# Протоколы передачи файлов. Общие сведения.

**Network File System** (**NFS**) — протокол сетевого доступа к файловым системам, первоначально разработан Sun Microsystems в 1984 году.

Эта система позволяет пользователю работать с удаленными данными так же, как с локальными — то есть абсолютно прозрачно, не считая временных задержек.

Протокол NFS использует клиент-серверную модель взаимодействия. В ранних версиях NFS для транспортирования данных использовался UDP-протокол, в современных — используется TCP.

Применение протокола TCP в качестве транспорта позволило решить вопросы совместного доступа прямолинейно, без обходных маневров, однако ценой этому является некоторое снижение производительности по сравнению с UDP.

NFS-клиенты получают доступ к файлам на NFS-сервере путём отправки RPC -запросов на сервер.

Изначальными требованиями при разработке NFS были:

* потенциальная поддержка различных операционных систем (не только [UNIX](https://ru-wiki.ru/wiki/UNIX));
* протокол не должен зависеть от каких-либо определённых аппаратных средств;
* должны быть реализованы простые механизмы восстановления в случае отказов сервера или клиента;
* приложения должны иметь прозрачный доступ к удаленным данным;
* для UNIX-клиентов должна поддерживаться семантика UNIX;
* производительность NFS должна быть сравнима с производительностью локальных дисков;
* реализация не должна быть зависимой от транспортных средств.

Процедура подключения сетевого ресурса средствами NFS называется «экспортированием». Клиент может запросить у сервера список представляемых им экспортируемых ресурсов. Сам сервер NFS, в отличие от, например, сервера SMB, не занимается широковещательной рассылкой списка своих экспортируемых ресурсов.

В зависимости от заданных опций, экспортируемый ресурс может быть смонтирован «только для чтения», можно определить список хостов, которым разрешено монтирование, указать использование защищенного RPC (secureRPC) и пр. Одна из опций определяет способ монтирования: «жесткое» (hard) или «мягкое» (soft).

* При **«жестком» монтировании** клиент будет пытаться смонтировать файловую систему во что бы то ни стало. Если сервер не работает, это приведет к тому, что весь сервис NFS как бы зависнет: процессы, обращающиеся к файловой системе, перейдут в состояние ожидания окончания выполнения запросов RPC. С точки зрения пользовательских процессов файловая система будет выглядеть как очень медленный локальный диск. При возврате сервера в рабочее состояние сервис NFS продолжит функционирование.
* При **«мягком» монтировании** клиент NFS сделает несколько попыток подключиться к серверу. Если сервер не откликается, то система выдает сообщение об ошибке и прекращает попытки произвести монтирование.

С точки зрения безопасности первые реализации NFS были крайне слабы: аутентификация выполнялась, по сути, только по ip-адресу клиента. В версиях NFS 3 и 4 эти вопросы были переработаны путем добавления поддержки списков доступа (ACL), защищенного rpc и ряда других решений.

**Server Message Block (SMB)** — это протокол, предложенный IBM для организации общего доступа к файлам, принтерам, последовательным портам, почтовым ячейкам (mail slots), именованным каналам (named pipes) и API сетевых компьютеров. Протокол SMB может быть использован поверх сетевых протоколов стека TCP/IP, а также поверх ряда других сетевых протоколов.

SMB — это типичный клиент-серверный протокол, который позволяет клиентскому приложению выполнять операции доступа к общему ресурсу (чтение, запись и т.п.) через запросы к серверу.

Продвижение протокола SMB обеспечила корпорация Microsoft, включив его поддержку в свои продукты.

Протокол SMB представляет четыре вида севисов:

* *Управление сессиями*. Создание, поддержание и разрыв логического канала между рабочей станцией и сетевыми ресурсами файлового сервера.
* *Файловый доступ*. Рабочая станция может обратиться к файл-серверу с запросами на выполнение типовых файловых операций (открытие файла, чтение данных и т.п.).
* *Сервис печати*. Рабочая станция может ставить файлы в очередь для печати на сервере и получать информацию об очереди печати.

