Нестандартная настройка proftpd

Введение

Часто в резюме, в графе профессиональные навыки, встречаю упоминания FTP серверов, например:

```
Навыки:
Опыт работы в компании с более 150 пользователей
Установка, настройка и администрирование Debian, Ubuntu
 ...
 ...
 ...
Администрирование ftp сервера: vsftpd
```

Не знаю как кто относится к подобным фразам, но у меня они вызывают лишь улыбку. Но в случае с profptd, особенно зная его широкий функционал, я бы сказал, что его смело можно указывать в резюме, если речь конечно не о типовой конфигурации - тривиальная настройка ftp сервера.

Итак, что же такое proftpd? Как я уже говорил выше, пожалуй один из самых «мощных» и продвинутых ftp серверов. Правда у этой медали есть и обратная сторона - в нем чаще, чем в других ftp серверах находят уязвимости. Поэтому, если вы используете его на боевых серверах, то необходимо следить за обновлениями.

ProFTPd — FTP-сервер для Linux и UNIX-подобных операционных систем. ProFTPd использует один конфигурационный файл proftpd.conf. Сам сервер может быть настроен для работы нескольких виртуальных хостов, также поддерживает chroot и IPv6. Может быть запущен в виде отдельного сервера (демона) или в составе суперсервера xinetd.

Ниже перечислю наиболее значимый функционал

- проект активно развивается и поддерживается
- поддержка файлов конфигурации «.ftpaccess», некий аналог .htaccess из apache.
- поддержка виртуальных хостов, на момент написания статьи существует только ip/port based vhost support. Но в 1.3.5 уже начали реализовывать поддержку команды HOST, которая описана в RFC 7151
- отдельная конфигурация для создания анонимного ftp доступа.
- возможность запуска в режиме домена, так и из под inetd/xinetd, в зависимости загрузки системы
- анонимный FTP корень не требует никакие специальных прав или структуры каталогов
- нет поддержки команды SITE EXEC, которая представляет угрозу в безопасности
- поддержка скрытия папок и файлов в зависимости от прав пользователя
- запуск из под непривилегированного пользователя, как правило - nobody/nobody. Для снижения риска нанесения вреда, в случае обнаружения уязвимости
- логирование и поддержка utmp/wtmp
- Поддержка модулей, за счет которых функционал сервера может значительно расширяться

- доступны следующие модули: MySQL/PQSQL, LDAP, SSL/TLS, SFTP, PAM, RADIUS, квоты и многое другое.

Итак, почему я назвал статью нестандартная настройка? Потому что у нас ftp сервер по сути будет выполнять 3 функции

1. ftp сервер на стандартном 21 порту, будет обслуживать программу для снимков экрана. Так как программа была написана очень давно и понимает только ftp и только на стандартному порту.
2. sftp сервер с поддержкой системных пользователей. Используется для разработчиков.
3. ftps сервер с аутентификацией в LDAP. Используется front-end разработчиками для проверки html/ssi.

У нас будет 4 виртуальных хоста со следующими настройками:

1. ftp, port 21, VirtualUser (plain text file)
2. sftp, port 2121, PAM
3. ftpes, port 3333, LDAP
4. sftp, port 4444, VirtualUser (public key only)

По возможности, я стараюсь избегать использование стандартного FTP протокола везде, где только можно. Но иногда, в силу ряда причин, часто не зависящих от меня, полностью от него отказаться не получается.

Подготовка и настройка ProFTPD

Установить сам proftpd вы можете любым удобным для вас способом, но в CentOS 6 к сожалению в репозитариях идет старая версия 1.3.3g-4.el6, к тому же в ней нет поддержки TLS. Но это не проблема, в составе тарбола самого proftpd идет spec файл, так что все, что нам нужно - это собрать пакет с нужными нам опциями.

```
# wget ftp://ftp.proftpd.org/distrib/source/proftpd-1.3.5a.tar.gz
# rpmbuild -ta --target=x86_64 --with ldap --with mysql --with pcre --with
ssl --with wrap proftpd-1.3.5.tar.gz
Building target platforms: x86_64
Building for target x86_64
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.HmqOvI
+ umask 022
+ cd /root/rpmbuild/BUILD
+ cd /root/rpmbuild/BUILD
+ rm -rf proftpd-1.3.4e
+ /bin/tar -xf -
...
...
...
Wrote: /root/rpmbuild/SRPMS/proftpd-1.3.5-0.1.a.el6.src.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/proftpd-1.3.5-0.1.a.el6.x86_64.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/proftpd-ldap-1.3.5-0.1.a.el6.x86_64.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/proftpd-mysql-1.3.5-0.1.a.el6.x86_64.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/proftpd-wrap-1.3.5-0.1.a.el6.x86_64.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/proftpd-devel-1.3.5-0.1.a.el6.x86_64.rpm
```

```
Wrote: /root/rpmbuild/RPMS/x86_64/proftpd-utils-1.3.5-0.1.a.el6.x86_64.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/proftpd-
debuginfo-1.3.5-0.1.a.el6.x86_64.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.1ngMCo
+ umask 022
+ cd /root/rpmbuild/BUILD
+ cd proftpd-1.3.5a
+ rm -rf /root/rpmbuild/BUILDROOT/proftpd-1.3.5-0.1.a.el6.x86_64
+ rm -rf /root/rpmbuild/BUILD/proftpd-1.3.5

+ exit 0
```

После этого ставим сам proftpd

```
# cd /root/rpmbuild/RPMS/x86_64/
# rpm -ivh proftpd-1.3.5-0.1.a.el6.x86_64.rpm  proftpd-
utils-1.3.5-0.1.a.el6.x86_64.rpm proftpd-ldap-1.3.5-0.1.a.el6.x86_64.rpm
```

Остальные пакеты, а именно

- proftpd-devel-1.3.5-1.el6.x86_64.rpm
- proftpd-mysql-1.3.5-1.el6.x86_64.rpm
- proftpd-wrap-1.3.5-1.el6.x86_64.rpm

необходимо будет доустановить по мере необходимости. А теперь переходим к самому интересному, настройкой proftpd, а именно к редактированию файла /etc/proftpd.conf.

