

VSFTPD

vsFTPd

Laboratorio de Software de Comunicaciones

Área de Ingeniería Telemática

©LSC 2004/2005

ÍNDICE

ÍNDICE	2
1. Objetivos y Alcance	4
1.1. Introducción.....	4
1.2. Motivación y Funcionalidad del Servicio.....	5
1.3. Documentación Bibliográfica.....	8
2. Base Teórica	8
2.1. Descripción del Servicio y Conceptos Implicados	8
2.2. Análisis de Protocolos	13
2.2.1. Funcionamiento y Estructura del Protocolo.....	13
2.2.2. Uso del Protocolo en el Servicio.....	19
3. Solución Adoptada para Ofrecer el Servicio	22
3.1. Soluciones Existentes en el Mercado	22
3.2. Referencias y Características de la Solución Adoptada	24
3.3. Equipamiento Necesario.....	25
4. Parámetros de Configuración del Servidor	25
5. Proceso de Instalación/Administración del Servidor.....	31
5.1. Obtención del Código.....	31
5.2. Instalación del Servidor.....	31
5.3. Configuración del Servidor	34
5.4. Puesta en Funcionamiento del Servicio.....	36
5.5. Administración, Monitorización y Prueba del Funcionamiento.....	37
6. Análisis del Intercambio de Mensajes por la Red.....	43
7. Interfaz Gráfica de Gestión	47

8.	Ampliaciones/Mejoras del Servicio	47
9.	Incidencias y Principales Problemas Detectados	47
10.	Resumen y Conclusiones	49

1. Objetivos y Alcance

1.1. Introducción

Vsftpd es un servidor para sistemas Unix, incluyendo Linux. La licencia de *vsftpd* la tiene GPL aunque es totalmente gratuito. *Vsftpd* es un servidor seguro, extremadamente rápido y estable, nosotros podremos ver más adelante claras evidencias de estas tres afirmaciones, también veremos muchas empresas importantes que están satisfechas con el uso de este servidor seguro de FTP.

A pesar de ser pequeño para tener el propósito de velocidad y de seguridad, tiene muchos mecanismos complicados de FTP. Los siguientes mecanismos no es de ninguna manera una lista exclusiva, *vsftpd* maneja lo siguiente:

- Configuración de IP virtual
- Usuarios Virtuales
- Operación *xinetd* o funcionamiento *Standalone*
- Gran variedad de configuración del usuario
- Ancho de banda limitado
- Configuración de fuentes IP
- Límites de fuentes IP
- Soporta IPv6
- Encriptación soportada a través de la integración SSL
- etc....

Ciertamente, *vsftpd* fue diseñado e implementado desde un principio pensando en la seguridad. Para ello realiza cosas como:

1. Arregla el diseño defectuoso presente en muchas instalaciones de *wu-ftpd*, *pro-ftpd* y *bsd-ftpd* debido a que no permite el acceso al peligroso usuario de la raíz (*root*).
2. Hace uso de instalaciones seguras y potentes como las *capacidades* y *chroot*.
3. Emplea técnicas de codificación segura solucionando el problema del desbordamiento del buffer.

1.2. Motivación y Funcionalidad del Servicio


Si tu principal requisito para un servidor FTP es uno de los siguientes, entonces *vsftpd* es probablemente el servidor FTP que más te conviene:

- Seguridad.
- Rápido funcionamiento.
- Estable.

La única razón por la que deberías preferir un servidor FTP diferente a *vsftpd* es si tú realmente necesita un servidor FTP con una configuración más extensa. Habiendo dicho esto, hacer notar que *vsftpd* abastece la gran mayoría de los casos de uso. Aún así *vsftpd* parece echar de menos una característica, a menudo es satisfecho por un componente externo como un PAM o *xinetd/tcp_wrappers*. En este vistazo, *vsftpd* está siendo un componente modular correcto en el interior de UNIX.

Finalmente, si consideras que pasarse a *vsftpd* supone algún sacrificio en las características de tu habitual servidor FTP, seguramente la seguridad, la rapidez y la estabilidad de *vsftpd* se impondrá si es lo que esperas.

Veamos que dicen algunas críticas en Internet sobre *vsftpd*:

-  El equipo SAC de SANS recomienda *vsftpd* como el servidor FTP seguro preferido: “*For those of you looking for a secure FTP daemon alternative, the SAC team recommends vsftpd*” (Para aquellos que buscan una alternativa de un servidor FTP seguro, el equipo SAC recomienda *vsftpd*)



- IBM recomienda *vsftpd* en sus artículos “*Securing Linux Servers for Service Providers*”. Esto es la parte superior de una sección que se titula “Recommended FTP servers”.



- Red Hat elogia el funcionamiento y la dimension de *vsftpd* en una revista: “*Individual servers handled more than 2,500 concurrent downloads*”... “*The other change was to use a very lightweight FTP daemon, vsftpd, designed for the demands place on a Server under this level of load*”.
- La siguiente lista de aproximadamente del Junio de 2004 indica los sitios en Internet en los cuales se utilizan *vsftpd*.

<ftp.redhat.com>

<ftp.suse.com>

<ftp.freebsd.org>

<ftp.gnu.org>

<ftp.kernel.org>

<Rpmfind.net>

<ftp-stud.fht-esslingen.de>

<Gd.tuwien.ac.at>

<ftp.engardelinux.org>

<ftp.sunsite.org.uk>

<ftp.redhat.com>

<ftp.suse.com>

<ftp.debian.org>

<ftp.openbsd.org>

<ftp.gnome.org>

<ftp.kde.org>

<ftp.linux.org.uk>

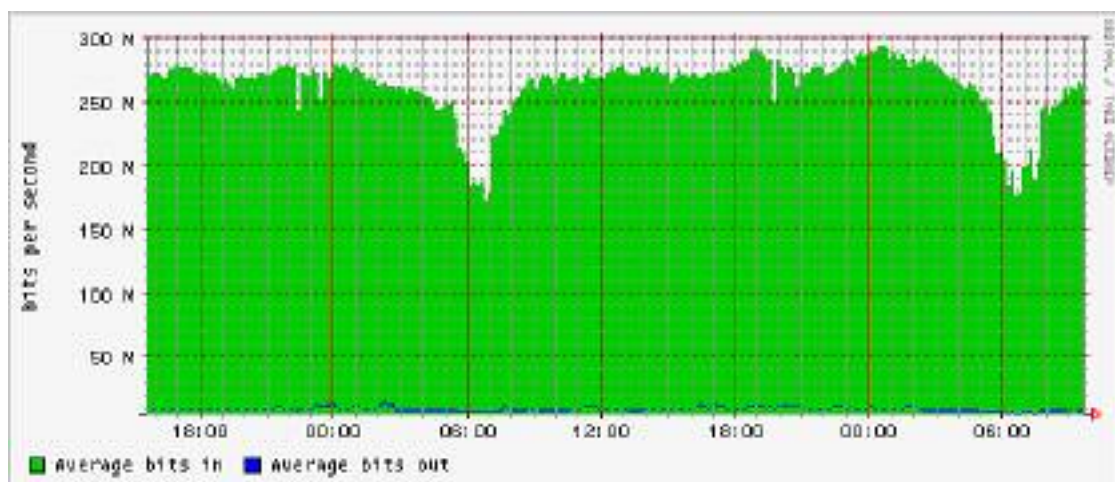
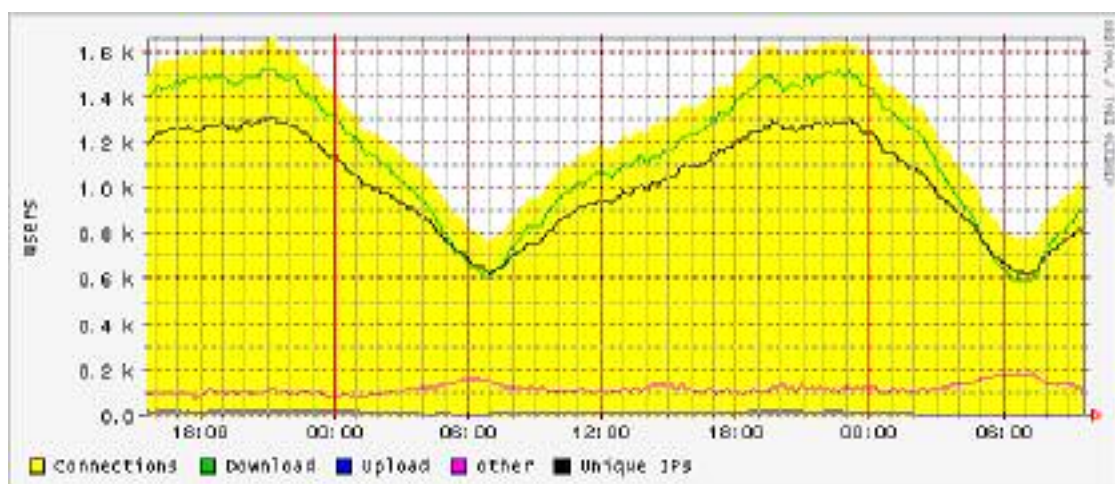
<ftp.gimp.org>

<ftp.sunet.se>

<ftp.ximian.com>

- En gráficas estadísticas del uso de la red se observa que *vsftpd* es el doble de rápido que BSD-*ftpd* (por tanto no es un servidor lento como en el caso de *wu-ftpd*).

- Se ha probado midiendo comparativamente *vsftpd* sobre localhost que transmite a 70 MBps lo cual esta por encima del impresionante TUX que transmite a 55 MBps.
- Alguien midiendo en los sistemas de redes de Linux sobre redes ethernet gigabit usando *vsftpd*, y resulta que *vsftpd* puntúa hasta 86Mbps
- Refiriéndose al uso de Red Hat de *vsftpd*, Alan Cox en su diario dice “*finally we have a scalable ftpd for Linux*” (Finalmente encontramos un ftpd escalable para Linux)
- Aquí hay un par de gráficas enviada por un usuario satisfecho, que dirige un sitio de bastante tráfico de Internet mediante *vsftpd*. En 24 horas, *vsftpd* ha servido 2.6TB (correcto, *Terabytes*) contando hasta con 1500 usuarios en paralelo. Todo esto es una única máquina servidora.



1.3. Documentación Bibliográfica

- <http://vsftpd.beasts.org/>
- “*Protocolos de INTERNET*”©, editorial: Ra-Ma, autores: Ángel López y Alejandro Novo.
- Apuntes no oficiales de Software de Comunicación.
- Prácticas guiadas de Laboratorio de Software de Comunicación.
- Artículo del www.linuxfocus.net: “*An introduction to the very secure FTP daemon*” © , autores: Mario Knopf.
- www.mundopc.net/cursos/ftp/9.php
- <http://lordcrow.blogspot.com/>
- www.ulfix.com
- <http://www.vsftpdrocks.org>

2. Base Teórica

A continuación se describe de forma teórica el servidor *vsftpd*.

2.1. Descripción del Servicio y Conceptos Implicados

Vamos a describirlo en partes como:

DISEÑO

Vamos a explicar el diseño y decisiones que hay tomadas detrás de *vsftpd*. En un mundo ideal, de buenos y seguros códigos sin ningún tipo error, un diseño seguro del código no es necesario. Desafortunadamente esto no es un mundo ideal, y los códigos tienen bastantes errores, hasta el código más seguro tiene errores, siendo por tanto muy importante la revisión del código y el hecho de llevar un cierto camino en la eliminación de errores de códigos que puedan aparecer posteriormente. Incluso así, nosotros no tenemos plena garantía de que una revisión del código detectará todos los errores posibles.

Por tanto un diseño seguro aumenta la seguridad de que otras personas malintencionadas no descubran los defectos de nuestro programa o por lo menos lo tenga más complicado de detectar, por tanto hay que seguir los pasos para minimizar el impacto que estos fallos de programación pueden ocasionar en la seguridad de tu programa. Un ejemplo de algo que podemos hacer es aplicar “*el principio de la mínima prioridad*”, el cual asegura que todas las partes de un programa se ejecutan con la mínima prioridad que requiere y no más.

