuras empresariales modernas. Permiten un aislamiento efectivo entre los usuarios y un control mucho más granular.

GRUPOS Y PERMISOS

Tanto si usas usuarios del sistema como virtuales, muchos servidores FTP te permiten agrupar usuarios para aplicar políticas comunes. Por ejemplo, podrías crear un grupo *clientes* que comparta ciertos permisos sobre directorios particulares, sin tener que configurar cada usuario individualmente.

Los permisos en FTP determinan qué puede hacer cada usuario: si puede leer, escribir, eliminar archivos, qué carpetas ve, cuánto espacio puede usar, a cuántas conexiones simultáneas tiene derecho, o incluso desde qué direcciones IP puede conectarse.

MODOS DE CONEXIÓN FTP

El protocolo FTP define dos modos de operación distintos para establecer la conexión de datos entre cliente y servidor: el **modo activo** y el **modo pasivo**. Ambos modos mantienen el mismo comportamiento para el canal de control (puerto TCP 21), pero difieren significativamente en cómo se establece y gestiona el canal de datos, lo que implica diferentes implicaciones para la arquitectura de red, configuración de firewalls y compatibilidad con sistemas NAT.

La elección del modo de conexión tiene repercusiones directas en:

- La arquitectura de las conexiones TCP y la dirección en que se inician
- La configuración requerida en sistemas de filtrado de tráfico (firewalls)
- La compatibilidad con redes que emplean traducción de direcciones (NAT)
- El diagnóstico y resolución de problemas de conectividad

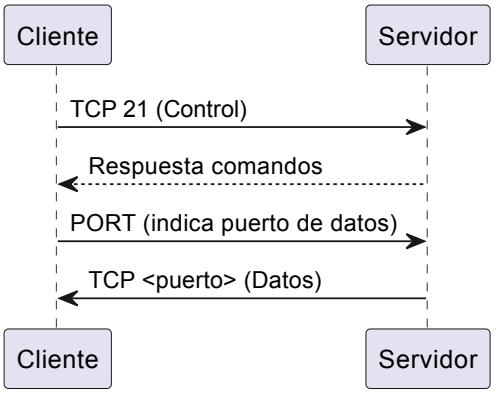
MODO ACTIVO

En el **modo activo**, el cliente desempeña un papel pasivo durante la transferencia de datos, permitiendo que el servidor inicie la conexión hacia el cliente. Este modo constituye la implementación original descrita en el RFC 959 y sigue el siguiente procedimiento:

1. El cliente establece una **conexión TCP hacia el puerto 21** del servidor (canal de control), mediante el que se mantiene la sesión FTP.
2. Cuando el cliente requiere transferir datos, envía el comando `PORT` especificando su dirección IP y el número de puerto en el que estará escuchando conexiones entrantes.
3. El **servidor responde confirmando la recepción del comando** `PORT` y procede a iniciar una conexión TCP desde su puerto 20 (o un puerto dinámico configurado) hacia el puerto indicado por el cliente.
4. Una vez establecida la conexión de datos, se realiza la transferencia de información (descarga o carga de archivos/directorios).

Implicaciones técnicas:

- El cliente debe estar configurado como servidor de puertos (escuchando conexiones entrantes).
- El servidor inicia la conexión de datos de forma activa hacia el cliente.
- La implementación requiere que los firewalls del lado del cliente permitan conexiones TCP entrantes en puertos dinámicos.
- La presencia de dispositivos NAT complica significativamente esta modalidad, ya que la dirección IP comunicada al servidor puede no ser válida desde su perspectiva de red.



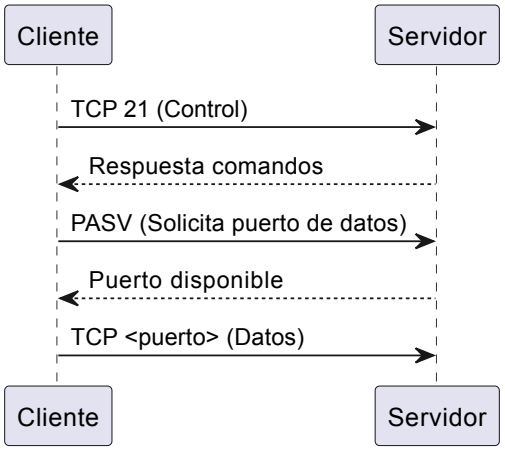
MODO PASIVO

En el **modo pasivo**, el cliente actúa como iniciador en ambas conexiones, permitiendo que el servidor permanezca pasivo durante el establecimiento de la conexión de datos. Este modo fue introducido posteriormente como mejora para resolver las limitaciones del modo activo en ciertos contextos de red. El procedimiento es el siguiente:

1. El cliente establece una **conexión TCP hacia el puerto 21** del servidor (canal de control), a través del cual se intercambiarán comandos FTP.
2. Cuando el cliente requiere realizar una operación de transferencia de datos, envía el comando `PASV` (Passive Mode) al servidor.
3. El **servidor responde con el comando** `227 Entering Passive Mode`, indicando su dirección IP y el número de puerto aleatorio que ha abierto para la transferencia de datos.
4. El cliente, recibida esta información, **inicia una nueva conexión TCP hacia el puerto indicado por el servidor** para establecer el canal de datos.
5. Se realiza la transferencia de información mediante esta conexión recién establecida.