*Сервис сообщений*. SMB поддерживает простую передачу адресных и широковещательных сообщений по локальной сети.

Механизм защиты протокола SMB состоит из двух уровней защиты: user-level (пользовательский уровень) и share-level (уровень совместно используемого ресурса). Под share (выложенный в сеть ресурс) понимается файл, каталог, принтер, любая услуга, которая может быть доступна клиентам по сети.

Аутентификация на уровне user-level означает, что клиент, который пытается получить доступ к ресурсу на сервере, должен иметь username (имя пользователя) и password (пароль). Если аутентификация прошла успешно, клиент имеет доступ ко всем доступным ресурсам сервера, кроме тех, что с share-level-защитой. Этот уровень защиты даёт возможность системным администраторам конкретно указывать, какие пользователи и группы пользователей имеют доступ к определённым данным.

Аутентификация на уровне share-level означает, что доступ к ресурсу контролируется паролем, установленным конкретно на этот ресурс. В отличие от user-level, этот уровень защиты не требует имя пользователя для аутентификации и не устанавливается никакая уникальность текущего пользователя.

**Common Internet File System** (CIFS) - это открытый стандартный протокол на основе SMB, который обеспечивает доступ к файлам и сервисам на удаленных компьютерах в сетях TCP/IP. В отличие от SMB, основным транспортом для CIFS является протокол TCP.

CIFS — это протокол, основанный на технологии клиент-сервер, который предоставляет клиентским приложениям простой способ для чтения и записи файлов, а также запроса служб у серверных программ в различных типах сетевого окружения.

Клиенты соединяются с сервером, используя протоколы TCP/IP. После того, как соединение установлено, клиенты могут посылать команды серверу, который даёт им доступ к ресурсам, позволяет открывать, читать файлы, писать в файлы и вообще выполнять весь перечень действий, которые можно выполнять с файловой системой.

CIFS определяет серию команд, используемых для передачи информации между сетевыми компьютерами. Запросы на перенаправление отправляют пакеты, предназначенные для удаленных компьютеров в структуре CIFS. Перенаправитель также использует CIFS для выполнения запросов к стеку протоколов локального компьютера. Сообщения CIFS могут быть в целом классифицированы следующим образом:

* Сообщения установления соединения, состоящие из команд, которые запускают и завершают соединение перенаправителя с общим ресурсом на сервере.
* Сообщения о пространстве имен и манипуляциях с файлами, использующиеся перенаправителем для получения доступа к файлам на сервере и для их чтения и записи.
* Сообщения принтера, использующиеся перенаправителем для отправки данных в очередь печати на сервере и получения информации о состоянии очереди печати.
* Различные сообщения, использующиеся перенаправителем для записи в mailslots (клиент-серверный интерфейс) и именованные каналы.

*File Transfer Protocol (*FTP) - сетевой протокол, предназначенный для передачи файлов в компьютерных сетях. Протокол FTP позволяет подключаться к серверам FTP, просматривать содержимое каталогов и загружать файлы с сервера или на сервер, кроме того, возможен режим передачи файлов между серверами.

Этот протокол гарантирует передачу (либо выдачу ошибки) за счёт применения протокола TCP.

Протокол построен на архитектуре «клиент-сервер» и использует разные сетевые соединения для передачи команд и данных между клиентом и сервером. Пользователи FTP могут пройти аутентификацию, передавая логин и пароль открытым текстом, или же, если это разрешено на сервере, они могут подключиться анонимно.

*Trivial File Transfer Protocol (*TFTP) простой протокол передачи файлов. Используется главным образом для первоначальной загрузки бездисковых рабочих станций. TFTP, в отличие от FTP, не содержит возможностей аутентификации (хотя возможна фильтрация по IP-адресу) и основан на транспортном протоколе UDP.