Несколько комментариев к данному файлу. Все, что касается описания виртуальных хостов и их характеристик я специально вынес в отдельные файлы, которые располагаются в /etc/proftpd/conf.d/. Это сделано специально, чтобы основной конфигурационный файл было удобно читать и он не загромождался описанием виртуальных хостов. Примерно также принято делать и в apache/nginx. Т.е. здесь мы описываем глобальные переменные и поведение сервера в целом, а уже в конкретном виртуальном хосте - конкретизируем эти требования.

```
ServerAdmin                     root@localhost
ServerType                      standalone
AccessGrantMsg                  "User %u logged in."

# Загружаем необходимые модули
LoadModule mod_auth_pam.c
LoadModule mod_ldap.c
LoadModule mod_sftp.c
LoadModule mod_tls.c
LoadModule mod_rewrite.c

<IfModule mod_auth_pam.c>
    AuthPAMConfig proftpd
</IfModule>
```

```
# Указываем какие версии протокола можно использовать при шифровании канала в ftps режиме
<IfModule mod_tls.c>
    # TLSv1.1 TLSv1.2 будет работать только в 1.3.5+. В 1.3.4 можно указывать только TLSv1
    TLSProtocol TLSv1 TLSv1.1 TLSv1.2
</IfModule>

# Отключаем сервер по умолчанию (default server)
DefaultServer off
Port 0

# Задаем маску по умолчанию
Umask 022

# Задаем режим отображения содержимого директорий
ListOptions "-a"

# Включаем поддержку докачки и дозагрузки файлов
AllowRetrieveRestart            on
AllowStoreRestart               on

# Максимальное количество дочерних процессов. Так же ограничивает кол-во одновременных соединений
MaxInstances 20

# Пользователь и группа, от которых работает proftpd
User nobody
Group nobody

# Включаем использование sendfile() функциональности, что снижает затраты на ресурсы,
# но его включение нарушает отображение статистики в таких утилитах как ftptop и ftpwho
UseSendfile yes

# Путь к служебному файлу Scoreboard, который используется для хранения т.н. "scoreboard" информации,
# например, для корректной обработки MaxClients
ScoreboardFile /var/run/proftpd.score

# Задаем временную зону, соответствующую настройкам сервера
TimesGMT off
SetEnv TZ :/etc/localtime

# Отключаем проверку reverse DNS записей и Ident
UseReverseDNS off
IdentLookups off

# Ограничиваем максимальный размер загружаемого файла.
MaxStoreFileSize 512 Mb
```

```
# Включаем поддержку более безопасной загрузки файлов. Вначале файл загружается с именем .in.filename.ext
# И только после успешной и полной загрузки он переименовывается в filename.ext. Позволяет избежать проблем
# частичной загрузки файлов, а так же случаев, когда частично загруженный файл начинает использоваться системой
HiddenStores on

# Подключаем необходимый виртуальный хост, если в /etc/sysconfig/proftpd есть соответствующая директива
# например для включение VHOST1_21 добавляем PROFTPD_OPTIONS="-DVHOST1_21"
<IfDefine VHOST1_21>
    Include /etc/proftpd/conf.d/vhost1_default_21.conf
</IfDefine>

<IfDefine VHOST2_2121>
    Include /etc/proftpd/conf.d/vhost2_2121.conf
</IfDefine>

<IfDefine VHOST3_3333>
    Include /etc/proftpd/conf.d/vhost3_3333.conf
</IfDefine>

<IfDefine VHOST4_4444>
    Include /etc/proftpd/conf.d/vhost4_4444.conf
</IfDefine>

# Глобальные переменные
<Global>
    RequireValidShell off
    AllowOverwrite yes

    DenyFilter \*.*/

    <Limit SITE_CHMOD>
        DenyAll
    </Limit>
</Global>

# Включаем логирование
LogFormat default "%h %l %u %t \"%r\" %s %b"
LogFormat auth    "%v [%P] %h %t \"%r\" %s"
ExtendedLog /var/log/proftpd/access.log read,write
```

Ниже привожу содержимое /etc/sysconfig/proftpd

```
# cat /etc/sysconfig/proftpd
PROFTPD_OPTIONS="-DVHOST1_21 -DVHOST2_2121 -DVHOST3_3333 -DVHOST4_4444"
```

VHOST1: protocol - FTP, port - 21, auth - plain file (virtual users)

А теперь опишем первый виртуальный хост. Так как это будет чистый ftp и использоваться он будет только одной программой, то нет смысла заводить системного пользователя. Вместо этого мы будем использовать виртуальных пользователей. Данные о самом пользователе будут храниться в обычном текстовом файле.

```
# ftpasswd --passwd --file /etc/proftpd/virtual_users.passwd --sha512 --gid
99 --uid 99 --shell /sbin/nologin --name screen --home
/vhosts/screen.example.net

ftpasswd: using alternate file: /etc/proftpd/virtual_users.passwd
ftpasswd: creating passwd entry for user screen

Password:
Re-type password:

ftpasswd: entry created
```