Ejemplo.- (Diseño inseguro) Ejemplos de diseño inseguro pueden ser encontrados en muchos otros *ftpd*'s. Esto es una de las razones por la que *vsftpd* fue diseñado. Nosotros criticamos a *wu-ftpd* como un ejemplo concreto de diseño inseguro. Si yo accedo a *wu-ftpd* como un cliente anónimo se ejecuta un proceso con mi nombre y entraré en mi sesión ftp. Desafortunadamente, este proceso es ejecutado con todo los privilegios del *root* en la máquina remota.

Dos ejemplos concretos más son el fallo del formato de cadena de *wu-ftpd*, y un desbordamiento del buffer ocasionado por una ruta larga. Incluso *ftpd-bsd* tiene fallo del formato de cadena el cual lo dirige al procesamiento del *root* remoto de la máquina afectada.

Vsftpd está diseñado para ejecutarse bajo UNIX o sistemas relacionados como puede ser Linux, ya que su diseño de seguridad es construido a partir de las ventajas ofrecidas por UNIX. Idealmente, Unix debería tener un modelo de seguridad propio el cual debería ofrecer un buen control de acceso a todos los que interactúan con el sistema (ficheros, red, etc.), pero eso no ocurre así desafortunadamente. Pero por otra parte también es cierto que ofrece algunas herramientas y desconocidas ventajas que nos ayuda a implementar “*el principio de la mínima prioridad*”, Esas utilidades son las siguientes:

- Potente comunicación entre-procesos.

En UNIX, los procesos tienen una frontera muy delimitada. Diferentes procesos con privilegios acreditados se lo puede asignar a distinto proceso, los cuales no son capaces de interferir con otros procesos. Eso es una ventaja básica de UNIX. Tiene sentido usar esta ventaja para separar totalmente un programa en partes el cual no necesita tener el máximo privilegio en todas sus partes, sino todo lo contrario, lo normal es que tenga el mínimo privilegio que requiere.

Las partes privilegiadas y no privilegiadas de un programa se comunican a través de muchos mecanismos IPC (*inter-process communication*) de UNIX, quizás un *socketpair* o *IPC* (el anterior es interesante por que UNIX permite acceder a archivos de control bajo un *socket*.)

El proceso de mínimo privilegio realiza “*el principio de desconfianza*”, en el cual los seguros filtros del sistema preguntan a los procesos sin privilegio que quiere hacer, y si esto implica poner en peligro no se le permitirá el acceso.

- *chroot()*

chroot() es una herramienta muy frecuentemente ignorada pero no por eso útil. Puede ser usada de forma muy efectiva como una herramienta de reducción del daño provocado por alguien malintencionado. Supongamos un proceso remoto que puede generar un cierto riesgo para nuestro ordenador, el cual no se está ejecutando como *root*, pero tampoco usa *chroot ()*. Ahora miremos que es lo que puede hacer el supuesto “intruso”. Entre lo peor está la acceso ilegal de archivos legibles, y también intentar ejecutar al descubierto algún programa ejecutable para intentar aumentar privilegios.

Ahora supongamos el mismo proceso peligroso pero con un *chroot ()* a un directorio vacío. La opción del intruso para hacer cosas perjudiciales está substancialmente reducida, ya que su directorio raíz es distinto al del servidor.

La no utilización de *chroot ()* no es un camino idóneo a seguir, a no ser que seamos experto, realmente es eso con lo que nosotros debemos trabajar.

- *Capabilities* (Linux 2.2+)

A igual que *chroot ()*, las *capabilities* es una herramienta esencial para la reducción de daño. Está menos extendido que otras ventajas de Unix. No obstante las *capabilities* garantizan la seguridad, por eso Linux los utiliza, y por ello también son usados en *vsftpd* por que eso es una cosa primordial en la plataforma de desarrollo Linux.

Las *capabilities* separan todos los potentes privilegios *root* a privilegios ortogonales. Algunas de las *capabilities* representan privilegios los cuales son a menudo la base del requisito de un programa que se iba a ejecutar con todos los privilegios *root*.

Por usar *capabilities* nos asegura a nosotros no solo tener los privilegios que necesitamos, sino que también limitaremos el potencial daño de los agujeros de seguridad.

Veamos ahora la presentación del diseño seguro de *vsftpd* el cual usa las ventajas de Unix con buen efecto. Las decisiones de diseño tomadas son por ejemplo como las siguientes:

1º Todo análisis y actuación sobre datos potencialmente maliciosos de una red remota es hecho un proceso ejecutado como un cliente sin privilegios. Además, este proceso se ejecuta como un *chroot ()* encerrado, asegurando el área de archivos ftp que es accesible por éste.

2º Cualquier operación privilegiada es controlada por un proceso padre privilegiado. El código para este proceso padre privilegiado es lo mas pequeño posible por motivo de seguridad y se encarga de que el proceso hijo al cual se le está otorgando privilegios no haga uso malintencionado de este.

3º Este mismo proceso padre privilegiado recibe peticiones de un proceso hijo no-privilegiado sobre un *socket*. Todas las peticiones son tomadas con desconfianza. Aquí hay algunos ejemplos de peticiones:

- Petición *Login*. El proceso hijo envía nombre de usuario y *passwd*. Solo si los datos son correctos, el proceso padre privilegiado hace un nuevo proceso hijo con los credenciales de cliente apropiados.
- Petición *chown()*. El proceso hijo puede pedir un archivo subido recientemente el cual se consigue con *chown'ed()* a *root* por motivo de seguridad. El proceso padre es seguro y solo permite *chown()* a *root*, y solo de archivos propios para el cliente ftp.
- Petición de *socket* privilegiado. El protocolo ftp dice que debemos soportar conexiones de datos por el puerto 20, esto requiere privilegios. El proceso padre privilegiado crea un *socket* privilegiado y pasa esto a un proceso hijo sobre el *socket*.

4º Este mismo proceso padre privilegiado hace uso de las *capabilities* y *chroot()*, para ejecutar con el mínimo privilegio requerido. Después del *login*, dependiendo de que opciones se hayan seleccionado, el proceso padre privilegiado dinámicamente calcula que privilegios es requerido. En algunos casos, aunque no tenga importancia el hecho de no tener privilegios

5º *vsftpd-2.x.x* soporta SSL y TLS usando *OpenSSL*. Todo análisis de protocolo *OpenSSL* esta funcionando en un *chroot()* encerrado, ejecutándose bajo un cliente sin privilegios, esto es actualmente muy complicado de hacer, pero *vsftpd* arregla eso en el nombre, siendo por tanto muy seguro. Hasta el momento desconozco de otros

servidores FTP el cual soporta ambos SSL/TLS y separación de privilegios, y que se consiga de forma correcta.

TRUST (CONFIANZA)

A continuación se describe en que confía el código *vsftpd*, y en que no confía, y las razones detrás de cualquier decisiones de confianza. También veremos la importancia de relaciones de confianza y de responsabilidad.

Imagine un trozo de código seguro bien redactado. Ahora imagine que ese trozo de código delega una tarea a un programa externo. Ahora bien, si este programa externo está descuidadamente codificado y es inseguro, entonces lo hemos hecho desaprovechando un montón de esfuerzo haciendo nuestro programa original seguro. Nuestra errónea confianza en el programa externo afectado implica que nosotros tenemos un agujero de seguridad, aunque nosotros estemos seguros de nuestro código.

Supóngase que nuestro seguro programa llama alguna función complicada de la biblioteca la cual tiene un agujero de seguridad. Vamos a poner algunos ejemplos concretos sobre dos consideraciones similares.

1) el ayudante externo `/bin/ls` (comando `ls`)

Una operación muy común solicitada de los servidores de FTP es proveer la escucha de un directorio, es decir, ver el contenido de una carpeta. Se tiene la tentación de rehusar `/bin/ls` como un proceso hijo para evitar tener que reescribir una carga de código para el manejo de la escucha de directorios.

Para usar un comando externo `/bin/ls`, nosotros deberíamos mirar la seguridad de nuestro servidor FTP con los códigos `/bin/ls`. Es seguro no infravalorar la cantidad de caminos codificados en `/bin/ls` el cual son explorables por un cliente remoto malicioso. GNU `/bin/ls` tiene muchas opciones. Algunas de estas opciones son complejas tales como `-l` o las variables opciones de formato. Pudiendo haber un simple defecto de codificación en el manejo de una de esas opciones, y por tanto nuestra seguridad FTP se ve puesta en problema.

Por usar un `/bin/ls`, también heredas el riesgo de algunos peligros de las complejas Apis que eso usa. Por ejemplo, llamada a funciones `fnmatch()` o `glob()`, las cuales pueden dar a los clientes maliciosos datos controlado por el proceso padre.

Como conclusión decir que *vsftpd* no tiene intención de usar un programa */bin/lis* externo a causa de los riesgos que eso trae. La solución que se ha empleado es escribir una implementación minimizada interna de un generador de escucha */bin/lis*, aunque eso es muy difícil de implementar.

2) Librerías *APIS* complejas

Vsftpd es muy cuidadoso a la hora de evitar usar llamadas a librerías las cuales son potencialmente peligrosas. Típicamente deberíamos clasificar llamadas como peligrosas si ellos interactúan con la red de forma sencilla y conocida.

Algunos ejemplos son (*vsftpd* evita usar algunos de los siguientes):

1º *fnmatch()*. Este es una función de la librería matemática. El peligro viene del hecho que los clientes suplica el patrón *glob* –“*ls *.mp3*” debería ser un simple ejemplo. Además, el patrón *glob* es complejo e implica mucho manejo de cadenas.

2º *gethostyaddr()*. Esto es una *libc* utilizada para resolver una dirección IP de un *hostname*. Desafortunadamente, hacer esto es muy complicado. Cuando tu llamas *gethosbyaddr()*, mucho trabajo está al descubierto. Esto usualmente implica hacer una red llama a salir al servidor DNS, y, peligrosamente, analizar la respuesta.

Por claridad (y claridad es una parte importante de la seguridad), toda *APIs* externa usada por *vsftpd* son encapsulada dentro dos archivos del tipo *Interaction-System*, llamados “*sysutil.c*”, y “*sysdeputil.c*” (para las llamadas dependientes del sistemas). Esto proporciona un punto de revisión conveniente para determinar en que llamadas confía *vsftpd*.

vsftpd-2.x.x introduce soporte SSL/TLS usando *OpenSSL*. *OpenSSL* es una cantidad de código masivo el cual es esencial para analizar el complejo protocolo bajo el control lleno de clientes maliciosos. SSL/TLS está deshabilitado por defecto.

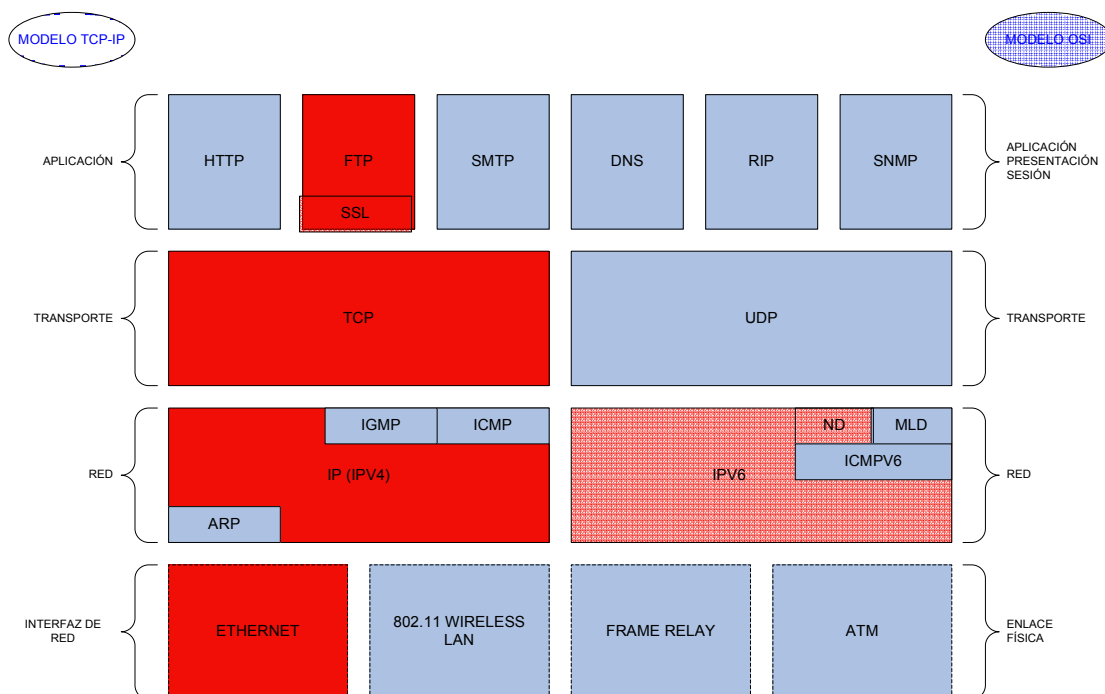
2.2. *Análisis de Protocolos*

Los dos protocolos más importantes del cual hace uso *vsftpd* son:

- Protocolo FTP, para la transferencia de datos.
- Protocolo SSL, para la seguridad de dichas transferencias.