Основное назначение TFTP — обеспечение простоты реализации клиента. В связи с этим он используется для загрузки бездисковых рабочих станций, загрузки обновлений и конфигураций в «умные» сетевые устройства, записи статистики с мини-АТС (CDR) и аппаратных маршрутизаторов/файрволов.

Поскольку протокол не поддерживает аутентификации, единственный метод идентификации клиента — это его сетевой адрес (который может быть подделан).

*Multicast File Transfer Protocol* (MFTP) — сетевой протокол передачи файлов. Файл передаётся одним сервером нескольким клиентам сразу.

Протокол состоит из двух подпротоколов:

* Multicast Control Protocol — для управления подключения и отключения клиентов;
* Multicast Data Protocol — для передачи данных.

*Multisource File Transfer Protocol (*MFTP) ) — сетевой протокол передачи файлов. В MFTP один и тот же файл запрашивается одновременно у нескольких источников, позволяя добиться более эффективного использования пропускной способности канала. Одной из ключевых особенностей MFTP является идентификация файлов не по имени, а по содержимому, путём подсчитывания набора контрольных сумм на основе MD4. Таким образом достигается надёжная идентификация файла вне зависимости от его имени и одновременно эти же суммы используются для контроля за целостностью получаемых данных.

*File eXchange Protocol (***FXP)** — способ передачи файлов между двумя FTP-серверами напрямую, не закачивая их на свой компьютер. При FXP-сессии клиент открывает два FTP-соединения к двум разным серверам, запрашивая файл на первом сервере, указывая в команде PORT IP-адрес второго сервера.

Несомненным преимуществом поддержки стандарта FXP является то, что на конечных пользователей, желающих скопировать файлы с одного FTP-сервера на другой, уже не действует ограничение пропускной способности их собственного интернет-соединения. Нет необходимости скачивать себе файл, чтобы потом загрузить его на другой FTP-сервер. Таким образом, время передачи файлов будет зависеть только от скорости соединения между двумя удаленными FTP-серверами, которая в большинстве случаев заведомо больше «пользовательской».

FXP стал использоваться злоумышленниками для атак на другие серверы: в команде PORT указывается IP-адрес и порт атакуемого сервиса на компьютере жертвы, и командами RETR/STOR производится обращение на этот порт от лица FTP-сервера, а не атакующей машины, что позволяло устраивать масштабные DDoS-атаки с использованием сразу многих FTP-серверов, либо обходить систему безопасности компьютера жертвы, если он полагается только на проверку IP клиента и используемый для атаки FTP-сервер находится в доверенной сети или на шлюзе. В результате сейчас практически все серверы проверяют соответствие IP-адреса, указанного в команде PORT, IP-адресу FTP-клиента и по умолчанию запрещают использование там IP-адресов третьих сторон. Таким образом, использование FXP невозможно при работе с публичными FTP-серверами.

File Transfer Access and Management (FTAM) – протокол прикладного уровня [OSI](https://ru-wiki.ru/wiki/Open_Systems_Interconnection) для *передачи, доступа и управления файлами.*

FTAM попытался объединить собой в один протокол два других — протокол передачи файлов, привычный для Интернета FTP, а также удалённый доступ к открытым файлам, привычный для NFS. Но FTAM не получил широкого распространения.

# Протокол FTP. Принципы работы.

*Функции протокола FTP*

* - решение задач разделения доступа к файлам на удаленных хостах
* - прямое или косвенное использование ресурсов удаленных компьютеров
* - обеспечение независимости клиента от файловых систем удаленных хостов
* - эффективная и надежная передачи данных.

