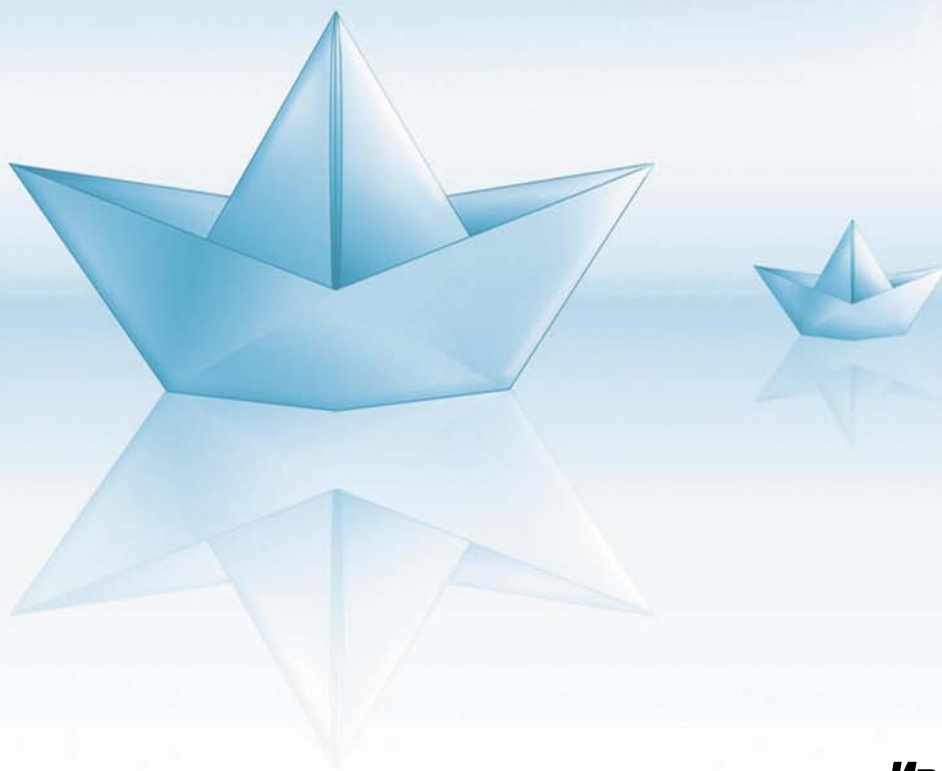


Безопасный FTP – это легко!



Иван Максимов

Именно такой лозунг у разработчиков Pure-FTPd, основанного на Troll-FTPd FTP-сервере. Простой в эксплуатации, легкий при установке и настройке, быстрый – так окрестили этот сервер пользователи, верно ли все это?

Этот FTP-сервер можно скачать с официальной страницы [1], но гораздо удобнее воспользоваться системой пакетов/портов в вашей *nix-операционной системе, благо данная программа включена в репозитории почти всех ОС данного семейства.

Из основных функций рассматриваемого FTP-сервера стоит выделить: работу с СУБД MySQL и PostgreSQL, сервером LDAP, возможностью PAM-аутентификации, экспериментальной поддержкой TLS/SSL-шифрования и возможность мониторинга и журналирования. К полезным дополнительным функциям стоит отнести возможность «запираания» пользователей (chroot) в домашних директориях, смену кодировки на лету (опять же экспериментально), собственную БД (puredb), воз-

можность ограничения пользователей по трафику, времени, скорости, IP-адресам, квоты и многое другое, о чем можно прочесть в штатной документации. Пожалуй, нужно сразу заметить, что документация на официальном сайте достаточно скудна, тогда как сам дистрибутив снабжен более полным описанием.

Все дальнейшие настройки мы будем выполнять в ОС Debian Linux 4, но большинство действий, кроме базовых, подготовительных работ с репозитарием, будут работать и в других подобных операционных системах.