PROTOCOLO FTP

En la figura se representa el protocolo FTP en el modelo OSI y TCP/IP.

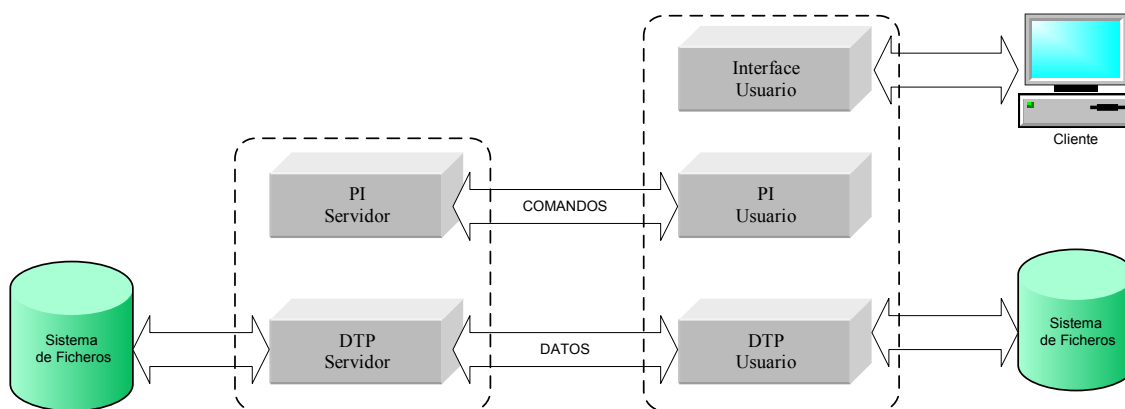


El servidor *vsftpd* se comunica con el cliente a través del protocolo FTP. El sistema FTP, que se define para TCP/IP, se basa sobre el servicio de conexiones extremo-extremo fiable TCP. Este sistema ha generado un gran volumen de tráfico de información en la red durante los últimos años y aunque en principio no estaba diseñado para usuarios, permite un acceso interactivo a través de la invocación mediante programas. FTP permite la autenticación además de la posibilidad de especificar el formato de los ficheros.

Veamos el modelo de trabajo del proceso FTP. El servidor FTP permite el acceso concurrente, es decir, varias conexiones simultáneas mediante TCP. En el servidor tendremos un proceso, denominado maestro, que está esperando conexiones por parte de los distintos clientes. Cada vez que se conecta un cliente, el proceso maestro crea un proceso esclavo que se encargará de la interacción con el cliente. Este proceso esclavo va a manejar una conexión de control, que permanece durante toda la transferencia de ficheros, y por otro lado una conexión de transferencia de datos, que se crea y destruye cada vez que se transfiere un fichero.

En función de la implementación del servidor es posible que el proceso esclavo cree un proceso específico para gestionar la conexión de datos. En el cliente tenemos un proceso que gestiona la conexión de control y otro proceso, o no, para la conexión de datos. Siempre tenemos que tener 2 conexiones, una fija y otra dinámica.

Un esquema que ilustra este comportamiento es el que se muestra en la figura:



PI: *Process Interpreter*. Proceso que controla la conexión de control.

DTP: *Data Process Transfer*. Controla la conexión de datos.

El diálogo, a través de una conexión de control, se realiza mediante el protocolo *Telnet* simplificado, donde se especifican como se envían y reciben los datos.

En una sesión FTP el servidor para recibir una conexión de control usa el puerto 20, y para la transferencia de datos usa el puerto 21. El cliente, tanto para una conexión de control como para la conexión de datos pueden usar cualquier puerto libre, debiendo de indicárselo al servidor durante la conexión mediante el comando `port`.

PROTOCOLO SSL

El servidor *vsftpd* proporciona mecanismos de seguridad mediante el uso del protocolo SSL (Secure Sockets Layer), basado en una combinación de técnicas de encriptación y la utilización de un Certificado Digital.

- *Encriptación*

El proceso de encriptación de la información depende del uso de claves, diferenciado según el tipo de criptografía empleada:

- Convencional o simétrica: los dos finales de una transacción tienen la misma clave, la cual es usada para decodificar cada una de las otras transmisiones. Ejemplos son los algoritmos *IDEA-ECB*, *RC2-ECB* ó *CDMF-ECB*, empleados en SSL.
- Pública o asimétrica: coexisten dos claves, una clave pública y otra clave privada. Los datos codificados con la clave pública sólo pueden ser decodificados con la clave privada y viceversa. Un ejemplo son los sistemas de claves RSA (**Rivest, Shamir y Adlman**) ó DH/DSS (*Diffie-Hellman/Digital Signatura Standard*).

El conocido PGP (*Pretty Good Privacy*), o su equivalente GNU GPG (*Gnu Privacy Guard*), es un programa de cifrado híbrido, es decir, combina cifrado simétrico y asimétrico.

- *Certificación*

Los servidores FTP's seguros utilizan un certificado para identificarse con los clientes FTP's. Dichos certificados pueden ser de dos tipos:

- Certificados generados por una Autoridad de Certificación, CA (por ejemplo, *la Fábrica Nacional de Moneda y Timbre*): una autoridad externa garantiza la identidad de quien posee dicho certificado.
- Certificados autofirmados: son generados de manera completamente autónoma por el servidor que desea autenticarse frente a los clientes FTP's.

El formato de los Certificados Digitales es estándar, siendo X.509 v3 el recomendado por la Unión Internacional de Comunicaciones (ITU) y el que está en vigor en la actualidad.

El protocolo SSL proporciona cifrado de alto nivel de los datos intercambiados (se cifran incluso las cabeceras), autenticación del servidor (y si es necesario también del cliente) e integridad de los datos recibidos (cada uno de los mensajes enviados por cliente o servidor se somete a un proceso de verificación). Para ello hace uso de los sistemas de seguridad anteriores mediante el siguiente esquema:

1º Intercambio de claves públicas (indican la identidad del emisor).

2º Verificación de las claves públicas mediante un certificado (aseguran la autenticidad de las claves).

3º La conexión se protege mediante criptografía asimétrica (asegura la confidencialidad de los datos). Los sistemas asimétricos se encargan de intercambiar de forma segura las claves simétricas.

De esta forma, se consigue:

- Confidencialidad: la identidad del servidor *vsftpd* (y a veces también del cliente) se consigue mediante el Certificado Digital correspondiente.
- Autenticación: se comprueba la validez del certificado antes de iniciar el intercambio de datos sensibles.
- Integridad de los datos intercambiados: de ello se encargan la Firma digital mediante funciones *hash* y la comprobación de resúmenes de todos los datos enviados y recibidos.

2.2.1. Funcionamiento y Estructura del Protocolo

El protocolo FTP fue diseñado para ser independiente de las representaciones particulares que cada *host* de la red pudiera tener de los datos. Para lograr esta genericidad es necesario que se indique al protocolo cuatro parámetros que establecerán la forma en que se van a representar los datos del fichero que se pretende transferir:

1. Tipo del fichero.
2. Formato del fichero.
3. Estructura del fichero.
4. Modo de transmisión.

Tipo del fichero

El protocolo FTP distingue entre cuatro tipos diferentes de ficheros:

- Local: permite que dos *hosts* que usan internamente tamaños de *byte* diferentes puedan intercambiarse ficheros. Hoy en día estamos acostumbrados a trabajar siempre con *bytes* de 8 bits, pero en las primeras etapas del desarrollo de TCP/IP y FTP era común encontrarse con *hosts* que tuvieran diferentes tamaños de *bytes*, por lo que este tipo era necesario para permitir los envíos de

ficheros. Hoy día no se suele usar, aunque como podría ser útil en algún caso concreto es bueno conocer de su existencia.

- Binario: indica que el fichero debe tratarse como un flujo de datos sin ningún formato interno específico. Normalmente será el tipo a usar en la transferencia de ficheros.
- EBCDIC: Aunque no está disponible en *vsftpd* lo vamos a explicar, permite la transferencia de ficheros entre dos sistemas que usen internamente codificación EBCDIC (*Extended Binary Coded Decimal Inerchange Code*). Este tipo se usará para la transferencia de ficheros de texto.
- ASCII: al igual que EBCDIC, está pensado para la transferencia de ficheros de texto, pero esta vez entre sistemas que usen la codificación ASCII. De no indicar un tipo explícitamente para la transferencia, el protocolo tomará ASCII como tipo de fichero por defecto.

Formato del fichero

FTP define tres formatos diferentes:

- Nonprint: es el que FTP define como formato por defecto para los ficheros de texto. Este formato indica que el fichero no contiene ningún tipo de código que sirva para formatear verticalmente el texto, como pueden ser los tabuladores verticales. Estos códigos son interpretados normalmente por las impresoras.
- Telnet: sí que usa códigos para que las impresoras puedan formatear verticalmente el texto.
- Fortran: también hace uso de códigos especiales para formatear el texto.

Estructura del fichero

Se definen las siguientes:

- Fichero: es la que se toma por defecto. No hay una estructura interna concreta definida, el fichero se considera un flujo de *bytes* continuo.
- Registro: el fichero se considera un conjunto de registros secuenciales.
- Página: permite el manejo de ficheros que se estructuren en páginas independientes.

Modo de transmisión

Se definen tres modos de transmisión en el protocolo FTP:

- Block: el fichero se divide en varios bloques, a cada uno de estos bloques se le añade una cabecera, y cada bloque será enviado junto con su cabecera de forma ordenada a través de la conexión.
- Compressed: se usa un algoritmo RLE (*Run Length Encoding*) para comprimir los datos del fichero antes de ser enviados. Un algoritmo RLE es muy sencillo, consiste en sustituir una secuencia de caracteres iguales por el carácter y el número de veces que se repite.
- Stream: el fichero se interpreta como un flujo de *bytes*.