FTP-протокол представляет собой набор команд, которые описывают правила подключения и обмена данными. Отличается от других приложений типа клиент-*сервер* тем, что он устанавливает два соединения между хостами. Одно соединение применяется для передачи данных, другое — для управления информацией (команды и отклики). Разделение команд и передачи *управляющих данных* делает *FTP* более эффективным. Управление соединением использует очень простые правила для связи. Нам нужна для передачи только линия команд или линия откликов. С другой стороны, соединение для данных нуждается в более сложных правилах из-за разнообразия типов данных.

Размер файлов, передаваемых с помощью FTP-протокола, не лимитируется. Предусмотрен также механизм докачки файла, если в процессе передачи произошел обрыв связи.

Главным недостатком FTP-протокола является отсутствие механизмов шифрования данных, что позволяет перехватить начальный трафик и определить с его помощью имя пользователя, а также его пароль подключения к FTP-серверу.

**FTP server** — это сервер, который предусматривает возможность использования протокола передачи файлов. Он обладает определенными особенностями, отличающими его от обычных вэбсерверов, а именно теми, которые в основном также присущи одноименному протоколу и представлены в табличке выше:

* необходима аутентификация пользователей (ввод логина и пароля);
* все операции производятся в рамках текущей сессии;
* возможность осуществления различных действий: загрузка и выгрузка файлов, их переименование и удаление, создание и удаление каталогов и т.д.;
* применяется отдельный канал для каждого соединения;
* поддерживается два варианта передачи: текстовый и двоичный (бинарный), что позволяет передавать файлы различного размера;

**FTP client** — это программа, позволяющая подключиться к удаленному серверу по ФТП и также выполнять на нем необходимые действия с элементами файловой системы. К слову, **клиентом вполне может быть браузер**, в адресную строку которого следует ввести урл, представляющий собой путь (c протоколом FTP) до определенного каталога или файла на удаленном сервере.

Взаимодействие «клиент-сервер» при ФТП-соединении можно представить следующим образом:

1. Пользователь активирует клиентское приложение и соединяется с сервером, введя логин и пароль.
2. Устанавливается управляющее соединение между соответствующими модулями — интерпретаторами протокола со стороны клиента и сервера.
3. Пользователь посредством клиента посылает команды серверу, определяющие различные параметры FTP-соединения (**активный или пассивный режим**, порт, вид передачи данных, их тип), а также директивы для действий, которые юзер намерен осуществить (например, удалить, переименовать, закачать файл и т.д.).
4. После того, как установлены все необходимые параметры, один из участников (к примеру, клиент), являющийся пассивным, становится в режим ожидания открытия соединения на порт, который задан для передачи информации. Затем активный участник открывает соединение и начинает передавать данные по предназначенному для этого каналу.
5. По завершении передачи это соединение закрывается, но управляющий канал между интерпретаторами остается открытым, вследствие чего пользователь в рамках той же сессии может вновь открыть передачу данных.

Кроме стандартного соединения с сервером, предусматривающего ввод данных аутентификации, существует **понятие анонимного FTP**, когда любой пользователь может подключиться к серверу без предоставления персональных данных.

На практике при анонимном ФТП обычно в качестве логина для доступа применяется известное в сети слово «anonymous», а в качестве пароля — адрес электронной почты, который, впрочем, в большинстве случаев не проверяется.

FTP может передавать через соединение для передачи данных следующие типы файлов:

* **ASCII-файл**. Это формат, используемый по умолчанию для трансляции текстовых файлов. Каждый символ закодирован с использованием ASCII-символов. Передатчик преобразует файл из собственного представления в ASCII, и приемник преобразует символы ASCII в собственное представление.
* **EBCDIC-файл**. Если оба конца соединения используют кодирование *EBCDIC*, файл может быть передан с использованием *EBCDIC*-кодирования.
* **Image-файл**. Этот файл по умолчанию — формат для передачи двоичных файлов. Файл посылается как непрерывный поток бит без всякой интерпретации и кодирования. Он в большинстве случаев используется для передачи двоичных файлов, таких как компилированная программа.