В результате у нас получится файл с примерно таким содержимым

```
# cat /etc/proftpd/virtual_users.passwd
screen:$6$Vs7rWzTNHOp4JNCW$bvB7i5bd.t.lVEhXmn5J/0/LbtKcX6bMV352t4jULazNpZLhb
jqNjR/zxx8Fzjuse83lgplE7PFR9mjzVJHGw1:99:99::/vhosts/screen.example.net:/sbi
n/nologin
```

```
###
# /etc/proftpd/conf.d/vhost1_default_21.conf
###

<VirtualHost screen.example.net>
# Задаем общие настройки виртуального хоста
ServerName "FTP: Screenshots."
DefaultRoot ~
DefaultServer on
Umask 002
Port 21

# Задаем поддержку только FTP протокола. На момент написания статьи 26.02.2016
proftpd
# поддерживал следующие протоколы - ftp/ftps/sftp/scp(mod_sftp)
Protocols ftp

# Разрешаем аутентификацию только через модуль mod_auth_file, который является
авторитативным.
# Так как ProFTPD обрабатывает модули в порядке их указания, то добавление * в конце
имени модуля
# говорит о том, что поиск стоит прекратить, так как ответ данного модуля "заслуживает
```

```
доверие"
AuthOrder mod_auth_file.c*

# А здесь мы указываем путь к файлу, который мы создали перед этим. Именно на основании его содержимого
# и будет производиться аутентификация. Так же задаем дополнительное ограничение, домашний каталог
# пользователя должен быть /vhosts/screen.example.net/. Так же можно ограничить и имя пользователя
AuthUserFile /etc/proftpd/virtual_users.passwd home
^/vhosts/screen.example.net$

# Разрешаем вход только пользователю screen
<Limit LOGIN>
   AllowUser screen
   DenyAll
</Limit>

# В целях безопасности запрещаем удаление или переименование файлов, только upload и download
<Directory /vhosts/screen.example.net/*>
   <Limit DELE RMD XRMD RNFR RNTO>
      DenyAll
   </Limit>
</Directory>
</VirtualHost>
```

VHOST2: protocol - SFTP, port - 2121, auth - PAM

Второй виртуальный хост используется разработчиками, для тестирования различных CMS. Тут как бы все просто.

```
###
# /etc/proftpd/conf.d/vhost2_2121.conf
###

<VirtualHost sftp.example.net>
# Задаем общие настройки виртуального хоста
ServerName "SFTP: Staging."
DefaultRoot ~
Umask 002
Port 2121

# Полностью сбрасываем все привилегии root, как только пользователь прошел аутентификацию
# Данная опция будет работать, только если номер порта > 1024
RootRevoke on
```

```
# Задаем поддержку только FTP протокола. На момент написания статьи 26.02.2016
proftpd
# поддерживал следующие протоколы - ftp/ftps/sftp/scp(mod_sftp)
Protocols sftp

# Разрешаем аутентификацию только через модуль mod_auth_unix, который является
авторитативным.
# Т.е. будут использоваться системные пользователи
AuthOrder mod_auth_unix.c*

# Включаем модуль sftp
<IfModule mod_sftp.c>
   SFTPEngine on
   SFTPLog /var/log/proftpd/sftp.log
   SFTPHostKey /etc/ssh/ssh_host_rsa_key
   SFTPHostKey /etc/ssh/ssh_host_dsa_key
   SFTPAuthMethods password
   SFTPCompression delayed
</IfModule>
</VirtualHost>
```

VHOST3: protocol - FTPS, port - 3333, auth - LDAP

И на последок я оставил, пожалуй, «сложный» виртуальный хост. Он используется front-end разработчиками для проверки SSI (Server Side Includes). Для того, чтобы на сервере был порядок и было понятно, кто и какой проект загрузил, то я решил использовать существующую базу LDAP. Что очень удобно, так как позволяет заходить пользователю с уже имеющимися доменными учетными данными. Но так, как сам по себе ftp протокол никак не защищен, то мы будем использовать шифрование - FTPS. Но почему же тогда не использовать sftp? Тут чисто технические ограничения - пользователям удобно, когда папка на FTP сервере монтируется как сетевой диск. В случае с sftp протоколом я не смог найти нормальной и бесплатной программы, которая поддерживала бы монтирование сетевого диска с удаленного сервера по sftp протоколу. Если вы такую знаете - буду рад услышать.

У FTPS есть два режима работы - explicit и implicit. Если производить аналогии с smtp, то в первом случае мы будем использовать некий аналог starttls, чтобы указать серверу, что необходимо шифрование. Во втором случае мы явно должны установить шифрованный тунель и лишь потом производить любой обмен данными. Для включения implicit режима в TLSOptions необходимо добавить UseImplicitSSL.

FileZilla: Explicit FTP

```
ftpes://user:pass@ip.add.re.ss:port/
```

Command line: Explicit FTP

```
# openssl s_client -connect ip.add.re.ss:port -starttls ftp
```

FileZilla: Implicit FTP

```
ftps://user:pass@ip.add.re.ss:port/
```

Command line: Implicit FTP

```
# openssl s_client -connect ip.add.re.ss:port
```

