Configuration Directive List

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Configuration Directive List

Table of Contents

List of Tables

List of Examples

Chapter 1. List of Directives

AccessDenyMsg

Name

AccessDenyMsg -- Customise the response on failed authentication

Synopsis

AccessDenyMsg [`"message"`]

`Default`
: Dependent on login type

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_auth

`Compatibility`
: 1.2.2 and later

Description

Normally, a 530 response message is sent to an FTP client immediately after a failed authentication attempt, with a standard message indicating the the reason of failure. In the case of a wrong password, the reason is usually "Login incorrect." This message can be customized with the AccessDenyMsg directive. In the message argument, the magic cookie '%u' is replaced with the username specified by the client during login.

See also

Examples

AccessDenyMsg "Guest access denied for %u."

AccessGrantMsg

Name

AccessGrantMsg -- Customise the response on successful authentication

Synopsis

AccessGrantMsg [`"message"`]

`Default`
: Dependent on login type

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_auth

`Compatibility`
: 0.99.0pl5 and later

Description

Normally, a 230 response message is sent to an FTP client immediately after authentication, with a standard message indicating that the user has either logged in or that anonymous access has been granted. This message can be customized with the AccessGrantMsg directive. In the message argument, the magic cookie '%u' is replaced with the username specified by the client during login.

See also

Examples

AccessGrantMsg "Guest access granted for %u."

Allow

Name

Allow -- Access control directive

Synopsis

Allow [`["from"] "all"|"none"|host|network[,host|network[,...]]`]

`Default`
: Allow from all

`Context`
: <Limit>

`Module`
: mod_core

`Compatibility`
: 0.99.0pl6 and later

Description

The Allow directive is used inside a <Limit> context to explicitly specify which hosts and/or networks have access to the commands or operations being limited. Allow is typically used in conjunction with Order and Deny in order to create sophisticated (or perhaps not-so-sophisticated) access control rules. Allow takes an optional first argument; the keyword from. Using from is purely cosmetic. The remaining arguments are expected to be a list of hosts and networks which will be explicitly granted access. The magic keyword all can be used to indicate that all hosts will explicitly be granted access (analogous to the AllowAll directive, except with a lower priority). Additionally, the magic keyword none can be used to indicate that no hosts or networks will be explicitly granted access (although this does not prevent them from implicitly being granted access). If all or none is used, no other hosts or networks can be supplied. Host and network addresses can be specified by name or numeric address. For security reasons, it is recommended that all address information be supplied numerically. Relying solely on named addresses causes security to depend a great deal upon DNS servers which may themselves be vulnerable to attack or spoofing. Numeric addresses which specify an entire network should end in a trailing period (i.e. 10.0.0. for the entire 10.0.0 subnet). Named addresses which specify an entire network should begin with a leading period (i.e. .proftpd.net for the entire proftpd.net domain).

See also

Allow Order Limit

Examples

```
<Limit LOGIN>
Order allow,deny
Allow from 128.44.26.,128.44.26.,myhost.mydomain.edu,.trusted-domain.org
Deny from all
</Limit>
```

AllowAll

Name

AllowAll -- Allow all clients

Synopsis

AllowAll [`AllowAll`]

`Default`
: Default is to implicitly AllowAll, but not explicitly

`Context`
: <Directory>, <Anonymous>, <Limit>, .ftpaccess

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The AllowAll directive explicitly allows access to a <Directory>, <Anonymous> or <Limit> block. Although proftpd's default behavior is to allow access to a particular object, the default is an implicit allow. AllowAll creates an explicit allow, overriding any higher level denial directives.

See also

DenyAll

Examples

AllowClass

Name

AllowClass -- Class based allow rules

Synopsis

AllowClass [`["AND"|"OR"|"regex"] class-expression`]

`Default`
: None

`Context`
: <Limit>

`Module`
: mod_core

`Compatibility`
: 1.2.10rc1 and later

Description

AllowClass specifies a class-expression that is specifically permitted access within the context of the <Limit> block it is applied to. class-expression has a similar syntax as that used in AllowGroup, in that it should contain a comma delimited list of classes or "not" classes (by prefixing a class name name with the `!' character) that are to be allowed access to the block.

By default, the expression is parsed as a boolean "OR" list, meaning that ANY elements of the expression must evaluate to logically true in order to the explicit allow to apply. In order to treat the expression as a boolean "AND" list, meaning that ALL of the elements must evaluate to logically true, use the optional "AND" keyword. Similarly, to treat the expression as a regular expression, use the "regex" keyword.