Если файл закодирован в ASCII или *EBCDIC*, другие атрибуты должны дополняться, чтобы определить возможность печати файла:

1. Запрещенный для печати. Это формат по умолчанию для передачи текстовых файлов. Файл не содержит "вертикальных" спецификаций для печати. Это означает, что файл не может быть напечатан без предварительной обработки, потому что он не содержит символов, интерпретируемых для вертикального передвижения *печатающей головки*. Этот формат используется для файлов, которые будут накоплены и обработаны позднее.
2. TELNET. В этом формате файл содержит ASCII вертикальные символы, такие, как CR (перевод каретки), LN (перевод строки), NL (новая строка) и VT (вертикальное табулирование). Эти файлы могут быть напечатаны после передачи.

FTP может передавать файл по соединению для передачи данных, используя один из трех следующих режимов передачи:

* **Поточный режим**. Это режим по умолчанию. Данные доставляются от FTP к TCP как непрерывный поток данных. TCP отвечает за разбиение данных на сегменты соответствующего размера. Если данные — просто поток байтов (файловая структура), то не нужно никакого признака окончания файла. Окончание файла в этом случае — это разъединение соединения данных отправителем. Если данные разделены на записи (структура по записи), каждая запись будет иметь однобайтный символ окончания записи (*EOR* — end of record).
* **Блочный режим**. Данные могут быть доставлены от FTP и TCP в блоках. В этом случае блоку предшествует трехбайтный заголовок. Первый байт называется дескриптор блока, следующие два байта определяют размер блока в байтах.
* **Сжатый режим**. Если файл большой, данные могут быть сжаты. Метод сжатия использует нормальное кодирование длины. В этом методе последовательное повторное появление блока данных заменяется одним вхождением и числом повторений. В тексте файла это обычно пробел (пустоты). В двоичном файле нулевые символы обычно сжимаются.

FTP использует соединение управления для того, чтобы установить связь между процессом управления клиента. В течение этой связи команды посылаются от клиента к серверу, и отклики посылаются от сервера к клиенту

Команды, которые посылаются от *FTP*-процесса управления клиента в форме заглавных букв *ASCII*, могут сопровождаться или не сопровождаться аргументом. Мы можем грубо разделить команды на шесть групп: команды доступа, команды управления файлами, команды форматирования данных, команды определения порта, команды передачи файла и прочие команды.

**Команды доступа**. Эти команды позволяют пользователю обращаться к удаленной системе.

|  |  |  |
| --- | --- | --- |
| **Команды** | **Аргументы** | **Описание** |
| USER | ID пользователя | Пользовательская информация |
| PASS | Пароль пользователя | Пароль |
| ACCT | Загруженная учетная запись | Учетная информация |
| REIN |  | Перезапуск |
| QUIT |  | Выход из системы |
| ABOR |  | Прерывание предыдущей команды |

**Команды управления файлом**. Эти команды дают пользователю возможность обращаться к удаленному компьютеру, передвигаться по структуре директории, создавать новые директории, удалять файлы и так далее.

|  |  |  |
| --- | --- | --- |
| **Команды** | **Аргументы** | **Описание** |
| CWD | Имя директории | Изменение другой директории |
| CDUP |  | Изменение вышестоящей директории |
| DELE | Имя файла | Удаление файла |
| LIST | Имя директории | Список поддиректорий и файлов |
| NLIST | Имя директории | Список имен поддиректорий или файлов, не имеющих атрибутов |
| MKD | Имя директории | Создать новую директорию |
| PWD |  | Имя текущей директории на дисплее |
| RMD | Имя директории | Удалить директорию |
| RNER | Имя файла (старое имя) | Идентификатор файла, который будет переименован |
| RNTO | Имя файла (новое имя файла) | *Переименование файла* |
| SMNT | Системное имя файла | Вершина системы |

**Команды форматирования данных**. Эти команды дают пользователю возможность определить данные о структуре, типе файла и *режиме передачи*. Определенный формат затем используется командами передачи файлов.