Implicaciones técnicas:

- El cliente inicia ambas conexiones (control y datos), actuando como cliente en ambos casos.
- El servidor adopta un rol pasivo en el establecimiento de la conexión de datos, limitándose a escuchar en un puerto asignado dinámicamente.
- Esta modalidad elimina la necesidad de que los firewalls del cliente permitan conexiones entrantes, ya que todas las iniciativas de conexión proceden del cliente.
- La compatibilidad con NAT es sustancialmente mejorada, puesto que el cliente obtiene la información del servidor desde su perspectiva de red.
- Se ha convertido en el modo predominante en implementaciones modernas debido a su mayor compatibilidad con infraestructuras de red contemporáneas.



COMANDOS FTP

Los comandos FTP son instrucciones enviadas por el cliente al servidor a través del canal de control para realizar diversas operaciones relacionadas con la gestión y transferencia de archivos. Estos comandos son en texto plano y siguen una sintaxis específica definida por el protocolo FTP. Se les conoce comúnmente como **comandos en crudo** (*raw commands*) para diferenciarlos de las órdenes que los usuarios pueden introducir en los clientes FTP.

El servidor responde a cada comando con un código numérico y un mensaje que indica el resultado de la operación solicitada.

REFERENCIA DE COMANDOS

La siguiente tabla recoge los principales comandos definidos por el protocolo FTP y su función real dentro de una sesión:

Comando	**Tipo**	**Descripción**
USER <usuario>	Autenticación	Indica el nombre de usuario con el que se desea iniciar sesión.
PASS <contraseña>	Autenticación	Envía la contraseña asociada al usuario.
QUIT	Control	Finaliza la sesión FTP y cierra la conexión de control.
PWD	Navegación	Muestra el directorio de trabajo actual en el servidor.
CWD	Navegación	Cambia el directorio de trabajo.
LIST	Transferencia	Solicita un listado detallado de archivos y directorios.
NLST	Transferencia	Solicita un listado simple de nombres de archivos.
RETR <archivo>	Transferencia	Descarga un archivo desde el servidor al cliente.
STOR <archivo>	Transferencia	Sube un archivo desde el cliente al servidor.
DELE <archivo>	Gestión	Elimina un archivo del servidor.
TYPE <tipo>	Configuración	Establece el tipo de transferencia (ASCII o binaria).
PORT <ip,puerto>	Modo activo	Indica al servidor el puerto del cliente para la conexión de datos.

Comando	Tipo	Descripción
PASV	Modo pasivo	Solicita al servidor que abra un puerto para la conexión de datos.
SYST	Información	Solicita información sobre el sistema operativo del servidor.
NOOP	Control	No realiza ninguna acción; se usa para mantener viva la conexión.

Estos comandos son independientes del sistema operativo y forman parte del estándar FTP.

CÓDIGOS DE RESPUESTA

Las respuestas del servidor FTP son mensajes enviados al cliente en respuesta a los comandos recibidos. Cada respuesta consta de un **código numérico de tres dígitos** seguido de un mensaje descriptivo. El código indica el resultado de la operación solicitada y se clasifica en varias categorías según su valor:

Categoría	Significado
1xx	Respuesta preliminar (acción en curso)
2xx	Acción completada con éxito
3xx	Se necesita información adicional
4xx	Error temporal
5xx	Error permanente

La siguiente tabla recoge algunos de los códigos de respuesta más habituales en una sesión FTP:

Código	Significado
220	Servicio FTP listo
331	Usuario correcto, se necesita contraseña
230	Autenticación completada
150	Se va a abrir la conexión de datos
226	Transferencia completada
530	Autenticación fallida
550	Archivo no disponible o acceso denegado

NEGOCIACIÓN DE LA CONEXIÓN DE DATOS

La transferencia de datos en FTP requiere que cliente y servidor negocien cómo se establecerá la conexión de datos. Existen dos comandos específicos para esta negociación: `PORT` (utilizado en modo activo) y `PASV` (utilizado en modo pasivo). Ambos comandos permiten que el cliente comunique al servidor (o que el servidor comunique al cliente) la información necesaria para establecer el canal de datos de forma correcta.