Установка Pure-FTPd

Перед началом работы определим, какие функции необходимы в FTP-сервере. Во-первых, нужна аутентификация пользователей, она будет проходить

с помощью БД puredb. Во-вторых, шифрование при передаче данных, так как пользователи работают через защищенные сети. В-третьих, файловый сервер хранит данные в кодировке UTF8, для клиентов нужна будет перекодировка на лету из UTF8 в CP1251. В-четвертых, конечно же, нужны мониторинг и журналирование.

Итак, определившись с требованиями, начнем. Установку любого программного обеспечения всегда стоит начинать с обновления системы пакетов/портов:

```
# aptitude update
# aptitude upgrade
```

Далее найдем необходимый нам пакет:

```
# aptitude search pure-ftpd
```

```
pure-ftpd - Pure-FTPd FTP server
pure-ftpd-common - Pure-FTPd FTP server (Common Files)
pure-ftpd-ldap - Pure-FTPd FTP server with LDAP user authentication
pure-ftpd-mysql - Pure-FTPd FTP server with MySQL user authentication
pure-ftpd-postgresql - Pure-FTPd FTP server with PostgreSQL user authentication
```

К сожалению, готовый откомпилированный Pure-FTPd не содержит нужных нам функций, поэтому загрузим исходные коды нашего FTP-сервера для перенастройки:

```
# apt-get source pure-ftpd
```

Исходные коды получены, но перед тем как перейти в каталог и начать конфигурирование, установим дополнительные пакеты, такие как `openssl` и библиотеки к нему.

```
# aptitude install openssl
# aptitude install libssl-dev
```

Создадим самоподписанный цифровой сертификат по алгоритму RSA с 1024-битным ключом:

```
# openssl req -x509 -nodes -newkey rsa:1024 \
    -keyout /etc/ssl/private/pure-ftpd.pem \
    -out /etc/ssl/private/pure-ftpd.pem
```

Теперь перейдем в каталог с исходными кодами Pure-FTPd и запустим конфигуратор с необходимыми нам ключами:

```
# ./configure --with-rfc2640 --with-puredb --with-tls \
--with-certfile=/etc/ssl/private/pure-ftpd.pem \
--with-ftpwho --with-altlog
```

Рассмотрим подробнее используемые ключи:

- **--with-rfc2640** – использовать спецификацию rfc2640 для работы с кодировками «на лету»;
- **--with-puredb** – для аутентификации и авторизации пользователей использовать собственную БД (puredb);
- **--with-tls** – включаем поддержку SSL/TLS;
- **--with-certfile** – использовать сертификат по указанному пути;
- **--with-ftpwho** – опция, необходимая для работы утилиты мониторинга pure-ftpwho;
- **--with-altlog** – используем журналирование.

Если все прошло благополучно, в конце операции нам предложат посетить сайт разработчиков и подписаться на почтовую рассылку, если нет – смотрим ошибки, возможно, не все зависимости удовлетворены, или есть проблемы с правами доступа.

Откомпилируем пакет командами:

```
# make & make install
```

Настройка Pure-FTPd

Итак начнем настройку. Цифровой сертификат мы уже создали, теперь пришла очередь `puredb`, создадим ее и заполним пользовательскими учетными записями.

Перво-наперво создадим группу:

```
# groupadd ftpgroup
```

Далее учетную запись реального системного пользо-

вателя, от имени которой (и с правами которой) в дальнейшем будут работать виртуальные пользователи:

```
# useradd -m -g ftpgroup -d /dev/null -s /etc ftpuser
```

где параметр указывает:

- **-m** – автоматически создавать домашнюю папку пользователя;
- **-g** – использовать gid (group identifier) указанной группы, в нашем случае это вышесозданная ftpgroup;
- **-d** – указывает на домашнюю папку пользователя, в нашем случае – в никуда (/dev/null), но можно и так – /sbin/nologin;
- **-s** – указывает на используемую пользователем оболочку (shell).