|  |  |  |
| --- | --- | --- |
| **Команды** | **Аргументы** | **Описание** |
| TYPE | A (ASCII), E (*EBCDIC*), I (IMAGE), N (Nonprint), T(Telnet) | Определяет тип файла, если необходим формат для печати |
| STRU | F (File), R (Record), P (Page) | Определяет организацию данных |
| MODE | S (Stream), B (Block), C (Compressed) | Определяет режим передачи |

**Команды определения порта**. Эти команды определяют номер порта для соединения передачи данных на стороне клиента.

|  |  |  |
| --- | --- | --- |
| **Команды** | **Аргументы** | **Описание** |
| PORT | 6-цифровой идентификатор | Клиент выбирает порт |
| PASV |  | Сервер выбирает порт |

**Команды передачи файла**. Эти команды позволяют передачу файлов.

|  |  |  |
| --- | --- | --- |
| **Команды** | **Аргументы** | **Описание** |
| RETR | Имя файла (ов) | Извлечение файла: файл(ы) передан(ы) от сервера к клиенту |
| STOR | Имя файла (ов) | Накопление файла: файл(ы) передан(ы) от клиента к серверу |
| APPE | Имя файла (ов) | Совпадает с STOR за исключением того, что если файл существует, то данные могут быть прикреплены к нему |
| STOU | Имя файла (ов) | То же самое, что STORE, за исключением того, что имя файла будет уникальным в этой директории; однако существующий файл не должен быть переписан |
| AALLO | Имя файла (ов) | Распределение места для накопления файлов в сервере |
| REST | Имя файла (ов) | Установка отметки в определенной точке данных |
| STAT | Имя файла (ов) | Возврат состояния файла |

**Различные команды**. Эти команды доставляют информацию к пользователю FTP на стороне клиента.

|  |  |  |
| --- | --- | --- |
| **Команды** | **Аргументы** | **Описание** |
| HELP |  | Запрос информации |
| NOOP |  | Проверка, является ли сервер действующим |
| SITE | Команды | Определение сайта заданных команд |
| SYST |  | Запрос об операционной системе, используемой сервером |

Каждая *FTP*-*команда* вызывает по крайней мере один отклик. Отклик имеет две части: номер из трех цифр, следующий за текстом. Числовая часть определяет код; текстовая часть определяет необходимые параметры или дополнительные пояснения.

#### Первая цифра

Первая цифра определяет состояние команды. В этой позиции может быть использована одна из пяти цифр:

* **1yz (положительный предварительный ответ)**. Действие началось. Сервер будет посылать другие отклики перед принятием другой команды.
* **2yz (положительный отклик завершения)**. Действие завершено. Сервер будет принимать другую команду.
* **3yz (положительный промежуточный отклик)**. Команда принята, но нужна дальнейшая информация.
* **4yz (отклик отрицательного переходного завершения)**. Действие не произошло, но ошибка временная. Та же самая команда будет послана позднее.
* **5yz (отклик отрицательного постоянного завершения)**. Команда не принята и должна быть повторена позже.

#### Вторая цифра

Вторая цифра также определяет состояние команды. В этой позиции может быть использована одна из шести цифр:

* **x0z (синтаксис);**
* **x1z (информация);**
* **x2z (подключение);**
* **x3z (идентификация и учет);**
* **x4z (не определено);**
* **x5z (система файлов)**.