EL COMANDO PORT

El comando `PORT` es utilizado en el **modo activo** de FTP para indicar al servidor la dirección IP y el puerto TCP en el que el cliente está escuchando para la conexión de datos. La sintaxis del comando es la siguiente:

```
PORT h1,h2,h3,h4,p1,p2
```

Donde:

- `h1,h2,h3,h4`: representan los cuatro octetos de la dirección IP del cliente.
- `p1,p2`: representan el puerto TCP en el que el cliente está escuchando, calculado como `p1*256 + p2`.

Comando PORT

En el siguiente ejemplo se muestra un comando `PORT` típico:

```
PORT 192,168,1,50,8,69
```

En este ejemplo, la dirección IP del cliente es `192.168.1.50` y el puerto es `2077` (8*256 + 69). Tras recibir este comando, el servidor FTP intentará establecer una conexión TCP al puerto `2077` de la dirección `192.168.1.50`.

EL COMANDO PASV

El comando `PASV` es utilizado en el **modo pasivo** de FTP para solicitar al servidor que abra un puerto TCP para la conexión de datos. Cuando el cliente envía el comando `PASV`, el servidor responde con un mensaje que incluye la dirección IP y el puerto que el cliente debe usar para establecer la conexión de datos. La respuesta del servidor tiene la siguiente estructura:

```
227 Entering Passive Mode (h1,h2,h3,h4,p1,p2)
```

Donde:

- `h1,h2,h3,h4`: representan los cuatro octetos de la dirección IP del servidor.
- `p1,p2`: representan el puerto TCP que el servidor ha abierto para la conexión de datos, calculado como `p1*256 + p2`.

Comando PASV

En el siguiente ejemplo se muestra un comando `PASV` típico:

```
PASV
```

En este ejemplo, el servidor responde con un mensaje como `227 Entering Passive Mode (192,168,1,50,8,69)`, indicando que el servidor está escuchando en la dirección `192.168.1.50` y el puerto `2077` (8*256 + 69). Tras recibir la respuesta del servidor, el cliente FTP intentará establecer una conexión TCP al puerto `2077` de la dirección `192.168.1.50`.

ANÁLISIS DEL PROTOCOLO FTP

Los comandos FTP se envían a través del canal de control utilizando una conexión TCP establecida entre el cliente y el servidor. Cada comando debe ser seguido por una secuencia de caracteres de nueva línea (CRLF) para indicar el final del comando. Herramientas como `telnet` y/o netcat (`nc`) pueden utilizarse para enviar comandos FTP manualmente y observar las respuestas del servidor.

Sesión FTP en modo activo

El siguiente ejemplo ilustra una sesión FTP en modo activo utilizando `telnet` para la conexión de control y `netcat` para la conexión de datos. Nosuestra cómo un cliente FTP se conecta a un servidor (`192.168.10.251`), autentica al usuario (`jose` con contraseña `pepe`), solicita un listado de archivos (`LIST`) y cierra la sesión.

Conexión de control con Telnet

Cliente-LX:~# telnet 192.168.10.251 21
Connected to 192.168.10.251
220-FileZilla Server 0.9.60 beta
220-written by Tim Kosse (`tim.kosse[at]filezilla-project.org`)
220 Please visit `filezilla-project.org`
USER jose
331 Password required for jose
PASS pepe
230 Logged on
PORT 192,168,10,66,200,1
200 Port command successful
LIST
150 Opening data channel for directory listing of "/"
226 Successfully transferred "/"
QUIT
221 Goodbye
Connection closed by foreign host
Cliente-LX:~#

Antes de ejecutar el comando `LIST` , es necesario abrir una conexión de datos utilizando `netcat` en el puerto especificado en el comando `PORT` (en este caso, puerto `51201` , que se calcula como `200*256 + 1`).

Conexión de datos con netcat

Cliente-LX:~# nc -l -p 51201
X1-L1:*# nc -l -p 51201
-rw-r-xr-x 1 ftp ftp 136774728 Jan 12 09:59 Autofirma_64_v1_9_installer.exe
-rw-r--r-- 1 ftp ftp 89 Jan 17 23:44 datos.txt
-rw-r-xr-x 1 ftp ftp 2241216 Jan 17 22:18 filezilla-server-0-9-60-2.exe

Sesión FTP en modo pasivo

El siguiente ejemplo ilustra una sesión FTP en modo pasivo utilizando `telnet` para la conexión de control y `telnet` nuevamente para la conexión de datos. Muestra cómo un cliente FTP se conecta a un servidor (`192.168.10.251`), autentica al usuario (`jose` con contraseña `pepe`), descarga un archivo (`RETR datos.txt`) y cierra la sesión.