See also

AllowUser DenyUser AllowGroup DenyGroup DenyClass

Examples

```
# A regular expression AllowClass directive
AllowClass regex ^known

# An AND-evaluated ClassUser directive
DenyClass AND bad,scanner
```

AllowFilter

Name

AllowFilter -- Regular expression of command arguments to be accepted

Synopsis

AllowFilter [`regular-expression`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>, <Anonymous>, <Directoryl>, .ftpaccess

`Module`
: mod_core

`Compatibility`
: 1.2.0pre7 and later

Description

AllowFilter allows the configuration of a regular expression that must be matched for all command arguments sent to ProFTPD. It is extremely useful in controlling what characters may be sent in a command to ProFTPD, preventing some possible types of attacks against ProFTPD. The regular expression is applied against the arguments to the command sent by the client, so care must be taken when creating a proper regex. Commands that fail the regex match result in a "Forbidden command" error being returned to the client. If the regular-expression argument contains whitespace, it must be enclosed in quotes.

See also

DenyFilter

Examples

```
# Only allow commands containing alphanumeric characters and whitespace
AllowFilter "^[a-zA-Z0-9 ,]*$"
```

AllowForeignAddress

Name

AllowForeignAddress -- Control the use of the PORT command

Synopsis

AllowForeignAddress [`on|off`]

`Default`
: AllowForeignAddress off

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.1.7 and later

Description

Normally, proftpd disallows clients from using the ftp PORT command with anything other than their own address (the source address of the ftp control connection), as well as preventing the use of PORT to specify a low-numbered (< 1024) port. In either case, the client is sent an "Invalid port" error and a message is syslog'd indicating either "address mismatch" or "bounce attack". By enabling this directive, proftpd will allow clients to transmit foreign data connection addresses that do not match the client's address. This allows such tricks as permitting a client to transfer a file between two FTP servers without involving itself in the actual data connection. Generally it's considered a bad idea, security-wise, to permit this sort of thing.
AllowForeignAddress only affects data connection addresses; not tcp ports. There is no way (and no valid reason) to allow a client to use a low-numbered port in its PORT command.

See also

Examples

AllowGroup

Name

AllowGroup -- Group based allow rules

Synopsis

AllowGroup [`["AND"|"OR"|"regex"] group-expression`]

`Default`
: None

`Context`
: <Limit>

`Module`
: mod_core

`Compatibility`
: 1.1.1 and later

Description

AllowGroup specifies a group-expression that is specifically permitted within the context of the <Limit> block it is applied to. group-expression has the same format as that used in DefaultRoot, in that it should contain a comma separated list of groups or "not" groups (by prefixing a group name with the `!' character) that are to be allowed access to the block.

By default, the expression is parsed as a boolean "AND" list, meaning that ALL elements of the expression must evaluate to logically true in order to the explicit allow to apply. In order to treat the expression as a boolean "OR" list, meaning that ANY of the elements must evaluate to logically true, use the optional "AND" keyword. Similarly, to treat the expression as a regular expression, use the "regex" keyword.

See also

DenyGroup, DenyUser, AllowUser

Examples

```
# An OR-evaluated AllowGroup directive
AllowGroup OR www,doc

# A regular expression DenyGroup directive
DenyGroup regex ^sys
```

AllowLogSymlinks

Name

AllowLogSymlinks -- Permit logging to symlinked files

Synopsis

AllowLogSymlinks [`"on"|"off"`]

`Default`
: AllowLogSymlinks off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_log

`Compatibility`
: 1.2.2rc2 and later

Description

By default, the server will the path of any configured SystemLog, any configured TransferLogs, and any configured ExtendedLogs to see if they are symbolic links. If the paths are symbolic links, the server will refuse to log to that link unless explicitly configured to do so via this directive.

Security note:

Security note: this behaviour should not be allowed unless for a very good reason. By allowing the server to open symbolic links with its root privileges, you are allowing a potential symlink attack where the server could be tricked into overwriting arbitrary system files. You have been warned.

See also

Examples

```
AllowLogSymlinks on
```

AllowOverride

Name

AllowOverride -- Toggles handling of .ftpaccess files

Synopsis

AllowOverride [`on|off ["user"|"group"|"class" expression]`]

`Default`
: on

`Context`
: server config, <Global>, <VirtualHost>, <Anonymous>

`Module`
: mod_core

`Compatibility`
: 1.2.7rc1 and later

Description

Normally, the server will look for and parse any files in the encountered directories called ".ftpaccess". The files provide a functionality similar to Apache's .htaccess files -- mini-configuration files. This directive controls when those .ftpaccess files will be parsed.