#### Третья цифра

Третья цифра обеспечивает дополнительную информацию.

|  |  |
| --- | --- |
| **Коды** | **Описание** |
| **Положительный предварительный ответ** | |
| 120 | Сервис будет вскоре готов |
| 125 | Соединение данных открыто: передача данных вскоре начнется |
| 150 | Состояние файла хорошее (OK). Соединение будет вскоре открыто |
| **Положительный отклик завершения** | |
| 200 | Команда в порядке (OK) |
| 211 | Состояние системы или справочный отклик |
| 212 | Состояние директории |
| 213 | Состояние файла |
| 214 | Справочное сообщение |
| 215 | Обозначение типа системы (операционной системы) |
| 220 | Сервис готов |
| 221 | Сервис закрыт |
| 225 | Соединение для передачи данных открыто |
| 226 | Закрытие соединения для передачи данных |
| 227 | Введение пассивного режима; сервер посылает свой IP-адрес и номер порта |
| 230 | Пользовательский вход корректен (OK) |
| 250 | Запрос файла действует корректно (OK) |
| **Положительный промежуточный отклик** | |
| 331 | Имя пользователя корректно (OK). Необходим пароль |
| 332 | Необходимость учета для регистрации |
| 350 | Действие файла рассматривается: необходимо больше информации |
| **Отклик отрицательного переходного завершения** | |
| 425 | Нельзя открыть соединение для передачи данных |
| 426 | Соединение разъединено: передача прервана |
| 450 | Действие с файлом не производится, файл недоступен |
| 451 | Действие прервано; локальная ошибка |
| 452 | Действие прервано; недостаточно памяти |
| **Отклик отрицательного постоянного завершения** | |
| 500 | Синтаксическая ошибка; неизвестная команда |
| 501 | Синтаксическая ошибка в параметрах или аргументе |
| 502 | Команда невыполнима |
| 503 | Ошибочная последовательность команд |
| 504 | Параметр команды невыполним |
| 530 | Абонент не зарегистрирован |
| 532 | Необходима учетная запись для накапливающего файла |
| 550 | Действие не выполнено: файл недоступен |
| 552 | Запрос на прерывание действия; превышена выделенная память |
| 553 | Запрос на прекращение действия; не разрешенное имя файла |

FTP не разрабатывался как защищённый (особенно по нынешним меркам) протокол и имеет многочисленные уязвимости в защите. В мае 1999 авторы RFC 2577 свели уязвимости в следующий список проблем:

* Скрытые атаки (bounce attacks)
* Спуф-атаки (spoof attacks)
* Атаки методом грубой силы (brute force attacks)
* Перехват пакетов, сниффинг (packet capture, sniffing)
* Защита имени пользователя
* Захват портов (port stealing)

FTP не может зашифровать свой трафик, все передачи — открытый текст, поэтому имена пользователей, пароли, команды и данные могут быть прочитаны кем угодно, способным перехватить пакет по сети.

# Безопасные версии протоколов передачи файлов.

1.**FTPS** (FTP + SSL) — фактически это расширение стандартного протокола передачи файлов, которое обеспечено криптографическим протоколом SSL (Secure Sockets Layer — уровень защищенных сокетов). На сегодняшний день более передовым его аналогом является TLS (Transport Layer Security — защита транспортного уровня). При этом есть два метода предоставления безопасности:

* **Неявный** является устаревшим и использует стандартный протокол,

требующий применения SSL или TLS, которые могут обеспечить шифрование информации. При таком методе обязательно нужно использовать порты, отличные от обычных, что создает неудобства, поскольку нарушается совместимость клиентов и серверов, не поддерживающих FTPS.

* **Явный** — намного более удобный, так как применяет команды стандартного

протокола FTP, но при отправке ответа зашифровывает информацию, что позволяет сохранить совместимость, поскольку в этом случае применяются одни и те же порты как для FTPS, так и для FTP. При этом для шифрования данных клиентом отправляется команда «AUTH TLS» или «AUTH SSL».