2.2.2. Uso del Protocolo en el Servicio

Vamos a indicar los comandos FTP en la siguiente tabla:

Comando	Descripción
user	Identifica al usuario ante el servidor. En servidores de acceso público es normal indicar como nombre de usuario <i>anonymous</i> o <i>ftp</i> , en otro caso se deberá indicar un nombre de usuario que esté dado de alta en el sistema.
pass	Permite especificar el <i>password</i> de un usuario identificado mediante el comando <i>user</i> .
acct	Permite especificar una cuenta de usuario. Se usa normalmente para llevar un control contable.
cwd	Permite cambiar nuestro directorio actual en el servidor FTP.
cdup	Permite ir al directorio padre del actual en el servidor FTP.
smnt	Permite montar un sistema de ficheros.
rein	Reinicia la conexión. Tras este comando el servidor esperará un comando <i>user</i> . Cualquier transferencia pendiente de la sesión anterior podrá continuar sin ser interrumpida.
quit	Finaliza la sesión FTP. Si no se están realizando transferencia este comando cierra la conexión de control inmediatamente. Si se estuviese realizando alguna transferencia, la conexión permanece abierta para que el servidor pueda enviar su respuesta, tras lo cual se cierra la conexión. Un cierre inesperado de la conexión de control equivale por parte del servidor a recibir los comandos <i>abort</i> y <i>quit</i> .

port	Permite al cliente especificar una dirección IP y un puerto al servidor, para que éste abra la conexión de datos.
pasv	Es una alternativa a port , en este caso no es el servidor quien inicia la conexión de datos hacia el cliente, sino que es el cliente el que inicia la conexión de datos hacia el servidor. Cuando el cliente le indica al servidor el comando pasv , este responde con una dirección IP y un puerto que serán los que deba usar el cliente para abrir la conexión de datos.
type	Permite indicar el tipo y el formato del fichero que se va a transferir.
stru	Indica la estructura del fichero.
mode	Modo de transferencia del fichero.
retr	Indica al servidor de FTP que envíe uno de sus ficheros, normalmente la recepción la hará el cliente que indica esta acción, aunque no tiene por qué ser necesariamente así, se podría recibir el comando desde un cliente C1 y que la recepción del fichero la realizase un cliente C2 diferente al primero.
stor	El cliente indica al servidor que le va a enviar un fichero. Tomará como parámetro un nombre de fichero. Si el fichero ya existe en el servidor y se tienen permisos suficientes, la copia existente será sobrescrita.
stou	Se comporta como stor , pero es el servidor quien elige el nombre con el que se grabará el fichero para que éste sea único.
appe	Se comporta como stor , pero si el fichero ya existe no lo sobrescribe sino que añade los nuevos contenidos recibidos al final del existente. Si el fichero no existiese se crea.
allo	Algunos servidores necesitan que se les indique con allo el tamaño que deben reservar para el fichero antes de que este sea transferido. Este comando toma como parámetro un número indicando los bytes que el servidor debe reservar. Tras allo el cliente deberá hacer un stor , stou o appe .
rest	Permite indicar el punto desde el cual se reiniciará una transferencia que se hubiese interrumpido. Toma como parámetro el punto de control desde el cual se quiere reiniciar la transferencia. Tras rest el cliente deberá usar un comando que inicie la transferencia.

rnfr	Permite al cliente renombrar un fichero del servidor. rnfr toma como parámetro el nombre del fichero a renombrar. Este comando debe ser inmediatamente seguido del comando rnto que indica el nuevo nombre que tomará el fichero.
rnto	Indica el nuevo nombre que tomará un fichero identificado con rnfr.
abor	Interrumpe una transferencia de un fichero antes de que ésta se complete. Ya se ha comentado como este comando se apoya en el protocolo Telnet para enviar esta señal de manera urgente. La interrupción de la transferencia implica el cierre de la conexión de datos por parte del servidor.
dele	Permite a un cliente que disponga de los permisos necesarios para borrar un fichero que se encuentre en el servidor. No se realiza confirmación ninguna por parte del servidor para proceder con el borrado. Tomará como parámetro el nombre del fichero a borrar.
rmd	Permite borrar un directorio si se dispone de los permisos necesarios. Como antes el servidor no solicitará ningún tipo de confirmación antes de proceder con el borrado. rmd toma como parámetro el nombre del directorio que se quiere borrar.
mkd	Permite al cliente crear un directorio en el servidor. Toma como parámetro el nombre del directorio.
pPwd	Con este comando se insta al servidor a que nos devuelva nuestro directorio actual.
list	Permite al cliente solicitar una lista de ficheros en el directorio actual, también podría indicarse como parámetro el directorio del cual se quiere recibir el listado. Este listado de ficheros es transmitido a través de la conexión de datos.
nlst	Este comando funciona como list, con la diferencia de que únicamente devuelve nombres de ficheros sin añadir información adicional.
site	Todos los comandos disponibles en una sesión de FTP son los ofrecidos por el servidor. Además de los servicios comunes a todos los servidores definidos en la especificación del protocolo, cada servidor puede ofrecer comandos propios.
syst	Permite al cliente conocer el sistema operativo que se usa en el servidor

	remoto. El servidor devolverá un código identificando el sistema operativo, estos códigos están definidos por el IANA (<i>Internet Assigned Numbers Authority</i>).
stat	Permite obtener información acerca de una transferencia que se esté realizando. Si se pasa como parámetro un nombre de directorio, el comando stat se comporta igual que list , con la diferencia de que los datos llegan por la conexión de control en vez de por la de datos.
help	El comando sin parámetros nos dará una lista de los comandos disponibles en el servidor. Si se indica como parámetro un nombre de comando el servidor nos devolverá información de uso sobre ese comando. Toda la información se devuelve por la conexión de control.
noop	Este comando realmente no hace nada. Un cliente puede usar noop para mantener una conexión abierta y evitar que se cierre por inactividad.

3. Solución Adoptada para Ofrecer el Servicio

3.1. Soluciones Existentes en el Mercado

En el mercado se encuentra multitud de servidores FTPd a continuación se les presenta una tabla con los servidores FTPd más utilizados, junto a una serie de información como una breve descripción del mismo (algunas características), download, referencia de manuales, licencia del servidor y entorno donde se ejecuta.

Servidor FTP	Descripción	Download	Documentación	Licencia	Entorno
WU-FTPD	WU-FTPD es seguramente el servidor ftp más usado en Internet, por su antigüedad y posibilidades técnicas. Además, está fuertemente auditado por equipos de programadores y hackers. Es muy frecuente que aparezcan nuevas revisiones cada cierto tiempo de esta aplicación, simplemente tapando algún que otro agujero de seguridad, lo que nos da buenas garantías y tranquilidad para usar con libertad este servidor. Si dispones de un servidor web remoto y quieres dar acceso a los ficheros para su actualización por FTP, te aconsejamos que uses WU-FTPD.	ftp://ftp.wu-ftp.org/pub/wu-ftp/wu-ftp-2.6.2.tar.gz	http://www.wu-ftp.org/	GPL	Consola
PureFTPD	Basado en 'Troll-FTPD', Pure FTP Server es rápido y estándar, como debe ser un servicio de este tipo en Internet. La seguridad es un punto fuerte de este servidor FTP, y como algo fuera de lo común, dispone de interface de configuración en XWindow. Las principales características incluidas son: - Ayuda del PAM.	http://prdownloads.sourceforge.net/pureftpd/pure-ftpd-1.0.7.tgz	http://pureftpd.sourceforge.net/	GPL	X-windows

	<ul style="list-style-type: none"> - Dominios virtuales - Los 'ls' vienen incorporados - Sistema anti-warez - Regulación de anchura de banda - Accesos limitados para conexiones pasivas - Ratios (proporción) de upload/download - Ayuda del directorio LDAP - Informe del estado del HTML y XML <p>Así mismo destaca sus posibles integraciones con bases de datos y servidores de directorio, como MySQL, PostgreSQL y LDAP. También soporta configuraciones especiales de filtros de red.</p>				
ProFTPD	<p>ProFTPD es otra alternativa para Linux en el campo de los servidores FTP. Presenta cualidades muy apreciadas en este terreno, como son escalabilidad, rendimiento, seguridad y facilidad de configuración. Permite definir dominios virtuales, FTP anónimo y control de permisos.</p> <p>Incluye numerosas nuevas características, incluyendo:</p> <ul style="list-style-type: none"> * Allow/DenyFilter estan ahora extendidas a los contextos de directorio y ftpaccess. * Nueva directiva SQLLogFile * Arreglos para TRU64 y OS/X * Numerosos bugs arreglados 	http://ftp.proftpd.org/distrib/source/proftpd-1.2.8rc2.tar.bz2	http://www.proftpd.net/	GPL	Consola
BetaFTPD	BetaFTPD es una versión mejorada del conocido ftpd de Linux. Además de ser rápido (ya que se ejecuta como un sólo proceso muy simple en Linux), es muy pequeño, ya que el ejecutable ocupa tan sólo 20 Kb.	http://betaftpd.sourceforge.net/download/betaftpd-0.0.7.tar.gz	http://betaftpd.sourceforge.net/	GPL	Consola
glFtpD	glFtpD es un servidor FTP gratuito para Linux/BSD, uno más de tantos, aunque éste destaca por su sencillez.	http://www.glftpd.org/download/glftpd-lnx_1.24.tgz	http://www.glftpd.org/	GPL	Consola
Troll-FTPD	<p>Troll-FTPD es un servidor FTP específicamente desarrollado para la plataforma Linux, ya que requiere la ejecución de algunos comandos del sistema especiales, como setsuid(), vsnprintf(), glob() y la presencia del directorio /proc.</p> <p>Está optimizado para consumir pocos recursos, y así, por ejemplo, utiliza la función mmap() en vez de read() para acceder a los ficheros de forma muy ligera. Es ya un producto con experiencia, pues se desarrolló en 1995 por Trolltech AS, y es muy conocido en el mundo de los servidores de Internet.</p>	http://ftp.trolltech.com/freebies/ftpd/troll-ftpd-1.27.tar.gz	http://www.trolltech.com/developer/download/ftpd.html	GPL	Consola
FTP4ALL	FTP4ALL es un servidor FTP para sistemas UNIX que permite su instalación sin necesidad de disponer de la cuenta root. Simplemente teniendo accesible un compilador de C, podrás instalar el servicio en tu servidor Linux/Unix.	http://www.ftp4all.de/v3/archives/ftp4all-3.012.tar.gz	http://www.ftp4all.de/	GPL	Consola
utftpd	utftpd es una revisión de tftpd, que aporta algunas mejoras, como un control de acceso más preciso, o soporte para blksize (vease el RFC 2348), así como opciones de timeout y soporte para control de revisiones.	http://www.ohse.de/uwe/releases/utftpd-0.2.4.tar.gz	http://www.ohse.de/uwe/software/utftpd.html	GPL	Consola

	<p>utftpd trabaja sobre el protocolo TFTP, que es más sencillo, y por lo tanto menos potente que FTP, pero que se usa mucho para efectuar por ejemplo, instalaciones de sistemas operativos en Red. No es aplicable a Internet, porque el protocolo es bastante inseguro, aunque utftpd, precisamente trata de mejorar este punto. lectura o escritura por parte de un cliente de/a un fichero de un servidor.</p> <p>Mediante un mecanismo de negociación de opciones, pueden hacerse extensiones al protocolo TFTP manteniendo compatibilidad hacia atrás.</p>				
NcFTPd	<p>NcFTPd es un servidor FTP de altísimo rendimiento, aunque no es código abierto, ni lleva licencia GPL. Podemos comparar el rendimiento de NcFTPd frente al de WU-FTPd. Claramente, NcFTPd está más indicado para servidores específicos de FTP, por ejemplo, donde el rendimiento es el punto clave.</p>	ftp://ftp.ncftp.com/ncftp/2.7.0/ncftp-2.7.0-linux-x86-export.tar.gz	http://www.ncftp.com/ncftp/	Comercial	Consola
BSDftpd-ssl	<p>BSDftpd-ssl es un servidor seguro de FTP para Linux y FreeBSD, y en general para cualquier máquina Unix. Soporta el protocolo TLS/SSL para tanto la transmisión, como el canal de control del FTP, de modo que todas las comunicaciones cliente servidor son encriptadas y seguras.</p>	http://bsdftpd-ssl.sc.ru/files/bsdftpd-ssl/bsdftpd-6.0-ssl-0.5.2.tar.gz	http://bsdftpd-ssl.sc.ru/	GPL	Consola
Muddleftpd	<p>Muddleftpd es un nuevo servidor FTP de licencia GPL que puede realizar una gran variedad de tareas. Es fácil de instalar, rápido, seguro y razonablemente ligero de carga. Soporta una gran variedad de módulos de autenticación PAM (password authentication module): msq, authlibexample, authlibmysql, authlibmud, authlibmysql y authlibmb.</p> <p>De forma sencilla podemos configurar listas ACL (Lista de Control de Acceso). ACL es un mecanismo de NT para el control de usuarios:</p> <p>Muddleftpd tampoco requiere permisos de root para ser instalado en un servidor Linux. Soporta dominios virtuales y ser controlado por cuenta propia (standalone) o mediante inetd.</p>	http://www.arachnet.au/~wildfire/muddleftpd/muddleftpd.1.3.11.tar.gz	http://www.arachnet.au/~wildfire/muddleftpd/	GPL	Consola
hFTPd	<p>Hoser FTPD es un servidor FTP para Linux muy sencillo y ligero, que se comporta muy bien en máquinas con pocos recursos, y por otro lado, escala fenomenalmente en servidores grandes con cargas importantes en el servicio FTP.</p>	http://www.zabbonet/hftpd/hftpd-0.5.0.tar.gz	http://www.zabbonet/hftpd/	GPL	Consola

3.2. Referencias y Características de la Solución Adoptada

Para la realización de este trabajo se ha utilizado el servidor *vsftpd-2.0.3* que es la última versión que existe hasta el momento, cuyo autor es [Chris Evans](#).