И последнее, конечно же, указываем имя пользователя – ftpuser.

Проверим, создан ли пользователь:

```
# getent passwd ftpuser
```

```
ftpuser:x:1012:1012::/dev/null:/etc
```

Затем, если мы хотим импортировать учетные записи системных пользователей в список виртуальных, то можно воспользоваться утилитой `pure-pwconvert`, но мы будем создавать учетную базу пользователей «с нуля». Для этого воспользуемся другой утилитой – `pure-pw`.

Утилита содержит множество параметров для полного управления пользовательским бюджетом, но мы воспользуемся только основным и необходимым минимумом.

```
# pure-pw useradd max -u ftpuser -d /home/ftpusers/max
```

где параметр:

- **useradd** – добавить пользователя, в нашем случае `max`;
- **-u** – использовать `uid` реального пользователя;
- **-d** – домашняя папка пользователя.

Данная команда создаст учетную запись виртуального пользователя, после ввода нам предложат ввести для нее пароль.

Проверим это:

```
# cat /etc/pureftpd.passwd
```

```
max:$1$d7Op62.0$DfYS500mjbflEwMTWPGTF.:1012:1012::/home/ftusers/  
max/./::::::::::::
```

На первом месте указано имя пользователя, далее – md5 хэш-пароля (md5 всегда легко узнается по характерному началу хэша «\$1\$» и последующим 31 символам), gid, uid пользователя и в конце путь до пользовательской папки. В данном примере другие параметры отсутствуют, но легко заметить по множественным символам-разделителям «:», что параметров может быть гораздо больше, это именно так, воспользуемся утилитой `pure-pw`:

```
# pure-pw show max
```

```

Login      : max
Password   : $1$d7Op62.0$DfYS500mjb1eWmTWEgTF.
UID        : 1012 (ftpuser)
GID        : 1012 (ftpgroup)
Directory  : /home/ftpusers/max/.
Full name  :
Download bandwidth : 0 Kb (unlimited)
Upload  bandwidth : 0 Kb (unlimited)
Max files   : 0 (unlimited)
Max size    : 0 Mb (unlimited)
Ratio       : 0:0 (unlimited:unlimited)
Allowed local IPs :
Denied local IPs :
Allowed client IPs :
Denied client IPs :
Time restrictions : 0000-0000 (unlimited)
Max sim sessions : 0 (unlimited)

```

Хорошо видно, что при желании мы можем дополнительно задать и другие параметры пользовательской учетной записи, это упоминалось в самом начале, теперь это видно наглядно, в данном примере, пользователь свободен от многих ограничений. Возможно, кто-то задаст вопрос: откуда в пути к домашней папке появилась «./»? Это означает, что учетная запись пользователя «заперта» chroot.

После того как будут созданы (или импортированы) все пользователи, необходимо создать базу данных puredb на основе записей в файле /etc/pureftpd.passwd, это делается командой:

```
# pure-pw mkdb
```

Собственно, на этом базовая настройка FTP-сервера pure-ftpd закончена, запустим сервер:

```
# /usr/local/sbin/pure-ftpd -4 -A -B -E -j -U 022:022 -j
-O clf:/var/log/pureftpd.log -j
--fscharset=utf8 --clientcharset=cp1251 -j
-lpuredb:/etc/pureftpd.pdb --tls=2
```

Вот такой, казалось бы, громоздкой командой производится старт сервера, рассмотрим подробнее используемые ключи:

- **-4** – использовать только протокол IPv4;
- **-A** – «запирать» всех пользователей в своих домашних папках (chroot);
- **-B** – запускать сервер как службу (демон);
- **-E** – запретить работу анонимным пользователям;
- **-j** – разрешить автоматическое создание домашних директорий пользователей (стоит использовать лишь на первых порах);
- **-U 022:022** – все пользователи работают с указанной маской;
- **-O clf:/var/log/pureftpd.log** – журналировать все действия демона в формате, подобном log-файлу веб-сервера Apache (IP:пользователь:время:ресурс);
- **--fscharset=utf8 --clientcharset=cp1251** – задаем перекодировку на лету с локальной UTF8 на «классическую» клиентскую cp1251;
- **-lpuredb:/etc/pureftpd.pdb** – для авторизации пользователей использовать БД формата puredb, расположенную по указанному адресу;
- **--tls=2** – разрешать только зашифрованные соединения.