Однозначно сложно сказать, какой из режимов лучше использовать. Тут придется пробовать, многое будет зависеть от ftp клиентов, но implicit режим считается устаревшим.

```
###
# /etc/proftpd/conf.d/vhost3_3333.conf
###

<VirtualHost ftps.example.net>
# Задаем общие настройки виртуального хоста
ServerName "FTPES: SSI."
DefaultRoot ~
Umask 002
Port 3333

# Полностью сбрасываем все привилегии root, как только пользователь прошел аутентификацию
# Данная опция будет работать, только если номер порта > 1024
RootRevoke on

# Задаем поддержку только FTP протокола. На момент написания статьи 26.02.2016 proftpd
# поддерживал следующие протоколы - ftp/ftps/sftp/scp(mod_sftp)
Protocols ftps

# Разрешаем аутентификацию только через модуль mod_ldap, который является авторитативным.
# Т.е. будут использоваться пользователи из LDAP каталога
AuthOrder mod_ldap.c*

# Включаем шифрование
<IfModule mod_tls.c>
  # Включаем поддержку TLS
    TLSEngine on
    TLSLog /var/log/proftpd/tls.log

  # Требуем обязательного шифрования канала данных и управляющего канала
    TLSRequired on

  # Другие опция нашего шифрования
    TLSOptions NoCertRequest EnableDiags NoSessionReuseRequired

  # указываем сертификат и ключ
    TLSRSACertificateFile         /etc/pki/proftpd/ftps.example.net.crt
```

```
   TLSRSACertificateKeyFile      /etc/pki/proftpd/ftps.example.net.key
   TLSCipherSuite
kEECDH+AES128:kEECDH:kEDH:-3DES:kRSA+AES128:kEDH+3DES:DES-CBC3-
SHA:!RC4:!aNULL:!eNULL:!MD5:!EXPORT:!LOW:!SEED:!CAMELLIA:!IDEA:!PSK:!SRP:!SS
Lv2
</IfModule>

# Включаем поддержку LDAP
<IfModule mod_ldap.c>
  # Задаем адрес LDAP сервера, а так же где и как искать пользователей
   LDAPServer 127.0.0.1:389
   LDAPProtocolVersion 3
   LDAPAuthBinds on
   LDAPBindDN "uid=ldap_reader,ou=System,ou=users,dc=example,dc=net"
"7654321"
   LDAPUsers ou=users,dc=example,dc=net
"(&(uid=%u)(objectclass=posixAccount))"
   LDAPGroups ou=groups,dc=example,dc=net

  # Задаем, чтобы у всех аутентифицированных пользователей LDAP был одинаковый
uid/gid
   LDAPForceDefaultUID on
   LDAPForceDefaultGID on

   # 99/99 - nobody/nobody
   LDAPDefaultUID 99
   LDAPDefaultGID 99

  # Указываем автоматически создавать домашнюю папку для пользователя. Задаем права и
владельца
   CreateHome on 0775 skel /etc/skel/empty dirmode 0775 uid 99 gid 99
   LDAPGenerateHomedir on 0775
   LDAPGenerateHomedirPrefix /vhosts/ssi.example.net
   LDAPForceGeneratedHomedir on
</IfModule>
</VirtualHost>
```

Так как пользователей в LDAP более 500, то функция автоматического создания home директории при первом заходе - очень удобная и сильно облегчает жизнь администратору.

VHOST4: protocol - SFTP, port - 4444, auth - public key

На днях попался интересный вопрос на ServerFault. В котором человек спрашивал про возможность использовать sftp сервис, проходить аутентификацию с помощью открытых ключей и при этом, чтобы не было необходимости для каждого пользователя создавать учетную запись, даже виртуальную. При этом нужно сделать так, чтобы пользователи видели только свои данные и не имели доступа к данным друг друга.

Даже казалось бы такой сложный и «странный» сервис можно настроить средствами ProFTPD. За этого я его и люблю и мирюсь с относительно частыми проблемами безопасности. Это как раз тот случай, когда за функционал приходится платить.

```
###
# /etc/proftpd/conf.d/vhost4_4444.conf
###

<VirtualHost backup.example.net>
    ServerName "SFTP: Backup server. Public key auth only!"
    DefaultRoot ~
    Umask 002
    Port 4444

  # Задаем поддержку только SFTP протокола. На момент написания статьи 26.02.2016
proftpd
  # поддерживал следующие протоколы - ftp/ftps/sftp/scp(mod_sftp)
    Protocols sftp
  # Разрешаем аутентификацию только через модуль mod_auth_file, который является
авторитативным.
  # Так как ProFTPD обрабатывает модули в порядке их указания, то добавление * в конце
имени модуля
  # говорит о том, что поиск стоит прекратить, так как ответ данного модуля "заслуживает
доверие"
    AuthOrder mod_auth_file.c*

    RootRevoke on

# Включаем модуль sftp
<IfModule mod_sftp.c>
    SFTPEngine on
    SFTPLog /var/log/proftpd/sftp.log

    SFTPHostKey /etc/ssh/ssh_host_rsa_key
    SFTPHostKey /etc/ssh/ssh_host_dsa_key
  # Путь к файлу с DH параметрами (Diffie-Hellman)
    SFTPDHParamFile /etc/pki/proftpd/dhparam_2048.pem
  # Здесь у нас будут храниться открытые ключи пользователей в формате RFC4716
    SFTPAuthorizedUserKeys file:/etc/proftpd/sftp_authorized_keys
  # Здесь мы описываем нашего единственного пользователя
    AuthUserFile /etc/proftpd/sftp_users.passwd home ^/vhosts/backup/
    SFTPCompression delayed
  # Разрешаем проходить аутентификацию только по открытому ключу
    SFTPAuthMethods publickey

  # Настраиваем автоматическое создание домашней папки
    CreateHome on 0700 dirmode 0700 uid 99 gid 99

  # А здесь собственно и происходит весь трюк. С помощью mod rewrite правила мы
```

```
динамически
  # задаем домашнюю директорию для пользователя. Таким образом нет необходимости
создавать
  # запись в файле sftp_users.passwd для каждого пользователя. Достаточно лишь
добавить его
  # открытый ключ в файл sftp_authorized_keys.

    <IfModule mod_rewrite.c>
    # Включаем возможность перезаписи домашней директории
        RewriteHome on

    # Включаем поддержку mod rewrite
        RewriteEngine on
    # Подробный лог файл, в котором будут отображаться все шаги перезаписи.
    # Очень удобно при поиске ошибок.
        RewriteLog /var/log/proftpd/rewrite.log
    # Задаем условие, при котором будет срабатывать наше правило. Я, думаю,
    # вы заметили, что директивы аналогичны mod rewrite с apache.
        # %m - будет заменена на FTP команду. В нашем случае, нас интересует
REWRITE_HOME
        RewriteCondition %m REWRITE_HOME
    # Собственно производим замену.
        # %U - имя пользователя, которое было использовано в FTP команде user
        RewriteRule (.*) /vhosts/backup/%U
    </IfModule>
</IfModule>
</VirtualHost>
```