С появлением FTPS (File Transfer Protocol + SSL) интернет-соединение между сервером и клиентом стало намного безопаснее. Если же хакерские атаки и перехватывали какие-то фрагменты кода, файла или же целые файлы, то они ничего из этих данных не могли извлечь. Все очень просто, но гениально: перед тем как файл отправится клиенту на компьютер, он шифруется протоколом SSL. Если же открыть подобный файл, который зашифрован, то, кроме неразборчивых знаков, вы ничего там не увидите. Для понимания принципа работы рассмотрим, как взаимодействует сервер и клиент, использующие протокол FTPS: Клиент (например, компьютер) отправляет запрос на шифрование данных, которые ему нужны, после чего выполняются некоторые операции и отправляется запрос CSR на сервер. Как только запрос был доставлен на сервер, определяется то, каким образом будет шифроваться файл, а именно его алгоритм. После чего клиенту отправляется сертификат SSL со специальным ключом, чтобы выполнить следующий шаг. При помощи ключа, полученного от сервера, клиент имеет возможность прочитать информацию, которую содержит сертификат (в нем находятся инструкции), после чего осуществляется проверка на совпадение ключей. Когда ключи не совпадают или же файлы повреждены, то процедура отменяется и сервер получает код ошибки. В случае успешной сверки ключей происходит операция передачи зашифрованных данных.

Расшифровать полученные данные можно, только используя этот ключ, иного способа не существует. Данный вариант гарантирует максимальную безопасность данных, в отличие от обычного FTP.

Область использования

При неявном режиме, шифруется вся сессия FTPS. Явный режим отличается тем, что клиент полностью контролирует тот трафик, который должен быть зашифрован. Включение и отключение режима шифрования как для контрольного соединения, так и для соединения передачи данных, можно осуществить в любое время. Единственное ограничение состоит в том, что сервер может отклонять команды в зависимости от собственной стратегии обеспечения защиты данных.

#### Защищенное контрольное соединение

Перейти в режим защищенного контрольного соединения можно с помощью команд - AUTH TLS и AUTH SSL. В таком режиме все команды между сервером и клиентом будут зашифрованы. Как правило, рекомендуется переходить в такое состояние перед аутентификацией и авторизацией пользователя, чтобы избежать перехвата третьими лицами имени пользователя или пароля.

#### Защищенное соединение передачи данных

Перейти в режим защищенного соединения передачи данных можно с помощью команды PROT. Она не включена по умолчанию когда используется команда предоставления безопасности AUTH TLS. В таком режиме, все соединения для обмена данными между клиентом и сервером будут зашифрованы. Клиент может изменить режим передачи данных в любой момент при помощи команды CDC.

Плюсы FTPS:

* весьма популярен при создании серверов;
* сообщение никак не может прочитаться человеком из-за особого типа шифрования;
* автоматически поддерживается во многих цепях коммуникации, использующих интернет-соединение.

2.**SFTP** (SSH FTP) — протокол прикладного уровня для передачи файлов, который работает поверх безопасного канала. Если FTPS является просто расширением FTP, то SFTP это отдельный и никак не связанный с FTP протокол, который снабжен SSH (Secure Shell — безопасная оболочка).

Главное, в чем заключается его отличие от стандартного FTP и FTPS, это то, что SFTP шифрует абсолютно все команды, имена пользователей, пароли и другую конфиденциальную информацию. Так как это совершенно другая конфигурация, клиенты FTP (FTPS) не могут соединиться с SFTP-сервером.

Плюсы SFTP:

* возможно исключительно одно подключение;
* соединение будет установлено всегда;
* каталоги достаточно хорошо проработаны, поэтому могут быть читабельными для машин (компьютеров);
* высокая функциональность.

3.**FTP через SSH** — производит обычную FTP-сессию через SSH-туннель, то есть этот вариант нельзя путать с SFTP, который является отдельным независимым протоколом. Этот метод нельзя назвать стопроцентно безопасным. Дело в том, что если несколько SSH-клиентов устанавливают туннель для управляющего канала, то защищенным окажется именно этот канал. При передаче же данных клиентское программное обеспечение откроет новые TCP-соединения, которые будут находиться уже вне воздействия защитной оболочки SSH.