Собственно, на этом можно было бы и закончить, сервер работает, и, вооружившись соответствующим клиентом (поддерживающим шифрование данных), можно приглашать пользователей, но если настройка производится на реальном внешнем сервере, а не в локальном, для тестирования, скорее всего клиенты не получат доступ к FTP-серверу. Рассмотрим некоторые замечания...

Возможные трудности

Первая трудность, которая может возникнуть, состоит в том, что FTP с поддержкой шифрования работает только в пассивном режиме, при этом firewall на сервере (где установлен pure-ftpd), возможно, будет блокировать соединения по необходимому для работы портам. Прочитав врезку о различиях работы активного и пассивного FTP, отследим и внесем необходимые изменения в firewall сервера для нормальной работы pure-ftpd в данном режиме.

На практике все будет происходить так: клиенты проходят процедуру создания защищенного соединения (TLS), используя цифровой сертификат, затем – аутентификации, но при попытке получить доступ в домашние директории клиенты будут «виснуть». Следующая команда и ее вывод наглядно покажут нам причину «подвисания»:

```
# netstat -apn | grep ftp
```

```

tcp      0  0  85.250.250.250:43017  0.0.0.0:*        LISTEN    7698/pure-ftpd (IDL
tcp      0  0  0.0.0.0:21            0.0.0.0:*        LISTEN    7689/pure-ftpd (SER
tcp      0  0  85.250.250.250:4222   0.0.0.0:*        LISTEN    7707/pure-ftpd (IDL
tcp      0  0  85.250.250.250:65471  0.0.0.0:*        LISTEN    7699/pure-ftpd (IDL
tcp      0  0  85.250.250.250:21     85.251.251.251:1103 ESTABLISHED 7707/pure-ftpd (IDL
unix     2  [ ]  DGRAM                97031             7707/pure-ftpd (IDL
unix     2  [ ]  DGRAM                96948             7699/pure-ftpd (IDL
unix     2  [ ]  DGRAM                96932             7698/pure-ftpd (IDL
unix     2  [ ]  DGRAM                96837             7689/pure-ftpd (SER

```

Хорошо видно, что на стороне нашего сервера с адресом 85.250.250.250 открыто 3 соединения на портах выше 1024 (т.е. пассивного FTP), как и со стороны клиента, последний (85.251.251.251) – это шлюз, из сети которого работают пользователи.

Для решения данной задачи запустим наш Pure-FTPd-сервер с еще одним дополнительным параметром:

```
# /usr/local/sbin/pure-ftpd -4 -A -B -E -U 022:022 -j
-p 50000:51000 -O clf:/var/log/pureftpd.log -j
--fscharset=utf8 --clientcharset=cp1251 -j
-lpuredb:/etc/pureftpd.pdb --tls=2
```

Ключ «p» (--forcepassiveip) принудительно задает диапазон портов при работе FTP-сервера в пассивном режиме.

Добавим в наш firewall правило, разрешающее входящие TCP-соединения на портах 50000-51000. На Debian Linux 4 (firewall – iptables) правило будет иметь подобный вид:

```
# iptables -I INPUT -p tcp --dport 50000:51000 -j ACCEPT
```

Проверим, работает ли правило:

```
# iptables -L
```

```

Chain INPUT (policy DROP)
target    prot opt source                destination
ACCEPT    tcp  --  anywhere              anywhere            tcp dpts: 50000:51000

```

Остальные цепочки не отображены из-за неактуальности.