3.3. Equipamiento Necesario

- PC Pentium IV 1.5 GHz; 128 MB RAM; 20 GB HD.
- Red de Área Local *Ethernet 10BaseT*.
- *Router*.

4. Parámetros de Configuración del Servidor

Los parámetros de configuración del servidor se encuentra en el fichero *vsftpd.conf*, por defecto el servidor busca este fichero en la localización */etc/vsftpd.conf*. Sin embargo tu puedes ignorar esto para especificar un argumento en la línea de comando para *vsftpd*.

El formato del archivo *vsftpd.conf* es muy simple, cada línea o es un comentario o es una directiva. Los comentarios comienzan por *#* y son ignorados y las directivas tienen el siguiente formato

option=value

Es importante decir que es un error muy típico poner un espacio entre **option=value**

Cada opción tiene un valor por defecto el cual puede ser modificado en la configuración del archivo

PARÁMETROS BOOLEANOS

Se caracterizan por que tienen como valor YES o NO

Parámetro	Default	Descripción
anon_mkdir_write_enable	NO	Si se pone YES los usuarios anónimos se le permiten crear nuevos directorios bajo ciertas condiciones. Para hacer esto la opción <i>write_enable</i> debe estar activada, y el usuario ftp anónimo debe tener permiso de escritura.
anon_other_write_enable	NO	Si se pone YES, los usuarios anónimos se le permiten las operaciones de escritura así como subir ficheros, crear directorios, eliminar y cambiar de nombre. Esto generalmente no es recomendable.
anon_upload_enable	NO	Si vale YES, los usuarios anónimos se le permiten subir archivos bajo ciertas condiciones. Para ello, la opción <i>write_enable</i> debe estar activada, y los clientes ftp anónimos debe tener permiso de escritura en la

		localización de subida deseada
anon_world_readable_only	YES	Cuando está activada, los clientes anónimos solo se le permitirán bajar archivos los cuales sean legibles, es decir hacer solo lectura de archivos subidos.
anonymous_enable	YES	Permitir el acceso a clientes anónimos
ascii_download_enable	NO	Cuando está activada, la transferencia de datos se reconocerá en modo ASCII en la bajada.
ascii_upload_enable	NO	Cuando está activada, la transferencia de datos en modo ASCII será reconocida en la subida.
async_abor_enable	NO	Cuando esta activada, un comando FTP especial conocido como “async ABOR” será activado.
check_shell	YES	Esta opción solo tiene efecto para estructuras que no sean PAM de vsftpd. Si está desactivado, vsftpd no comprobará /etc/shells para un cliente válido.
chown_uploads	NO	Si está activado, todos los archivos desconocidos subidos tiene que pertenecer a un cliente especificado por la opción chown_username
chroot_list_enable	NO	Restringe a los clientes a sus directorios personales (no puede tomar archivos del arbol de directorio por debajo de \$HOME).
chroot_local_user	NO	Permite el acceso para todos los clientes en sus directorios personales
connect_from_port_20	NO	Esto se asegura si el puerto de conexión de datos que se va a usar en el servidor es el puerto 20 (ftp-data)
deny_email_enable	NO	Si está activada, tu puedes facilitar una lista de contraseñas anónimas los cuales sus direcciones e-mail son especificados en <i>vsftpd.banned</i> .
dirmessage_enable	NO	Si está activada. A los clientes remotos del servidor FTP le aparecen mensajes cuando ellos entra por primera vez en un directorio.
guest_enable	NO	Si está activada, todos los <i>logins</i> de clientes conocidos son clasificados como “guest” logins.
hide_ids	NO	Si está activada, todos los clientes y grupo de información en directorios que estén escuchando será mostrado como “ftp”
listen	NO	Permite el modo Standalone.
local_enable	NO	Controla si los clientes locales tienen permisos o no

log_ftp_protocol	NO	Cuando está activada, todas las peticiones y respuestas FTP son guardadas en el fichero logs.
ls_recurse_enable	NO	Activa la escucha recursiva
no_anon_password	NO	Especifica si los clientes anónimos o desconocidos tienen que presentar una contraseña
one_process_model	NO	Previene a vsftpd de preguntas para una contraseña anónima.
passwd_chroot_enable	NO	Si está activada, junto con chroot_local_user, cada zona reservada del cliente proviene de sus directorios personales en una cadena /etc/passwd
pasv_enable	YES	Poner NO si se quiere denegar el método PASV de obtener una conexión de datos
pasv_promiscuous	NO	Poner YES si se quiere desactivar la comprobación segura PASV.
port_enable	YES	Poner NO si se quiere denegar el puerto para el método de obtener una conexión de datos.
port_promiscuous	NO	Poner YES si lo que se desea es deshabilitar la comprobación segura del puerto para la conexión de datos.
setproctitle_enable	NO	Si está activada, vsftpd intentará y mostrará información del estado de la sesión en el proceso de escucha.
tcp_wrappers	NO	Si está activada, y vsftpd fue compilado con el soporte tcp_wrappers, las conexiones entrantes avanzará a través del control de acceso de tcp_wrappers.
text_userdb_names	NO	Para conseguir nombres textuales de los clientes, ya que por defecto tienen una identificación numérica.
use_localtime	NO	Si está activada, vsftpd mostrará los directorios de escuchas.
use_sendfile	YES	Una opción interna usada para testear el beneficio relativo de usar sendfile().
userlist_deny	YES	Cuando está activada, userlist_file posee una lista de los clientes con acceso denegado. Cuando está a NO, userlist_file posee una lista de clientes, y solo estos clientes, se le permiten el acceso.
userlist_enable	NO	Niega el acceso a los clientes especificados en user_list_file
write_enable	NO	Si no se activa no se pueden subir ficheros, ya que se

		descarta cualquier petición de escritura.
xferlog_enable	NO	Si está activada, el fichero log tendrá todos los detalles de las subidas y de las bajadas.
xferlog_std_format	NO	Si está activada el fichero log de transferencia escribirá en un formato xferlog estándar de ftpd, también usado por wu-ftp.

PARÁMETROS NUMÉRICOS

Se caracterizan por que tienen como valor un número entero.

Parámetro	Default	Descripción
accept_timeout	60	Tiempo establecido sin recibir respuesta de un cliente remoto en el establecimiento de la conexión.
anon_max_rate	0 *	Tasa máxima de transferencia de datos permitidos en bytes por segundos.
anon_umask	077	Mascara para el fichero creado del cliente anónimo.
connect_timeout	60	Tiempo esperando en segundos, para un cliente remoto respondiendo en nuestro puerto de conexión de datos.
data_connection_timeout	300	Tiempo esperando en segundos sin que el cliente envíe transferencia de datos.
file_open_mode	0666	Permiso para la subida de ficheros.
ftp_data_port	20	Puerto por el cual se conecta la transferencia de datos.
idle_session_timeout	300	Tiempo por defecto en segundos para cortar una conexión en pausa/idle.
listen_port	21	Para el modo Standalone este es el puerto por el que se escucha las conexiones FTP entrantes.
local_max_rate	0 *	Tasa máxima de transferencia de datos permitida en bytes por segundos para un cliente local autenticado.
local_umask	077	Máscara de permisos para clientes locales. En caso de activar escritura (subir ficheros, etc) esta opción marca la mascara por defecto de los ficheros a crear.
max_clients	0 *	En modo Standalone, es el número máximo de clientes que pueden estar conectados.
max_per_ip	0 *	En modo Standalone, es el número máximo de clientes que pueden estar conectados con la misma dirección de Internet.

pasv_max_port	0 **	Máximo puerto localizado para conexiones de datos tipo PASV.
pasv_min_port	0 **	Mínimo puerto localizado para conexiones de datos tipo PASV.

Nota.- * Ilimitado. ** Usa cualquier puerto

PARÁMETROS ALFABÉTICOS

Se caracteriza por que tiene como valor un **string**.

Parámetro	Default	Descripción
anon_root	(none)	Representa un directorio el cual vsftpd intentará cambiar a un <i>login</i> desconocido.
banned_email_file	/etc/vsftpd.banned_emails	Nombre de un fichero que contiene una lista de contraseñas e-mail desconocidas las cuales no son permitidas.
banner_file	(none)	Es el nombre de un archivo que contiene texto para mostrar cuando alguien se conecta al servidor.
chown_username	Root	Nombre de un cliente a quien se le da la propiedad de subir ficheros desconocidos.
chroot_list_file	/etc/vsftpd.chroot_list	Nombre de un fichero que contiene una lista de clientes locales los cuales serán colocados en un chroot() encasillados en sus directorios personales.
guest_username	ftp	Mira la opción booleana <code>guest_enable</code> para una descripción de que constituye un login guest
ftp_username	ftp	Nombre del cliente que usamos para manejar un FTP desconocido.
ftpd_banner	(none)	Permite saltarse el banner para que no aparezca el mensaje de bienvenida cuando comienza una conexión.
listen_address	(none)	En modo Standalone, la dirección que escucha por defecto.
local_root	(none)	Representa un directorio el cual vsftpd

		intentará cambiar después de un login local
message_file	.message	Nombre del archivo que buscamos cuando un nuevo directorio ha entrado. El contenido es mostrado en el cliente remoto.
nopriv_user	Nobody	Nombre del cliente que es usado por vsftpd cuando quiere estar en modo no privilegiado.
pam_service_name	ftp	Nombre de l servicio PAM que vsftpd usará.
pasv_address	(none) *	Para saltarse la dirección IP que vsftpd advertirá como respuesta del comando PASV.
secure_chroot_dir	/usr/share/empty	Debería ser el nombre de un directorio el cual este vacío.
user_config_dir	(none)	Permite pasar de alguna opción de configuración especificado en la página del manual.
userlist_file	/etc/vsftpd.user_list	Nombre de un fichero cargado cuando la opción userlist_enable está activada.
xferlog_file	/var/log/vsftpd.log	Nombre del fichero en el cual nosotros escribimos la transferencia de logs.

Nota.- * La dirección es tomada de la conexión del *socket*

Los ficheros que son utilizados por vsftpd son los siguientes

Fichero	Descripción
vsftpd.ftpusers	Usuarios a los que siempre se le deniega el acceso
vsftpd.user_list	Especifica los usuarios con acceso denegado (Permite el acceso si userlist_deny es NO)
vsftpd.chroot_list	Lista de usuarios locales a los que se le permiten el acceso (tienen acceso denegado si chroot_local_user es NO)
/etc/vsftpd.conf	Fichero de configuración del servidor vsftpd
/etc/pam.d/vsftpd	Script PAM vsftpd
/etc/rc.d/init.d/vsftpd	Script del servicio vsftpd, modo Standalone

/etc/xinetd.d/vsftpd	Script xinetd de vsftpd (Para el caso de usar xinetd)
----------------------	---

5. Proceso de Instalación/Administración del Servidor

A continuación vamos a describir el proceso de instalación y administración del servidor *vsftpd-2.0.3* que hemos utilizado para la elaboración de este trabajo.

5.1. Obtención del Código

Podemos obtener el código en muchas sitios de Internet, para la realización de este trabajo como hemos dicho se utilizó la versión del servidor *vsftpd-2.0.3*, que es la última versión en el momento de la realización de este trabajo, descargada de la dirección <ftp://vsftpd.beasts.org/users/cevans/> hay que destacar que el código es totalmente gratuito cuya licencia la tiene GPL.