Конвертируем открытый ключ в формат RFC4716

```
# ssh-keygen -e -f user1.public.key >> /etc/proftpd/sftp_authorized_keys
# ssh-keygen -e -f user2.public.key >> /etc/proftpd/sftp_authorized_keys
```

В файле /etc/proftpd/sftp_authorized_keys обязательно должна быть новая строка в конце файла! Иначе при попытке аутентификации вы будете получать

```
2016-02-26 14:27:44,050 mod_sftp/0.9.9[20427]: line too long (29) on line 13
of '/etc/proftpd/sftp_authorized_keys'
2016-02-26 14:27:44,050 mod_sftp/0.9.9[20427]: Make sure that
'/etc/proftpd/sftp_authorized_keys' is a RFC4716 formatted key
2016-02-26 14:27:44,050 mod_sftp/0.9.9[20427]: error comparing keys from
'/etc/proftpd/sftp_authorized_keys': Invalid argument
2016-02-26 14:27:44,051 mod_sftp/0.9.9[20427]: sending userauth failure;
remaining userauth methods: publickey
2016-02-26 14:27:44,052 mod_sftp/0.9.9[20427]: client at 192.168.1.2 sent
SSH_DISCONNECT message: No supported authentication methods available (No
other authentication mechanisms available)
```

Генерируем файл с DH параметрами

```
# mkdir /etc/pki/proftpd/
# openssl dhparam -out /etc/pki/proftpd/dhparam_2048.pem 2048
```

Создаем нашего единственного виртуального пользователя

```
# ftpasswd --passwd --file /etc/proftpd/sftp_users.passwd --sha512 --gid 99
--uid 99 --shell /sbin/nologin --name user1 --home /vhosts/backup
```

Думаю очевидно, что такое 99. Но в принципе вам никто не запрещает создать специального пользователя для этих целей.

```
# id nobody
uid=99(nobody) gid=99(nobody) groups=99(nobody)
```

Создаем корневую папку и выставляем права

```
# mkdir -p /vhosts/backup/
# chmod -R 700 /vhosts/backup/
# chown -R nobody:nobody /vhosts/backup/
```

Тестирование

Ну и заключительная часть - тестирование. А как же без него! Как мы помним, на 21 порту у нас запущен обычный FTP сервер с аутентификацией с поддержкой виртуальных пользователей, данные которых хранятся в обычном текстовом файле. Так же у пользователя мы убрали возможность удалять или переименовывать файлы. Проверим наши настройки.

```
# ftp 192.168.127.1 21
Connected to 192.168.127.1 (192.168.127.1).
220 ProFTPD 1.3.5 Server (FTP: Screenshots.) [::ffff:192.168.127.1]
Name (192.168.127.1:root): screen
331 Password required for screen
Password: *******
230 User screen logged in
Remote system type is UNIX.
Using binary mode to transfer files.

ftp> ls
227 Entering Passive Mode (192,168,127,1,137,204).
150 Opening ASCII mode data connection for file list
226 Transfer complete

ftp> put smart.txt
local: smart.txt remote: smart.txt
227 Entering Passive Mode (192,168,127,1,144,162).
150 Opening BINARY mode data connection for smart.txt
226 Transfer complete
588 bytes sent in 1.8e-05 secs (32666.67 Kbytes/sec)
```

```
ftp> ls
227 Entering Passive Mode (192,168,127,1,230,34).
150 Opening ASCII mode data connection for file list
-rw-rw-r--   1 nobody   nobody        588 Apr  7 11:38 smart.txt
226 Transfer complete

ftp> delete smart.txt
550 smart.txt: Operation not permitted

ftp> rename smart.txt temp.txt
550 smart.txt: Operation not permitted

ftp> quit
221 Goodbye.
```

Как видно, все работает именно так, как мы и планировали. Если запустить proftpd в режиме отладки, смотри ниже, то мы увидим следующие строки

```
2015-04-07 07:38:39,347 ftp.example.net proftpd[9621] 192.168.127.1
(192.168.127.2[192.168.127.2]): user 'screen' authenticated by
mod_auth_file.c
2015-04-07 07:38:39,347 ftp.example.net proftpd[9621] 192.168.127.1
(192.168.127.2[192.168.127.2]): Preparing to chroot to directory
'/vhosts/screen.example.net'
2015-04-07 07:38:39,347 ftp.example.net proftpd[9621] 192.168.127.1
(192.168.127.2[192.168.127.2]): Environment successfully chroot()ed
...
...
...
2015-04-07 07:49:36,807 ftp.example.net proftpd[9770] 192.168.127.1
(192.168.127.2[192.168.127.2]): deleting '/smart.txt' denied by <Limit>
configuration
2015-04-07 07:49:36,807 ftp.example.net proftpd[9770] 192.168.127.1
(192.168.127.2[192.168.127.2]): dispatching LOG_CMD_ERR command 'DELE
smart.txt' to mod_log
```