Теперь, вновь воспользовавшись утилитой netstat, можно будет увидеть, что работа ведется по указанным портам корректно.

Вторая проблема. Еще один firewall, который нам, возможно, нужно будет поправить, – шлюз (шлюзы), через которые работают клиенты. В этом может быть необходимость, если на нем жестко указаны порты, используемые клиентами для доступа в глобальную сеть. Зачастую подобные правила вводятся для ограничения особых пользователей из соображений безопасности, например бухгалтерии, где открытие лишних портов крайне нецелесообразно. Все это, конечно, возможно при условии, что имеется доступ к шлюзовому серверу клиентов, иначе им придется самостоятельно связываться с провайдером или местным системным администратором и передавать просьбу об открытии необходимых портов.

Возможно, кто-то заметит, что обычно при работе клиентов из-за NAT ответные соединения (established) разрешены всегда, что подобные проблемы возникают довольно редко и связаны скорее с изначально «неграмотной» настройкой шлюза клиентов, но, как показала практика, подобные ситуации все же имеют место быть, и, к сожалению, далеко не как исключения.

Итак, если необходимо сохранить прежние параметры клиентов, то добавим разрешение на открытие портов для подключения к пассивному FTP:

```
iptables -I FORWARD -i eth0 -s 192.168.0.20 -o eth1 -j ACCEPT
```

или разрешим данному клиенту прохождение пакетов по всем портам:

```
iptables -I FORWARD -i eth0 -s 192.168.0.20 -o eth1 -j ACCEPT
```

где в обоих случаях eth0 – локальный интерфейс, а eth1 – соответственно внешний.

Пожалуй, еще одно затруднение может возникнуть с клиентами. В рассматриваемом примере мы разрешаем клиентам присоединяться к серверу только с использованием шифрования, опция --tls=2, к сожалению, многие стандартные FTP-клиенты «не понимают» этого. Самое простое решение заключается в изменении параметров запуска Pure-FTPd с ключом --tls=1, разрешающим как зашифрованные, так и не зашифрованные соединения, но это не совсем правильный выход из сложившейся ситуации. В сети можно найти много FTP-клиентов под практически любые операционные системы, поддерживающие TLS/SSL-шифрование. Список подобных программ можно найти по адресу [2]. Вот наиболее функциональные: FTP Commander, Directory Opus, BitKinex, но из списка по ссылке больше всех приглянулась клиентам программа FileZilla: русскоязычная, с понятным пользовательским ин-

Пассивный и активный FTP

Как передаются данные через протокол FTP? Сервер открывает одиночное соединение на 21 порту, называемое сессией управления (FTP control session), далее при передаче определенных данных (аутентификация, тип соединения и т. д.) открывается еще один дополнительный порт. Подобные соединения бывают пассивными и активными.

В чем разница? При активном соединении клиент передает серверу номер порта и IP-адрес для соединения, далее клиент создает и слушает порт в диапазоне 1024-65535, а сервер подключается к нему, передавая данные с 20 порта (FTP-Data). С пассивным FTP все наоборот. При соединении клиент запрашивает IP-адрес и порт для подключения у сервера, затем сервер

создает порт в диапазоне 1024-65535, потом клиент на своей стороне открывает порт в диапазоне 1024-65535, далее идет передача данных. На рисунке изображена схема работы активного FTP; от пассивного она отличается только тем, что на стороне сервера вместо 20 порта используется диапазон 1024-65535. В зависимости от типа соединения в firewall клиента и сервера необходимо вносить соответствующие изменения. Этот вопрос обычно очень хорошо освещен в штатной документации любого *nix firewall, а также на сайте стандартов RFC [3]. Кстати говоря, первый документ, описывающий протокол FTP, датирован 16 апреля 1971 года, задержите внимание здесь и задумайтесь над датой, тогда еще не было Интернета, была только его родоначальница, сеть ARPANET.