5.2. Instalación del Servidor

Los pasos que vamos a seguir para la instalación del servidor a partir de su código fuente son los siguientes:

- 1) Nos bajamos el fichero comprimido *vsftpd-2.0.3.tar.gz*

```
$ wget ftp://vsftpd.beasts.org/users/cevans/vsftpd-2.0.3.tar.gz
```

- 2) Descomprimos el paquete *vsftpd* y entramos en la carpeta descomprimida

```
$ tar -xvzf vsftpd-2.0.3.tar.gz
```

```
$ cd vsftpd-2.0.3
```

- 3) Vemos si el paquete está ya instalado.

```
$ rpm -q vsftpd
```

Si está instalada una versión más antigua y la queremos actualizar, eliminamos la versión anterior para evitar posibles conflictos.

```
$ rpm -e vsftpd
```

- 4) Para la instalación de *vsftpd* escribimos lo siguiente:

```
$ make
```

Se necesita al usuario *nobody* (para ello hay que estar como *root*), si no existe se crea simplemente escribiendo:

```
$ useradd nobody
```

Además *vsftpd* requiere de un directorio */usr/share/empty*. Si no existe lo creamos con el siguiente comando:

```
$ mkdir /usr/share/empty
```

Si tu objetivo es tener un servidor FTP anónimo entonces escribimos lo

```
$ mkdir /var/ftp
```

```
$ useradd -d /var/ftp ftp
```

```
$ chown root:root /var/ftp
```

```
$ chmod og-w /var/ftp
```

5) Ahora instalamos escribiendo el siguiente comando:

```
$ make install
```

El comando **\$ make install** debería haber copiado todos los archivos necesarios en su sitio, pero eso no ocurre siempre, por tanto, debería copiarlo manualmente o asegurarte de que existe los siguientes archivos en su localización correcta, los comandos necesarios sería:

```
$ cp vsftpd /usr/local/sbin/vsftpd
```

```
$ cp vsftpd.conf.5 /usr/share/man/man5
```

```
$ cp vsftpd.8 /usr/share/man/man8
```

Ahora copiamos el fichero de configuración principal en su localización correspondiente:

```
$ cp vsftpd.conf /etc
```


6) Una vez hecho todo esto hay que crear un archivo que por causa aun desconocida el comando `make install` no lo realiza, este fichero es el PAM que se encarga de la autenticidad de los usuarios locales, por ello lo vamos a hacer manualmente, creamos el fichero *vsftpd* en la siguiente dirección */etc/pam.d/vsftpd*, el código de dicho archivo se encuentra a continuación:

/etc/pam.d/vsftpd

```
##PAM-1.0

auth      required      pam_listfile.so          item=user          sense=deny
file=/etc/vsftpd.ftpusers onerr=succeed

auth      required      pam_stack.so service=system-auth

auth      required      pam_shells.so

account   required      pam_stack.so service=system-auth

session   required      pam_stack.so service=system-auth
```

Otra forma de instalarlo es mediante la instalación de un paquete .rpm para ello hacemos lo siguiente:

- 1) Descarga del paquete .rpm de *vsftpd* que lo podemos encontrar en las siguientes direcciones <https://rhndhat.com/> o bien en <http://www.rpmfind.net/>.
- 2) Una vez adquirido el paquete .rpm instalamos escribiendo la siguiente línea:

```
$ rpm -uvh vsftpd-x.y.z-8.i386.rpm
```

Donde x, y y z indica la versión de *vsftpd* que nos hemos descargado. Para nuestro caso hemos utilizado el *vsftpd* 2.0.3 que también se encuentra disponible en las URL citadas anteriormente.

Una vez hecho eso se puede decir que ya está oficialmente instalado, es mucho más sencillo que compilándolo, pero si se produce un conflicto o error es más difícil de detectar ya que no conocemos los pasos que se han realizados. También hay que destacar que el destino de algunos archivos pueden diferir de si usamos un método u otro por lo que hay que tener cuidado con esto.

5.3. Configuración del Servidor

Vamos a ejecutar el servidor *vsftpd* en modo *standalone* y no iniciado con *xinetd* por lo que tenemos que agregar al archivo de configuración del servidor */etc/vsftpd.conf* la siguiente línea *listen=YES* ya que por defecto está deshabilitada.

Una vez realizado lo anterior vamos a crear el *script* de servicio que se encontrará ubicado en la dirección */etc/rc.d/init.d/vsftpd* y cuyo código es el siguiente:

/etc/rc.d/init.d/vsftpd

```
#!/bin/bash
# vsftpd      This shell script takes care of starting and stopping
#             standalone vsftpd.
# chkconfig: - 60 50
# description: Vsftpd is a ftp daemon, which is the program \
#             that answers incoming ftp service requests.
# processname: vsftpd
# config: /etc/vsftpd/vsftpd.conf
# Source function library.
. /etc/rc.d/init.d/functions
# Source networking configuration.
. /etc/sysconfig/network
# Check that networking is up.
[ ${NETWORKING} = "no" ] && exit 0
[ -x /usr/local/sbin/vsftpd ] || exit 0
RETVAL=0
prog="vsftpd"
start() {
    if [ -d /etc/vsftpd ] ; then
        for i in `ls /etc/vsftpd.conf`; do
            site=`basename $i .conf`
            echo -n "Starting $prog for $site: "
            /usr/local/sbin/vsftpd $i &
            RETVAL=$?
            [ $RETVAL -eq 0 ] && {
                touch /var/lock/subsys/$prog
                success "$prog $site"
            }
            echo
        done
    else
        RETVAL=1
    fi
    return $RETVAL
}

stop() {
    # Stop daemons.
    echo -n "Shutting down $prog: "
    killproc $prog
    RETVAL=$?
    echo
    [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/$prog
    return $RETVAL
}

case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart|reload)
        stop
        start
        RETVAL=$?
        ;;
    condrestart)
        if [ -f /var/lock/subsys/$prog ]; then
            stop
            start
            RETVAL=$?
        fi
        ;;
    status)
        status $prog
        RETVAL=$?
        ;;
    *)
        echo $"Usage: $0 {start|stop|restart|condrestart|status}"
        exit 1
esac
exit $RETVAL
```

Además vamos a hacer que se inicia automáticamente con el encendido del ordenador para ello escribimos lo siguiente:

```
$ chkconfig --level 5 vsftpd on
```

Si quisiéramos iniciarlo con el superservidor *xinetd*, aunque no es lo recomendado, lo vamos a explicar también como se haría:

Nos aseguráramos de que en el archivo de configuración del servidor */etc/vsftpd.conf* aparece la línea **listen=NO** que indica que no va a funcionar en modo *standalone*, es decir, de forma autónoma.

Ahora configuramos *xinetd*, para ello creamos el fichero */etc/xinetd.d/vsftpd* con el siguiente contenido

/etc/xinetd.d/vsftpd

```
# default: off
# description: The vsftpd FTP server serves FTP connections. It uses \
# normal, unencrypted usernames and passwords for authentication.
service ftp
{
  disable = no
  socket_type = stream
  wait = no
  user = root
  server = /usr/local/sbin/vsftpd
  nice = 10
}
```

5.4. Puesta en Funcionamiento del Servicio

Para la puesta en marcha del servidor *vsftpd* (modo *standalone*) solo hay que escribir el siguiente comando:

```
$ service vsftpd start
```

Para la puesta en marcha del servidor *vsftpd* (modo *xinetd*) solo hay que escribir el siguiente comando:

```
$ /etc/rc.d/init.d/xinetd restart
```

Si lo hemos configurado para que se inicie con el *pc* no haría falta escribir dichos comandos (solo sí aun no hemos reiniciado el ordenador desde la instalación).

NOTA.- A partir de ahora vamos a trabajar con el servidor en modo *standalone*.

5.5. Administración, Monitorización y Prueba del Funcionamiento

Vamos a indicar aquí algunos de los parámetros más importantes que se suelen usar en la utilización del servidor *vsftpd* y también indicaremos los resultados de dichas modificaciones comprobando así el correcto funcionamiento del servidor *vsftpd*.

Habilitación del login de acceso

En el siguiente ejemplo tomado del fichero *vsftpd.conf*, el FTP anónimo es habilitado asignándole el valor YES a la opción *anonymous_enable*. La opción *local_enable* permite a los usuarios locales de tu sistema usar el servidor FTP.

```
#Allow anonymous_enable=YES
```

```
#
```

```
#Uncomment this to allow local users to log in.
```

```
local_enable=YES
```

Si permite el acceso a los usuarios anónimos y no quiere que escriba una contraseña ya que ésta sería irrelevante, se conseguiría poniendo

```
no_anon_password=YES
```

El resultado que sale por pantalla cuando accedemos con un usuario local sería:

```
[root@localhost home]# ftp localhost
Connected to localhost.localdomain.
220 (vsFTPd 2.0.3)
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (localhost:root): usuariolocal
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

Y cuando entramos con un usuario anónimo sería:

```
[root@localhost home]# ftp localhost
Connected to localhost.localdomain.
220 (vsFTPd 2.0.3)
530 Please login with USER and PASS.
```

```
530 Please login with USER and PASS.  
KERBEROS V4 rejected as an authentication type  
Name (localhost:root): ftp  
230 Login successful.  
Remote system type is UNIX.  
Using binary mode to transfer files.
```

Permisos a los usuarios locales

Tenemos gran variedad de control de permisos de usuarios, como los usuarios locales pueden acceder a los ficheros del servidor. Si tú quiere permitir a los usuarios locales crear ficheros, cambiarles de nombre y eliminarlos o a los directorios en sus cuentas, tú como administrador tiene que habilitar el acceso de escritura con la opción `write_enable`. La opción anterior activa un rango de comandos para cambiar los archivos de tu sistema, incluyendo crear, cambiar de nombre y eliminar tanto ficheros como directorios. Por ello vamos a activarla escribiendo en el fichero de configuración:

```
write_enable=YES
```

Tu puedes especificar los permisos de subida usando la opción `local_umask` (022 es el que está por defecto, que significa lectura y escritura para el dueño y solo lectura para los otros usuarios, 644).

```
local_umask=022
```

Aunque los archivos subidos con formato *ASCII* es deshabilitado por defecto, tú puedes activar su funcionamiento. Subir archivos con formato *ASCII* provoca un cierto riesgo en la seguridad es por ello que se anula por defecto. Pero si tú quieres subir grandes ficheros de textos, podrías activarlos para este caso especial. Usa la opción `ascii_upload_enable` para permitir subidas con formato *ASCII*.

Permisos de los usuarios anónimos

Tú puedes también permitir a los usuarios anónimos subir y eliminar ficheros, a igual que crear y eliminar directorios. La subida para los usuarios anónimos es habilitada con la opción `anon_upload_enable`. Permitir a los usuarios anónimos cambiar de nombre o eliminar sus ficheros, se puede activar con la opción `anon_other_write_enable`. También se le puede permitir crear directorios, activando la opción `anon_mkdir_write_enable`.

```
anon_upload_enable=YES
```

```
anon_other_write_enable=YES
```

```
anon_mkdir_write_enable=YES
```

La opción `anon_world_readable_only` haría solo la lectura de ficheros previamente subidos, limitando el acceso de escritura solo a los usuarios que lo han creado, es decir, solo los usuarios que ha lo han subido los pueden eliminar.

Todos los archivos subidos son dueño del usuario anónimo FTP. Tú puedes tener acceso de lectura a los archivos creados por otros usuarios. Habilitando esta opción, tú puedes usar `chown_uploads` y especificar el nuevo usuario con `chown_username`. Nunca hacer un usuario de un usuario administrativo como por ejemplo el *root*.

```
chown_uploads=YES
```

```
chown_username=myftpfiles
```

El directorio de subida debería dar permisos de escritura para otros usuarios.

```
chmod 777 /var/ftp/upload/
```

Tú puedes controlar la clase de acceso que los usuarios tienen sobre los archivos, con la opción `anon_mask`, colocando el permiso de escritura/lectura por defecto para los archivos subidos. El valor por defecto es 077, el cual da permiso de lectura/escritura solo a los usuarios dueños (600). Permitir a todos los usuarios el acceso en lectura, tu deberías poner la mascara a 022, donde el 2 deshabilita el permiso de escritura pero pone el permiso de lectura (644). El valor 000 debería permitir tanto lectura como escritura para todos los usuarios.

Limitación del tiempo de conexión.

Para más control efectivo de las subidas al servidor; tú puedes poner limitación de tiempo a los usuarios conectados y cortar la transmisión. La opción `idle_session_timeout` desconecta a los usuarios conectados después de un tiempo específico, y la opción `data_connection_timeouts` desconecta la conexión de datos. El valor por defecto es el mostrado a continuación:

```
idle_session_timeout=600
```

```
data_connection_timeout=120
```

Mensajes.

La opción `dirmessage_enable` permite poner un mensaje en un directorio, ese archivo mensaje es lanzado siempre a un usuario que accede al directorio. `ftpd_banner` permite poner el mensaje a tu FTP, el valor por defecto es el valor mostrado a continuación:

```
ftpd_banner= Bienvenido al VSFTPD de jose luis cid castillo.
```

Se puede ver que el mensaje sale por pantalla.