При этом, если мы попробуем зайти под любым другим пользователем, даже который есть в файле /etc/proftpd/virtual_users.passwd, то в логах мы увидим подобные строки

```
2015-04-07 07:55:07,636 ftp.example.net proftpd[9843] 192.168.127.1
(192.168.127.2[192.168.127.2]): USER borodach (Login failed): Limit access
denies login
```

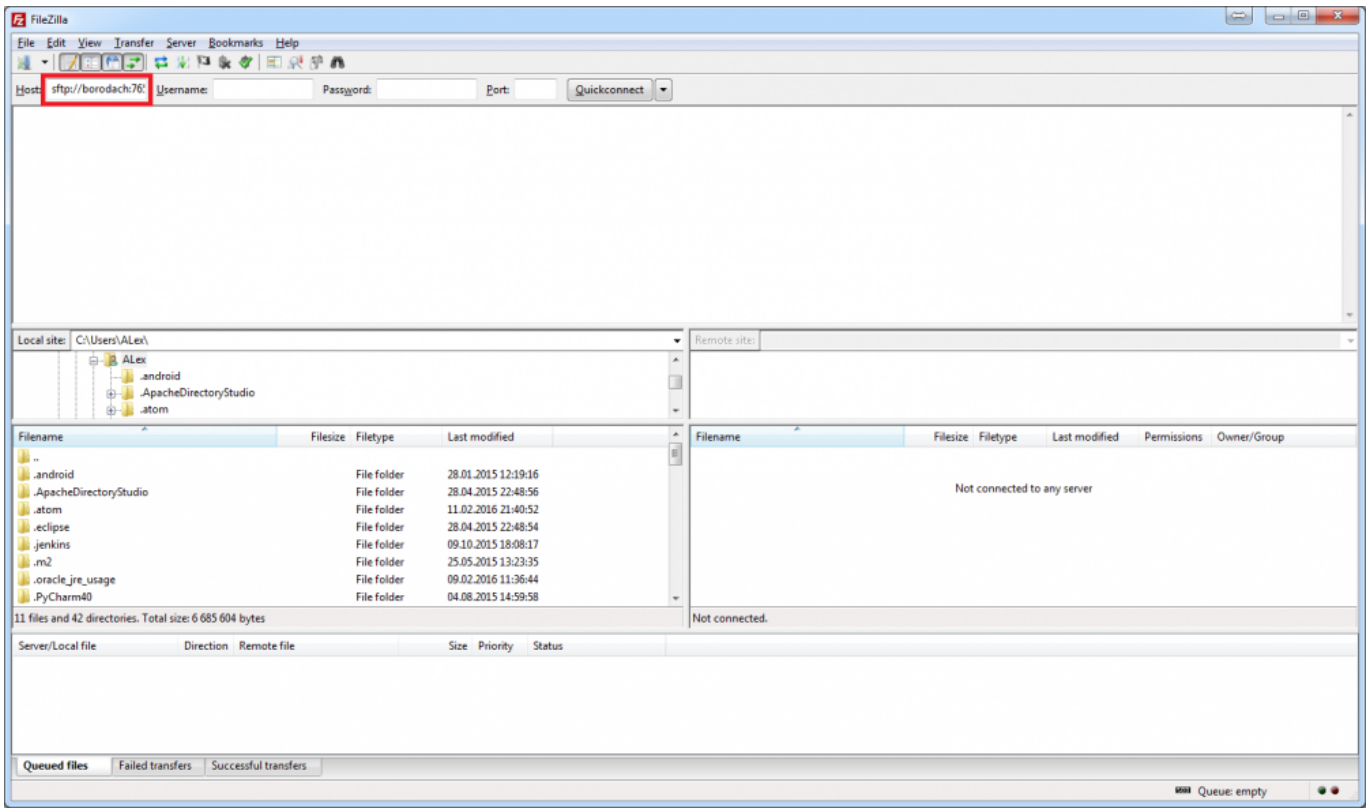
Теперь проверяем второй виртуальный хост, который работает по sftp протоколу

```
# sftp -oPort=2121 borodach@192.168.127.1
Connecting to 192.168.127.1...
borodach@192.168.127.1's password:

sftp> lpwd
Local working directory: /home/borodach
```

```
sftp> pwd
Remote working directory: /

sftp> put ./smart.txt
Uploading ./smart.txt to /smart.txt
./smart.txt
100%  588     0.6KB/s   00:00

sftp> ls
smart.txt

sftp> rm smart.txt
Removing /smart.txt

sftp> quit
```

Думаю, тут все предельно просто и ясно. Для удобства быстрого доступа пользователей через FileZilla, в описании проекта я генерирую ссылку быстрого доступа, которая имеет следующий вид:

```
   sftp://borodach:7654321@192.168.127.1:2121/
```



Если же вам необходимо проверить аутентификацию по публичному ключу, то необходимо использовать следующую команду

```
# sftp -oPort=2121 -oIdentityFile=borodach_private.key
borodach@192.168.127.1
```

И последний виртуальный хост - FTPS (explicit). Строка быстрого подключения для FileZilla будет выглядеть

```
   ftpes://borodach:7654321@192.168.127.1:3333/
```