терфейсом, портируемая (установленную программу можно записать на flash-память и держать всегда при себе), с автоматическим обновлением и, что немаловажно, свободно распространяемая.

Собственно, вышеперечисленные трудности, связанные с настройкой сервера Pure-FTPd, в целом – все. Если что-то не так работает, либо не собирается/транслируется/компилируется, «запускайте» клиентов, производите мониторинг системы и сетевых соединений, проверяйте права доступа и, конечно же, смотрите сообщения сервера в системных журналах в директории /var/log/.

Дополнительные функции

Пожалуй, первый вопрос, который чаще всего можно встретить на форумах о Pure-FTPd, звучит примерно так: можно ли осуществлять настройку FTP-сервера через «conf»-файлы, а не через командную строку? Ответ: конечно же можно. Рассмотрим пример запуска Pure-FTPd с использованием конфигурационного файла и заодно получим ответ на второй, вполне логичный вопрос: почему изначально настройка производилась не через «conf»-файл?

В архиве с исходными кодами находится каталог «configuration-file», в котором содержится файл-шаблон pure-ftpd.conf, скопируем его в папку /etc/. Далее в случае необходимости отредактируем его, но оставим пока все как есть. В том же каталоге сделаем исполняемый скрипт «pure-config.pl» командой:

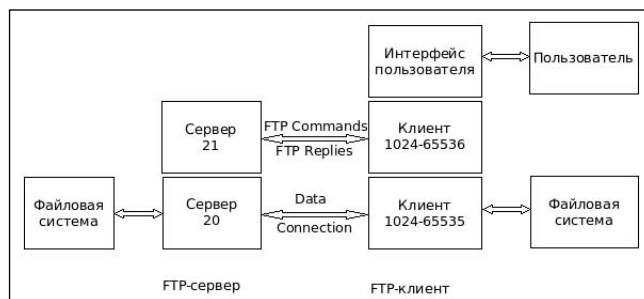


Схема работы активного FTP


```
# chmod 755 pure-config.pl
```

В каталоге находится еще один скрипт – «pure-config.py», отличается он лишь тем, что написан на языке Python, а не Perl, созданный для тех, кто не использует последний в своей работе.

Идем дальше, выполним команду:

```
# pure-config.pl /etc/pure-ftpd.conf
```

```
Running: /usr/local/sbin/pure-ftpd -A -c50 -B -C8 -D -fftp -H
-I15 -L2000:8 -m4 -s -U133:022 -u100 -k99 -Z
```

Вот и ответ на второй, логичный вопрос. Скрипт читает параметры из конфигурационного файла /etc/pure-ftpd.conf и передает их в консоль – pure-ftpd, запуская FTP-сервер. То есть, «идя этим путем», мы выполним лишние действия (хоть и немногочисленные), а получим тот же результат.

Если все же необходимо зафиксировать параметры запуска FTP-сервера, эффективнее будет создать скрипт подобного вида:

```
#!/bin/bash
/usr/local/sbin/pure-ftpd -4 -A -B -E -U 022:022 -J
-p 50000:51000 -O clf:/var/log/pureftpd.log -J
--fscharset=utf8 --clientcharset=cp1251 -J
-lpuredb:/etc/pureftpd.pdb -tls=2
```

Чуть ниже мы воспользуемся этим маленьким скриптом, а пока перейдем к следующей теме.

Второй популярный вопрос о Pure-FTPd касается автоматизации запуска сервера при старте системы. Самым простым решением будет запись параметров запуска в файл /etc/rc.local, но не всегда это возможно, да и более правильно будет добавить скрипты запуска на соответствующие уровни загрузки (SysVinit). Добавим на 3,4 уровни скрипты старта FTP-сервера.