```
[root@localhost home]# ftp localhost
Connected to localhost.localdomain.
220 Bienvenido al VSFTPD de jose luis cid castillo.
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (localhost:root):
```

Control de acceso a Vsftpd.

La opción `deny_email_enable` permite denegar el acceso para los usuarios anónimos, y la opción `banned_email` lo designa al fichero (normalmente `vsftpd.banned`) este sujeta las direcciones e-mail de estos usuarios. El archivo `ftpusers` contiene una lista de estos usuarios que nunca podrá acceder. Estos son normalmente usuarios del sistema como *root*, *mail*, y *nobody*.

Veamos que ocurre cuando queremos entrar con los usuarios anteriores:

```
[root@localhost home]# ftp localhost
Connected to localhost.localdomain.
220 Bienvenido al VSFTPD de jose luis cid castillo.
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (localhost:root): root
530 Permission denied.
Login failed.
ftp> user
(username) mail
530 Permission denied.
Login failed.
ftp> user
(username) nobody
530 Permission denied.
Login failed.
```

Acceso de usuario.

La opción `userlist_enable` controla el acceso para los usuarios, denegando el acceso a todos los que están listado en el archivo activado con la opción `userlist_file` (normalmente el fichero es `vsftpd.user_list`). Si tú quieres restringir el acceso a ciertos usuarios, tú puedes cambiar el significado y usar al fichero `user_list` para indicar solo el acceso a los clientes que están listado, denegando el acceso al resto. Haz esto, tu pon la opción `userlist_deny` a NO (por defecto esta a YES). Solo los usuarios listado en el archivo `user_list` podrán acceder al sitio FTP.

Hemos puesto al usuario del sistema *jose Luis* en el fichero *user_list* tenemos dos opciones:

- **userlist_deny=YES** el resultado por pantalla al acceder con el usuario *jose Luis* es:

```
[root@localhost vsftpd]# ftp localhost
Connected to localhost.localdomain.
220 Bienvenido al VSFTPD de jose luis cid castillo.
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (localhost:root): jose Luis
530 Permission denied.
Login failed.
```

- **userlist_deny=NO** el resultado por pantalla al acceder con el usuario *jose Luis* es:

```
[root@localhost vsftpd]# ftp localhost
Connected to localhost.localdomain.
220 Bienvenido al VSFTPD de jose luis cid castillo.
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (localhost:root): jose Luis
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

Restricciones al cliente.

La opción **chroot_list_enable** controla el acceso a los usuarios locales, permitiendo el acceso solo a sus directorios *home*, mientras restringe acceso al sistema. La opción **chroot_list_file** designa al fichero (normalmente *vsftpd.chroot*) esta contiene una lista de los clientes a los que se le permiten el acceso. Se permitirá el acceso para todos los clientes locales con la opción **chroot_local_user**. Si esta opción está activada, entonces el archivo designado por **chroot_list_file** tendrá un significado inverso, listando aquellos clientes a los cuales no se le permite el acceso. En el siguiente ejemplo, el acceso para los clientes locales es limitado a aquellos que estén listados en el fichero *chroot*.

chroot_list_enable=YES

`chroot_list_file=/etc/chroot_list`

Poniendo al usuario *joseluis* en el fichero *chroot_list* y accediendo al FTP con éste tenemos:

```
[root@localhost vsftpd]# ftp localhost
Connected to localhost.localdomain.
220 Bienvenido al VSFTPD de jose luis cid castillo.
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (localhost:root): joseluis
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /home/usuariolocal/
550 Failed to change directory.
```

Autenticación del usuario.

El servidor *vsftpd* hace uso de los servicios PAM para autenticar los clientes locales que acceden de forma remota a sus cuentas a través del servicio FTP. En el archivo *vsftpd.conf*, el *script* PAM usado por el servidor es especificado con la opción `pam_service_name=vsftpd`

En el directorio `etc/pam.d`, encontrarás un archivo PAM llamado *vsftpd* con entradas para controlar el acceso al servidor *vsftpd*. PAM es a la misma vez activado para autenticar a los usuarios con una cuenta valida, también deniega el acceso a los usuarios que estén en el fichero */etc/ftpusers*.

Para comprobarlo vamos a poner al usuario *joseluis* en el fichero *ftpusers* y vamos a intentar acceder al *ftpd* con este usuario.

```
[root@localhost vsftpd]# ftp localhost
Connected to localhost.localdomain.
220 Bienvenido al VSFTPD de jose luis cid castillo.
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (localhost:root): joseluis
331 Please specify the password.
Password:
530 Login incorrect.
Login failed.
```

Usuarios virtuales de vsftpd.

Los usuarios virtuales pueden ser implementados haciendo uso de los servicios PAM para autenticar clientes autorizados. En efecto, se está permitiendo el acceso a unos ciertos usuarios, mientras no tenga activadas las cuentas para ellos en el servidor FTP. Primero crear un archivo de base de datos de *login* PAM para usar con un archivo PAM en el directorio */etc/pam.d/* que accederá a la base de datos. Entonces crear un FTP virtual con sus correspondientes directorios con el que el usuario virtual accederá. Entonces en el archivo de configuración */etc/vsftpd.conf*, debes deshabilitar FTP anónimos:

```
anonymous_enable=NO
```

```
local_enable=YES
```

Habilitamos el acceso al huésped

```
guest_enable=YES
```

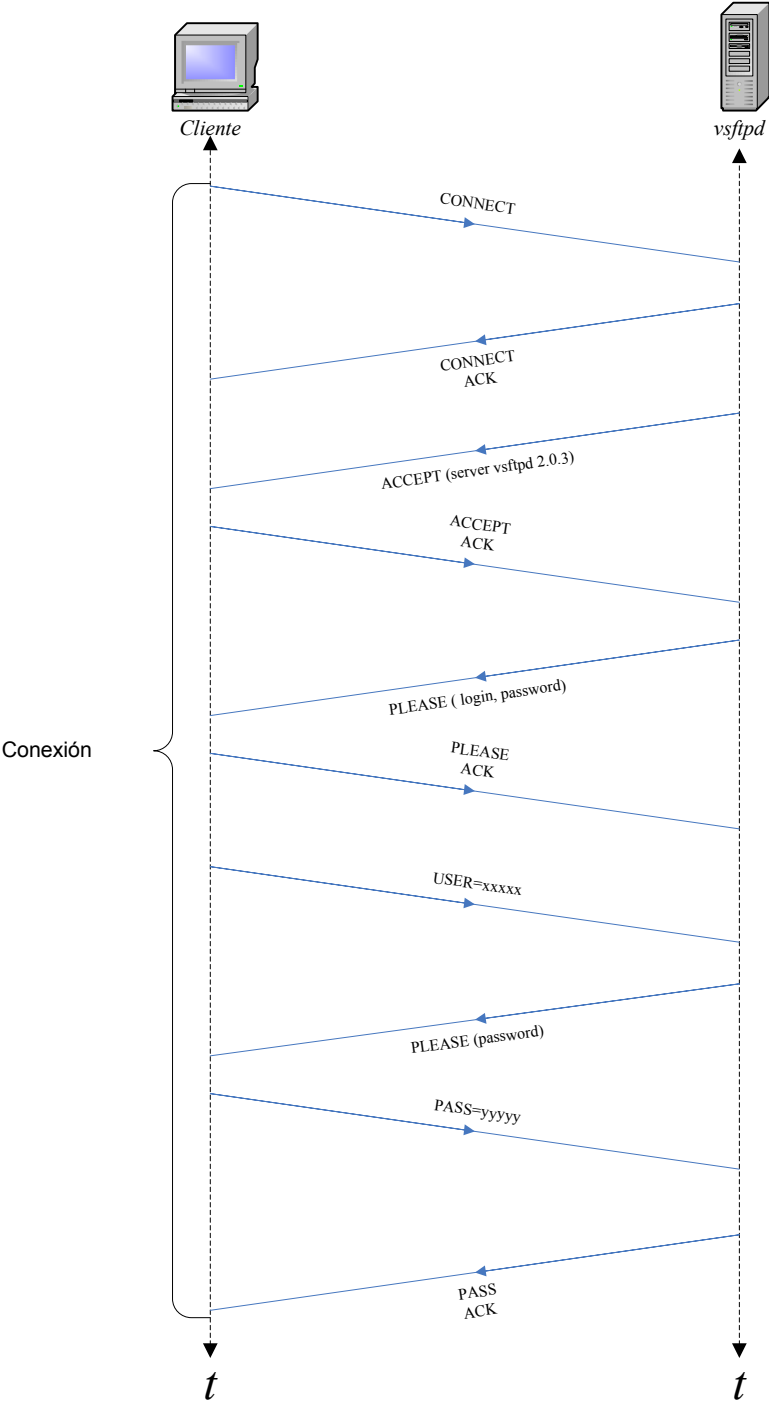
```
guest_username=virtual
```

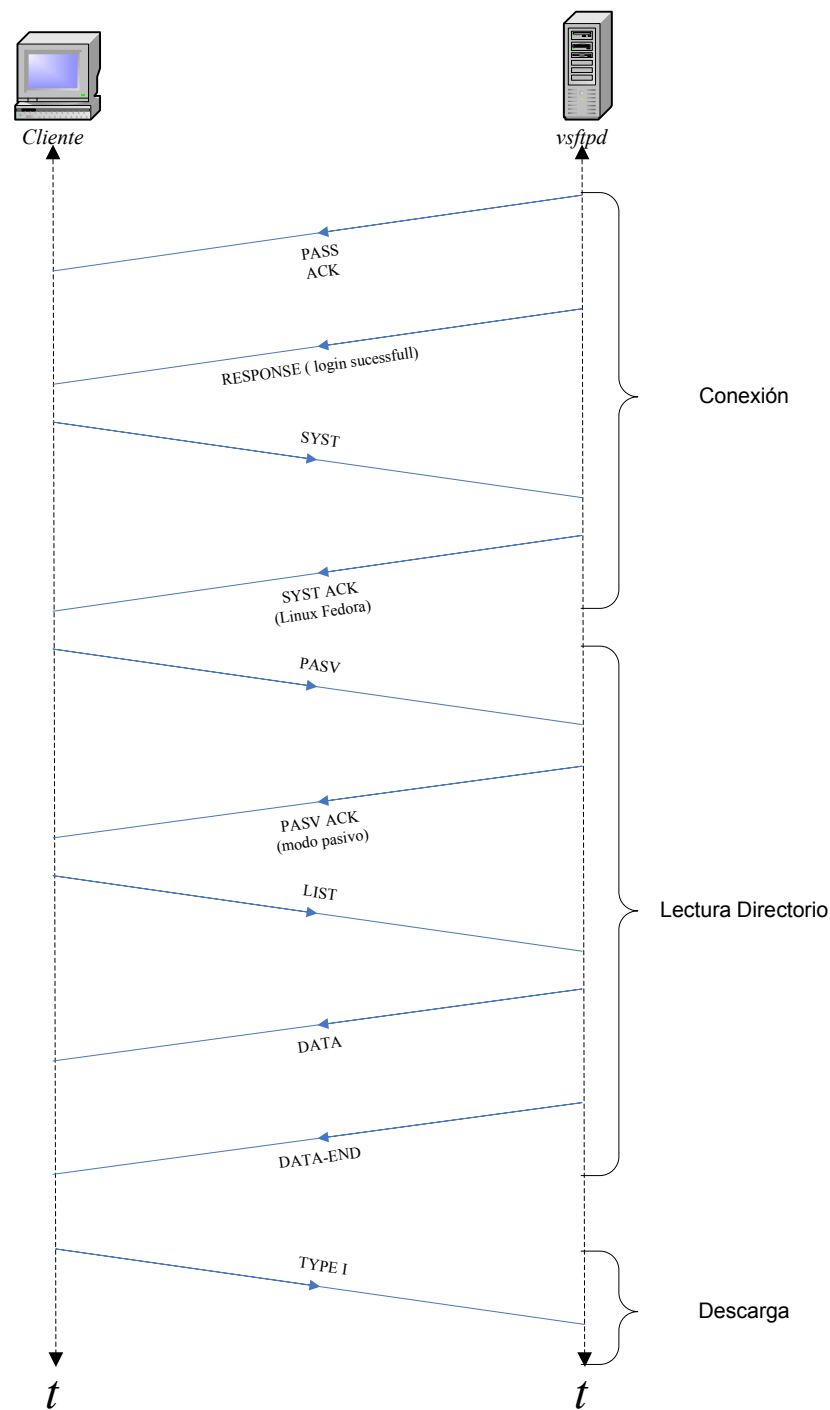
6. Análisis del Intercambio de Mensajes por la Red