The optional parameters are used to restrict the use of .ftpaccess files only to specific users. If the "user" restriction is given, then expression is a user-expression specifying to which users the rule applies. Similarly for the "group" restriction. For the "class" restriction, the expression is simply the name of connection class for whom the rule will apply.

See also

AllowOverwrite

Name

AllowOverwrite -- Enable files to be overwritten

Synopsis

AllowOverwrite [`on|off`]

`Default`
: AllowOverwrite off

`Context`
: server config, <VirtualHost>, <Anonymous>, <Directory>, <Global>, .ftpaccess

`Module`
: mod_xfer

`Compatibility`
: 0.99.0 and later

Description

The AllowOverwrite directive permits newly transfered files to overwrite existing files. By default, ftp clients cannot overwrite existing files.

See also

Examples

AllowRetrieveRestart

Name

AllowRetrieveRestart -- Allow clients to resume downloads

Synopsis

AllowRetrieveRestart [`on|off`]

`Default`
: AllowRetrieveRestart on

`Context`
: server config, <VirtualHost>, <Anonymous>, <Directory>, <Global>, .ftpaccess

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The AllowRetrieveRestart directive permits or denies clients from performing "restart" retrieve file transfers via the FTP REST command. By default this is enabled, so that clients may resume interrupted file transfers at a later time without losing previously collected data.

See also

AllowStoreRestart

Examples

AllowStoreRestart

Name

AllowStoreRestart -- Allow clients to resume uploads

Synopsis

AllowStoreRestart [`on|off`]

`Default`
: AllowStoreRestart off

`Context`
: server config, <VirtualHost>, <Anonymous>, <Directory>, <Global>, .ftpaccess

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The AllowStoreRestart directive permits or denies clients from "restarting" interrupted store file transfers (those sent from client to server). By default restarting (via the REST command) is not permitted when sending files to the server. Care should be taken to disallow anonymous ftp "incoming" transfers to be restarted, as this will allow clients to corrupt or increase the size of previously stored files (even if not their own).

The REST (Restart STOR) command is automatically blocked when HiddenStores is enabled, with the server returning a 501 error code to the client.

See also

AllowRetrieveRestart DeleteAbortedStores HiddenStores

Examples

AllowUser

Name

AllowUser -- User based allow rules

Synopsis

AllowUser [`["AND"|"OR"|"regex"] user-expression`]

`Default`
: None

`Context`
: <Limit>

`Module`
: mod_core

`Compatibility`
: 1.1.7 and later

Description

AllowUser specifies a user-expression that is specifically permitted access within the context of the <Limit> block it is applied to. user-expression has a similar syntax as that used in AllowGroup, in that it should contain a comma delimited list of users or "not" users (by prefixing a user name with the `!' character) that are to be allowed access to the block.

By default, the expression is parsed as a boolean "OR" list, meaning that ANY elements of the expression must evaluate to logically true in order to the explicit allow to apply. In order to treat the expression as a boolean "AND" list, meaning that ALL of the elements must evaluate to logically true, use the optional "AND" keyword. Similarly, to treat the expression as a regular expression, use the "regex" keyword.

See also

DenyUser AllowGroup DenyGroup

Examples

```
# A regular expression AllowUser directive
AllowUser regex ^ftp

# An AND-evaluated DenyUser directive
DenyUser AND system,test
```

AnonRatio

Name

AnonRatio -- Ratio directive

Synopsis

AnonRatio [`foo1 foo2 foo3`]

`Default`
: None known

`Context`
: <Directory>, <Anonymous>, <Limit>,.ftpaccess

`Module`
: mod_ratio

`Compatibility`
: at least 1.2.0 and later

Description

The AnonRatio directive ....

See also

AnonRatio

Examples

AnonRejectPasswords

Name

AnonRejectPasswords -- Block certain anonymous user passwords

Synopsis

AnonRejectePasswords [`regex`]

`Default`
: None

`Context`
: <Anonymous>

`Module`
: mod_auth

`Compatibility`
: 1.2.9rc1 and later

Description

The AnonRejectPasswords directive configures a regular expression filter for passwords given for anonymous logins. If the given anonymous password matches the configured regular expression, the anonymous login is denied.