Поместим ранее написанный маленький скрипт в папку /etc/inid.d/ назвав, допустим, ftpes и выполним команду (точка в конце обязательна):

```
# update-rc.d ftpes start 20 3 4 .
```

```
Adding system startup for /etc/inid.d/ftpes ...
/etc/rc3.d/S20ftpes -> ../init.d/ftpes
/etc/rc4.d/S20ftpes -> ../init.d/ftpes
```

По выводу команды видно, что были добавлены скрипты, запускающие FTP-сервер на указанных уровнях. Нужно заметить, что данный скрипт не сможет обрабатывать команды start/stop/restart/status, он всего лишь будет запускать FTP-сервер, если есть необходимость в подобном скрипте, его можно либо написать самостоятельно, либо взять из репозитория вместе с стандартным pureftpd. Если понадобится удалить скрипты запуска из всех rcX.d каталогов, выполним команду:

```
# update-rc.d -f ftpes remove
```

Что еще может понадобиться в работе FTP-сервера? Полезным может быть полное управление пользовательскими бюджетами, об этом уже говорилось в самом начале, теперь подробнее.

При компиляции можно задать ключи:

- **--with-peruserlimits** – позволяет задавать ограничения для каждого пользователя;
- **--with-throttling** – ограничение по скорости работы пользователей;
- **--with-ratios** – ограничения по соотношению download/upload для пользователей;
- **--with-quotas** – дисковые квоты пользователей;

Что это нам даст? Все функции, описанные в выводе команды:

```
# pure-pw --help
```

```
pure-pw useradd <login> [-f <passwd file>] -u <uid> [-g <gid>]
-D/-d <home directory> [-c <gecos>]
[-t <download bandwidth>] [-T <upload bandwidth>]
[-n <max number of files>] [-N <max Mbytes>]
[-q <upload ratio>] [-Q <download ratio>]
[-r <allow client host>[/<mask>]][,<allow client host>[/<mask>]]...]
[-R <deny client host>[/<mask>]][,<deny client host>[/<mask>]]...]
[-i <allow local host>[/<mask>]][,<allow local host>[/<mask>]]...]
[-I <deny local host>[/<mask>]][,<deny local host>[/<mask>]]...]
[-y <max number of concurrent sessions>]
[-z <chhmm>[-chhmm]] [-m]
```

Не нужно перевода, чтобы понять, данного набора опций хватит с излишком для любых административных задач при работе с пользователями.

На этом можно и остановиться, осталось еще много интересных решений на базе сервера Pure-FTPd и много не рассмотренных функций, но последние описаны в документации, а «интересные решения» обсуждаются на форумах в Интернете почти каждый день и при соответствующей задаче их всегда можно изучить.

Итоги

Что можно отметить и дополнить? Рассматриваемый FTP-сервер оказался простым в эксплуатации, поддерживающим множество опций, касающихся информационной безопасности, и не только. Конечно же, у него есть конкуренты, имеющие как свои плюсы, так и минусы.

У более известного сервера ProFTPD есть проблемы с перекодировкой на лету, они, конечно же, решаются с помощью сторонних патчей (в некоторых дистрибутивах он уже пропатченный), но все же... да и его настройку простой назвать трудно.

У vsftpd (Very Secure FTP) много плюсов, включая большой набор опций, касающихся безопасности сервера в целом, но в то же время инструментов, относящихся к пользовательским бюджетам, там меньше.

Блуждая по сети, можно найти много «за» и «против», «плюсов» и «минусов» в различных FTP-серверах, но все же Pure-FTPd – простое решение по организации удаленного и безопасного доступа к документам организации.

Главное, чтобы всегда был выбор. Успехов в работе! 🌐

1. <http://www.pureftpd.org> – официальный сайт сервера Pure-FTPd.
2. http://en.wikipedia.org/wiki/List_of_FTP_clients – список FTP-клиентов поддерживающих шифрование при передаче данных.
3. <http://rfc-editor.org> – сайт, посвященный документации серии RFC (Request for Comments).