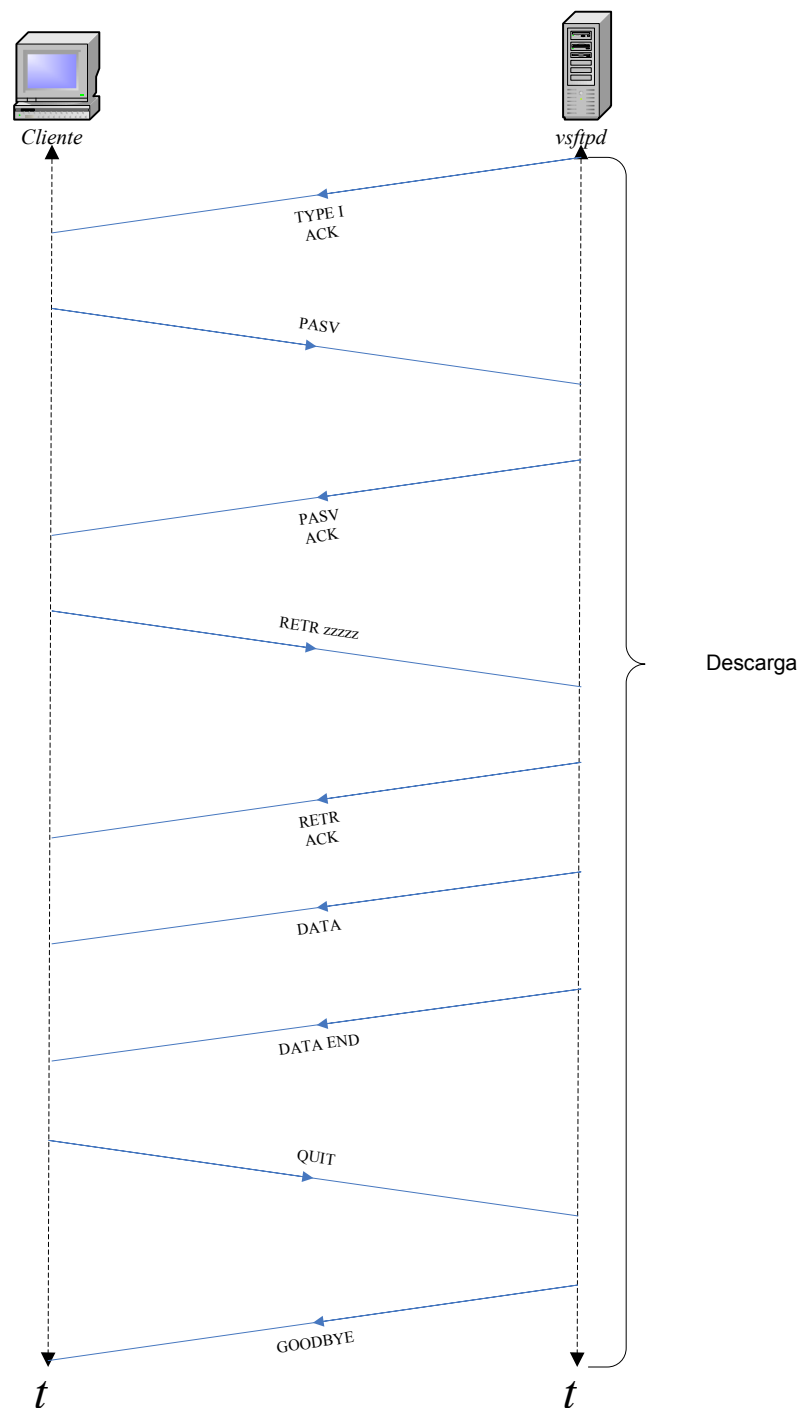
Para el análisis del intercambio de Mensajes hemos utilizado el *ethereal*, herramienta gráfica que trae el *Fedora Core* y que sirve para la captura e interpretación del tráfico que fluye por la red, o parte de ella. Como es lógico no vamos a escribir todos los paquetes del servidor y cliente FTP que ha capturado *ethereal*, pero si vamos a poner un esquema de lo que ha capturado.

El ejemplo que hemos tomados es el siguiente, hemos utilizado un cliente que ha hecho lo siguiente de forma muy esquemática:

1. Conectarse con el Servidor *vsftpd* pasándole su *login*: *xxxxx* y su *password*: *yyyyy*.
2. Leer lo que hay en la carpeta del usuario en el servidor.
3. Descargarse el fichero *zzzzz*.







Ahora vamos a describir un poco más el paso de mensajes:

1. Conexión: En primer lugar el cliente le envía un mensaje de conexión **connect** al servidor *vsftpd*, este le responde con un asentimiento. Una vez recibida esa petición, el servidor le envía un mensaje **accept** indicándole que acepta la conexión, dentro de este mensaje le indica también la versión del servidor, en

este caso *vsftpd 2.0.3*, el cliente le devuelve un asentimiento. A continuación el servidor le solicita el *login* y el *password* del cliente, una vez que este se lo ha enviado todo el servidor lo identifica y le envía un mensaje de conexión con éxito. Una vez hecho esto el cliente le solicita al servidor que sistema operativo está utilizando, esto se hace después de la conexión por motivos de seguridad.

2. Lectura del directorio /home: El cliente le envía un mensaje al servidor indicándole el sentido de la conexión de datos, una vez establecida le solicita *list* que sirve para ver la lista de ficheros del directorio actual, es servidor se lo envía en forma de **data**.
3. Descarga de un fichero: El cliente envía ahora **type** que indica que se quiere realizar una transferencia de un fichero, después le envía **pasv** para indicar la dirección de transferencia de datos por el puerto de transferencia de datos, a continuación se le envía **retr** con el nombre del fichero que indica que lo que se quiere hacer es recibir el fichero, el servidor se lo envía en forma de **data**, una vez realizado todo esto se cierra la conexión con la orden **quit**.

7. Interfaz Gráfica de Gestión

Hasta el momento no tengo constancia de una interfaz gráfica de gestión del servidor *vsftpd*.

8. Ampliaciones/Mejoras del Servicio

Enfrentamiento con el desbordamiento del buffer.

Probablemente el defecto más común en la implementación común causante de problemas de seguridad es el desbordamiento del buffer. El desbordamiento del buffer se produce de muchas formas –desbordamiento de la pila, desbordamiento al final de un área de un *malloc()* ed dinámico, desbordamiento en áreas de datos estáticas-. Puede ser fácil de detectar (donde un cliente puede poner una cadena de longitud arbitraria a un buffer de tamaño fijo), o incluso muy difícil de detectar (errores de cálculo del tamaño de buffer o un simple *byte* desbordado). O código complejo donde la definición de buffer y distintos usos están aparte.

La solución correcta es ocultar el buffer manipulando el código detrás de una API. Todo buffer asignado, copiado, calculado el tamaño, extensible, etc. está hecho por un simple trozo de código genérico. Las comprobaciones de seguridad del tamaño necesitan ser escritos una vez nada más.

Desde el punto de vista del cliente, esta no es la mejor relación con un buffer. El buffer debe ser encapsulado dentro del buffer API. Todas Las modificaciones para el buffer seguramente se trasladen a la API. Esto suena familiar, esto es porque la implementación de *vsftpd* es muy parecida a la clase String de C++ por si has hecho alguna vez programación orientada a objeto.

Un punto clave de tener la API buffer en lugar de lo otro es que es más difícil abusar de las API y se tenga que usar correctamente o si no, haga un intento y cree una corrupción del buffer de memoria o un escenario desbordante usando simplemente el buffer API.

Desafortunadamente, el uso seguro de cadenas/buffer a través de una API común no se ha escogido, en muy, a pesar de los beneficios que eso trae. Por ello se le da publicidad como una solución.

9. Incidencias y Principales Problemas Detectados

- 1) Un problema que aún no he llegado a entenderlo del todo, es el hecho que durante la instalación del servidor, cuando escribimos el comando `> make install` este no crea el fichero PAM `/etc/pam.d/vsftpd` que permite la autenticación de los usuarios locales, para solucionar esto lo tuvimos que hacer de forma manual, todo eso está explicado en el punto 5.2.6.
- 2) Otro problema detectado es que el servidor no se puede ejecutar directamente sino que necesita el uso del script `/etc/rc.d/init.d/vsftpd`, por tanto no se puede ejecutar el servidor escribiendo el siguiente comando:

```
$ /usr/local/sbin/vsftpd
```

- 3) También no hemos encontrado con el problema de que en el proceso de instalación el sistema no soportaba `tcp_wrappers` por tanto daría fallo activar la opción `tcp_wrappers`.

10. Resumen y Conclusiones

El objetivo del protocolo FTP es la transferencia de datos independientemente de la plataforma, el cuál se basa en una estructura cliente/servidor. La RFC 959[1] determina que el FTP partirá de dos canales, uno que sirve para la transferencia de datos (TCP-port 20) y el otro para la transferencia de control (TCP-port 21). Sobre la conexión de control ambos lados (Cliente/Servidor) intercambian comandos para la iniciación de la transferencia de datos.

Una conexión FTP implica cuatro pasos:

- Autenticación del usuario.
- Establecer el canal de control.
- Establecer el canal de datos.
- Continuar con la conexión.

FTP utiliza la conexión TCP (Transmisión Control Protocol) como protocolo de transmisión que asegura la llegada de los datos al destino. Por lo tanto no existe la necesidad de que FTP trate de buscar paquetes perdidos o buscar errores en los datos recibidos durante la transferencia de datos. FTP simplemente se cerciora de que está llegando cada paquete de datos solamente una vez –sin errores y en la secuencia correcta-.

Además hay dos modos de transferencia:

- Modo ASCII
- Modo Binario

El modo ASCII se está utilizando para la transferencia de los archivos de texto, mientras que el modo binario se está utilizando para la transferencia de programas y datos similares. El usuario no necesita seleccionar el modo de la transferencia específicamente puesto que todos los clientes FTP cambian al tipo reconocido del archivo que se transferirá.

Vsftpd representa un servidor para sistemas operativos como, funciona en plataformas como Linux, BSD, Solaris, Hp-ux-ux e IRIX. Tiene muchas características que faltan en otros servidores FTPs. Algunos de ellos son:

- Gran requisito de seguridad
- Límites del ancho de banda
- Buena escalabilidad

- La posibilidad para crear usuarios virtuales
- Ayuda de IPnG
- Mejor funcionamiento medio
- la posibilidad para asignar IPs virtual
- Alta velocidad

Vsftpd es conocido por ser un *daemon* muy seguro de ftp, que es una de las preocupaciones principales de su revelador, Chris Evans. Antes de comenzar con el desarrollo y el diseño del servidor FTP, la alta seguridad era una de las preocupaciones.

Un ejemplo es el hecho de que *el vsftpd* está funcionado en el modo de *chroot*, que significa que un programa (en este caso *vsftpd*) está asignando un nuevo directorio raíz, y ninguno de los programas tienen acceso a archivos fuera de este directorio. Por tanto el potencial del atacante será aislado del resto del sistema y los daños serán prevenidos.

Como conclusión final podemos decir que *vsftpd* es uno de los mejores servidores, no por su complejidad sino por todo lo contrario, por su sencillez de instalar y administrar, rapidez y su gran estabilidad. También nos ofrece una gran seguridad a través de mecanismos de UNIX, como los *chroot* y las *capabilities*, y el uso del protocolo SSL.

Por tanto podemos asegurar, y muchas críticas también lo confirman, que estamos ante el mejor servidor FTP del mercado.