Из командной строки можно проверить с помощью openssl

```
# openssl s_client -tls1_2 -starttls ftp -connect 192.168.127.1:3333
CONNECTED(00000003)
depth=0 description = 82pjOgKm8a9O4NG0, C = UA, CN = test.example.net,
emailAddress = webmaster@example.net
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 description = 82pj4NG0OgKm8a9O, C = UA, CN = test.example.net,
emailAddress = webmaster@example.net
verify error:num=27:certificate not trusted
verify return:1
depth=0 description = 82pjOgKm8a9O4NG0, C = UA, CN = test.example.net,
emailAddress = webmaster@example.net
verify error:num=21:unable to verify the first certificate
verify return:1
---
Certificate chain
 0
s:/description=82pjOgKm8a9O4NG0/C=UA/CN=test.example.net/emailAddress=webmas
ter@example.net
   i:/C=IL/O=StartCom Ltd./OU=Secure Digital Certificate Signing/CN=StartCom
Class 1 Primary Intermediate Server CA
---
...
...
...
---
No client certificate CA names sent
Server Temp Key: ECDH, prime256v1, 256 bits
---
SSL handshake has read 2711 bytes and written 385 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 4096 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID:
26369EB0834724AB1491D64F2890DF4F360DE45468D9617808B794A7E8347195
    Session-ID-ctx:
    Master-Key:
A37FEAE590ABD42CD731AFC4D4799D706C9238E974DF0CF30A14D7A02FB2CB972F221DA07894
```

```
BAC14CEF8249A16BF580
    Key-Arg   : None
    Krb5 Principal: None
    PSK identity: None
    PSK identity hint: None
    Start Time: 1428414903
    Timeout   : 7200 (sec)
    Verify return code: 21 (unable to verify the first certificate)
---
220 ProFTPD 1.3.5 Server (FTPES: SSI.) [::ffff:192.168.127.1]
user borodach
331 Password required for borodach
pass 7654321
230 User borodach logged in
quit
221 Goodbye.
read:errno=0
```

В режиме отладки обращаем внимание не следующие строчки

```
2015-04-07 09:51:32,215 ftp.example.net proftpd[11484] 192.168.127.1
(192.168.127.2[192.168.127.2]): user 'borodach' authenticated by mod_ldap.c
2015-04-07 09:51:32,215 ftp.example.net proftpd[11484] 192.168.127.1
(192.168.127.2[192.168.127.2]): creating home directory
'/vhosts/ssi.example.net/borodach' for user 'borodach'
2015-04-07 09:51:32,215 ftp.example.net proftpd[11484] 192.168.127.1
(192.168.127.2[192.168.127.2]): CreateHome: '/' already exists
2015-04-07 09:51:32,215 ftp.example.net proftpd[11484] 192.168.127.1
(192.168.127.2[192.168.127.2]): CreateHome: '/vhosts' already exists
2015-04-07 09:51:32,215 ftp.example.net proftpd[11484] 192.168.127.1
(192.168.127.2[192.168.127.2]): CreateHome: '/vhosts/ssi.example.net'
already exists
2015-04-07 09:51:32,215 ftp.example.net proftpd[11484] 192.168.127.1
(192.168.127.2[192.168.127.2]): CreateHome: directory
'/vhosts/ssi.example.net/borodach' created
2015-04-07 09:51:32,215 ftp.example.net proftpd[11484] 192.168.127.1
(192.168.127.2[192.168.127.2]): home directory
'/vhosts/ssi.example.net/borodach' created
2015-04-07 09:51:32,215 ftp.example.net proftpd[11484] 192.168.127.1
(192.168.127.2[192.168.127.2]): CreateHome: copying skel files from
'/etc/skel/empty' into '/vhosts/ssi.example.net/borodach'
```

Для тестирование нашего последнего виртуального хоста лучше использовать FileZilla или любой другой клиент с поддержкой SFTP протокола. Итак, подключаемся с ключом первого пользователя.

```
2016-02-26 14:57:36,758 backup.example.net proftpd[20481] 192.168.1.6
(192.168.1.2[192.168.1.2]): creating home directory '/vhosts/backup/user1'
for user 'user1'
2016-02-26 14:57:36,758 backup.example.net proftpd[20481] 192.168.1.6
(192.168.1.2[192.168.1.2]): CreateHome: '/' already exists
```

```
2016-02-26 14:57:36,758 backup.example.net proftpd[20481] 192.168.1.6
(192.168.1.2[192.168.1.2]): CreateHome: '/vhosts' already exists
2016-02-26 14:57:36,758 backup.example.net proftpd[20481] 192.168.1.6
(192.168.1.2[192.168.1.2]): CreateHome: '/vhosts/backup' already exists
2016-02-26 14:57:36,759 backup.example.net proftpd[20481] 192.168.1.6
(192.168.1.2[192.168.1.2]): home directory '/vhosts/backup/user1' created
2016-02-26 14:57:36,759 backup.example.net proftpd[20481] 192.168.1.6
(192.168.1.2[192.168.1.2]): Preparing to chroot to directory
'/vhosts/backup/user1'
2016-02-26 14:57:36,759 backup.example.net proftpd[20481] 192.168.1.6
(192.168.1.2[192.168.1.2]): Environment successfully chroot()ed
2016-02-26 14:57:36,759 backup.example.net proftpd[20481] 192.168.1.6
(192.168.1.2[192.168.1.2]): USER user1: Login successful
2016-02-26 14:57:36,759 backup.example.net proftpd[20481] 192.168.1.6
(192.168.1.2[192.168.1.2]): RootRevoke in effect, dropped root privs
2016-02-26 14:57:36,759 backup.example.net proftpd[20481] 192.168.1.6
(192.168.1.2[192.168.1.2]): USER user1: Login successful.
```