Conexión de control con Telnet

Cliente-LX:~# telnet 192.168.10.251 21
Connected to 192.168.10.251
220-FileZilla Server 0.9.60 beta
220-written by Tim Kosse (`tim.kosse@filezilla-project.org`)
220 Please visit `https://filezilla-project.org/`
USER jose
331 Password required for jose
PASS pepe
230 Logged on
PASV
227 Entering Passive Mode (192,168,10,251,230,126)
RETR datos.txt
150 Opening data channel for file download from server of "/datos.txt"
226 Successfully transferred "/datos.txt"
QUIT
221 Goodbye
Connection closed by foreign host
Cliente-LX:~#

Antes de ejecutar el comando `RETR` , es necesario abrir una conexión de datos utilizando `telnet` en la dirección IP y puerto especificados en la respuesta al comando `PASV` (en este caso, dirección `192.168.10.251` y puerto `58878` , que se calcula como `230*256 + 126`).

Conexión de datos con Telnet

Cliente-LX:~# telnet 192.168.10.251 59006
Connected to 192.168.10.251
Fichero de texto contiene 3 líneas
Esta es la segunda línea
Esta es la tercera línea
Connection closed by foreign host
Cliente-LX:~#

SEGURIDAD DEL PROTOCOLO FTP

Aunque FTP fue fundamental en la historia de Internet, su diseño presenta vulnerabilidades significativas que lo hacen inadecuado para entornos donde la seguridad es una preocupación. Estas debilidades no son defectos accidentales, sino consecuencia de que el protocolo fue desarrollado en una época en la que la seguridad de la información no era una prioridad, y las redes eran entornos controlados con un número reducido de usuarios de confianza.

VULNERABILIDADES PRINCIPALES

Transmisión de credenciales sin protección
El aspecto más crítico es que tanto el usuario como la contraseña se envían en texto plano durante la autenticación. Cualquiera que capture el tráfico de red en la ruta entre cliente y servidor puede leer estas credenciales directamente. Esto es particularmente peligroso cuando la conexión atraviesa redes públicas o no confiables, donde técnicas de captura de paquetes (sniffing) son relativamente sencillas.

Datos transferidos sin cifrado
Los archivos se transmiten sin protección, lo que significa que cualquiera con acceso a la red puede interceptar y leer el contenido de los archivos transferidos. Esto viola la confidencialidad y puede exponer información sensible.

Múltiples puertos dinámicos
El uso de puertos dinámicos para la conexión de datos complica significativamente la configuración de firewalls. No es posible simplemente abrir un puerto específico; los firewalls deben permitir un rango amplio de puertos, aumentando la superficie de ataque disponible.

Vulnerabilidad a ataques de fuerza bruta
FTP no implementa mecanismos nativos para limitar intentos de conexión fallidos. Los atacantes pueden enviar múltiples intentos de autenticación sin experimentar penalizaciones significativas, facilitando ataques de fuerza bruta contra contraseñas débiles.

Falta de mecanismos de integridad
No hay forma de verificar que los datos transferidos no han sido modificados durante la transmisión. Un atacante podría interceptar y alterar archivos sin que el cliente o el servidor lo detecten.

ALTERNATIVAS Y MEJORAS

Frente a estos problemas, la comunidad de seguridad ha desarrollado varias soluciones:

FTPS (FTP Secure)
Añade una capa de cifrado TLS/SSL al protocolo FTP tradicional. Esto protege tanto las credenciales como los datos transferidos, abordando las vulnerabilidades más críticas. Sin embargo, mantiene la complejidad del protocolo original (múltiples conexiones, puertos dinámicos).

SFTP (SSH File Transfer Protocol)
Basado completamente en SSH, ofrece una alternativa más moderna y segura. SFTP opera sobre una única conexión encriptada, elimina la complejidad de los puertos dinámicos, y es considerado significativamente más seguro. Es la opción recomendada para nuevas implementaciones.

Configuración y restricciones del servidor
Incluso con FTP tradicional, es posible mejorar la seguridad mediante políticas administrativas: desactivar el

acceso anónimo si no es absolutamente necesario, mantener contraseñas robustas, limitar el acceso por dirección IP, y usar firewalls configurados adecuadamente. Sin embargo, estas medidas son limitadas frente a los problemas inherentes del protocolo.

Auditoría y monitoreo

Registrar todas las actividades FTP permite detectar patrones sospechosos y responder a incidentes. Los logs pueden revelar intentos de acceso fallidos, transferencias inusuales, o comportamientos anómicos.