И вот что при этом будет в файле rewrite.log

```
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_subst_vars():
replacing variable '%m' with 'REWRITE_HOME'
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_subst(): var subst'd
pattern: 'REWRITE_HOME'
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_parse_map_str():
parsing 'REWRITE_HOME'
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_subst(): maps
subst'd pattern: 'REWRITE_HOME'
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_subst(): env subst'd
pattern: 'REWRITE_HOME'
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_match_cond: subst'd
cond: 'REWRITE_HOME'
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_match_cond():
checking regex cond against 'REWRITE_HOME'
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_fixup(): condition
met
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_fixup(): executing
RewriteRule
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_subst(): original
pattern: '/vhosts/backup/%U'
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_subst(): rule
backref subst'd pattern: '/vhosts/backup/%U'
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_subst(): pattern
'/vhosts/backup/%U' had no cond backrefs
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_subst_vars():
replacing variable '%U' with 'user1'
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_subst(): var subst'd
pattern: '/vhosts/backup/user1'
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_parse_map_str():
parsing '/vhosts/backup/user1'
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_subst(): maps
```

```
subst'd pattern: '/vhosts/backup/user1'
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_subst(): env subst'd
pattern: '/vhosts/backup/user1'
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrite_fixup():
REWRITE_HOME arg now '/vhosts/backup/user1'
2016-02-26 14:57:36,759 mod_rewrite/0.9[20481]: rewrote home to be
'/vhosts/backup/user1'
```

При подключении с ключом второго пользователя картина будет аналогичная, за исключением того, что rewrite будет сделан в папку /vhosts/backup/user2

```
2016-02-26 15:03:49,786 mod_rewrite/0.9[20496]: rewrite_match_cond():
checking regex cond against 'REWRITE_HOME'
2016-02-26 15:03:49,786 mod_rewrite/0.9[20496]: rewrite_fixup(): condition
met
2016-02-26 15:03:49,786 mod_rewrite/0.9[20496]: rewrite_fixup(): executing
RewriteRule
2016-02-26 15:03:49,786 mod_rewrite/0.9[20496]: rewrite_subst(): original
pattern: '/vhosts/backup/%U'
2016-02-26 15:03:49,786 mod_rewrite/0.9[20496]: rewrite_subst(): rule
backref subst'd pattern: '/vhosts/backup/%U'
2016-02-26 15:03:49,786 mod_rewrite/0.9[20496]: rewrite_subst(): pattern
'/vhosts/backup/%U' had no cond backrefs
2016-02-26 15:03:49,786 mod_rewrite/0.9[20496]: rewrite_subst_vars():
replacing variable '%U' with 'user2'
2016-02-26 15:03:49,786 mod_rewrite/0.9[20496]: rewrite_subst(): var subst'd
pattern: '/vhosts/backup/user2'
2016-02-26 15:03:49,786 mod_rewrite/0.9[20496]: rewrite_parse_map_str():
parsing '/vhosts/backup/user2'
2016-02-26 15:03:49,786 mod_rewrite/0.9[20496]: rewrite_subst(): maps
subst'd pattern: '/vhosts/backup/user2'
2016-02-26 15:03:49,786 mod_rewrite/0.9[20496]: rewrite_subst(): env subst'd
pattern: '/vhosts/backup/user2'
2016-02-26 15:03:49,786 mod_rewrite/0.9[20496]: rewrite_fixup():
REWRITE_HOME arg now '/vhosts/backup/user2'
2016-02-26 15:03:49,786 mod_rewrite/0.9[20496]: home directory
'/vhosts/backup/user2' not changed by RewriteHome
```

Отладка Proftpd

В proftpd реализован удобный режим отладки, так что если у вас что либо не работает, или сам демон не запускается, то советую запускать в режиме отладки, примерно таким образом

```
# proftpd -n -d 3 -DVHOST1_21
2015-04-07 06:27:25,490 ftp.example.net proftpd[8588]: using PCRE 7.8
2008-09-05
2015-04-07 06:27:25,494 ftp.example.net proftpd[8588]: mod_ldap/2.9.4:
compiled using LDAP vendor 'OpenLDAP', LDAP API version 3001
2015-04-07 06:27:25,494 ftp.example.net proftpd[8588]: mod_sftp/0.9.9: using
```

```
OpenSSL 1.0.1e-fips 11 Feb 2013
2015-04-07 06:27:25,495 ftp.example.net proftpd[8588]: mod_tls/2.6: using
OpenSSL 1.0.1e-fips 11 Feb 2013
2015-04-07 06:27:25,498 ftp.example.net proftpd[8588]: <IfModule>: using
'mod_auth_pam.c' section at line 27
2015-04-07 06:27:25,498 ftp.example.net proftpd[8588]: <IfModule>: using
'mod_tls.c' section at line 31
2015-04-07 06:27:25,498 ftp.example.net proftpd[8588]: <IfDefine>: using
'VHOST1_21' section at line 73
2015-04-07 06:27:26,194 ftp.example.net proftpd[8588]: <Directory
/vhosts/screen.example.net/*>: adding section for resolved path
'/vhosts/screen.example.net/*'
2015-04-07 06:27:26,687 ftp.example.net proftpd[8588]: <IfDefine>: skipping
'VHOST2_2121' section at line 77
2015-04-07 06:27:26,687 ftp.example.net proftpd[8588]: <IfDefine>: skipping
'VHOST3_3333' section at line 81
2015-04-07 06:27:26,835 ftp.example.net proftpd[8588] 192.168.127.1: set
core resource limits for daemon
2015-04-07 06:27:26,835 ftp.example.net proftpd[8588] 192.168.127.1: ProFTPD
1.3.5 (stable) (built Mon Apr 6 2015 02:25:07 EDT) standalone mode STARTUP
```

Из вывода видно, что, так как мы не указали -DVHOST2_2121 и -DVHOST3_3333, то соответствующие секции виртуальных хостов были пропущены «skipping» и соответственно эти виртуальные хосты не будут работать.

Если вам не хватает 3го уровня информативности, т.е. не видно причину ошибки, то можно увеличить уровень информативности вплоть до 10.