See also

AnonRequirePassword

Examples

```
# Reject all <Anonymous> logins that use "evil.org" as part of the password
AnonRejectPasswords @evil\.org$
```

AnonRequirePassword

Name

AnonRequirePassword -- Make anonymous users supply a valid password

Synopsis

AnonRequirePassword [`on|off`]

`Default`
: AnonRequirePassword off

`Context`
: <Anonymous>

`Module`
: mod_auth

`Compatibility`
: 0.99.0 and later

Description

Normally, anonymous FTP logins do not require the client to authenticate themselves via the normal method of a transmitted cleartext password which is hashed and matched against an existing system user's password. Instead, anonymous logins are expected to enter their e-mail address when prompted for a password. Enabling the AnonRequirePassword directive requires anonymous logins to enter a valid password which must match the password of the user that the anonymous daemon runs as. However using AuthUsingAlias authentication can be matched against the password of the login username. This can be used to create "guest" accounts, which function exactly as normal anonymous logins do (and thus present a "chrooted" protected file system to the client), but require a valid password on the server's host system.

See also

AnonymousGroup AuthAliasOnly AuthUsingAlias

Anonymous

Name

Anonymous -- Define an anonymous server

Synopsis

Anonymous [`root-directory`]

`Default`
: None

`Context`
: server config,<VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The Anonymous configuration block is used to create an anonymous FTP login, and is terminated by a matching </Anonymous> directive. The root-directory parameters specifies which directory the daemon will first chdir to, and then chroot, immediately after login. Once the chroot operation successfully completes, higher level directories are no longer accessible to the running child daemon (and thus the logged in user). By default, proftpd assumes an anonymous login if the remote client attempts to login as the currently running user; unless the current user is root, in which case anonymous logins are not allowed regardless of the presence of an <Anonymous> block. To force anonymous logins to be bound to a user other than the current user, see the User and Group directives. In addition, if a User or Group directive is present in an <Anonymous> block, the daemon permanently switches to the specified uid/gid before chroot()ing. Normally, anonymous logins are not required to authenticate with a password, but are expected to enter a valid e-mail address in place of a normal password (which is logged). If this behavior is undesirable for a given <Anonymous> configuration block, it can be overridden via the AnonRequirePassword directive.

Note: Chroot()ed anonymous directories do not need to have supplemental system files in them, nor do they need to have any sort of specific directory structure. This is because proftpd is designed to acquire as much system information as possible before the chroot, and to leave open those files which are needed for normal operation and reside outside the new root directory.

See also

Examples

```
Example of a typical anonymous FTP configuration:

<Anonymous /home/ftp>
  # After anonymous login, daemon runs as user/group ftp.
  User ftp
  Group ftp

  # The client login 'anonymous' is aliased to the "real" user 'ftp'.
  UserAlias anonymous ftp

  # Deny write operations to all directories, except for 'incoming' where
  # 'STOR' is allowed (but 'READ' operations are prohibited)

  <Directory *>
    <Limit WRITE>
      DenyAll
    </Limit>
  </Directory>

  <Directory incoming>
    <Limit READ >
      DenyAll
    </Limit>
    <Limit STOR>
      AllowAll
    </Limit>
  </Directory>

</Anonymous>
```

AnonymousGroup

Name

AnonymousGroup -- Treat group members as anonymous users

Synopsis

AnonymousGroup [`group-expression`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.1.3 and later

Description

The AnonymousGroup directive specifies a group-expression to which all matching users will be considered anonymous logins. The group-expression argument is a boolean logically ANDed list of groups to which the user must be a member of (or non-member if the group name is prefixed with a `!' character). For more information on group-expressions see the DefaultRoot directive. If the authenticating user is matched by an AnonymousGroup directive, no valid password is required, and a special dynamic anonymous configuration is created, with the user's home directory as the default root directory. If a DefaultRoot directive also applies to the user, this directory is used instead of the user's home dir. Great care should be taken when using AnonymousGroup, as improper configuration can open up user home directories to full read/write access to the entire world.

See also

AuthAliasOnly AuthUsingAlias AnonRequirePassword DefaultRoot

Examples

AuthAliasOnly

Name

AuthAliasOnly -- Allow only aliased login names

Synopsis

AuthAliasOnly [`on|off`]

`Default`
: AuthAliasOnly off

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_auth

`Compatibility`
: 1.1.3 and later

Description

AuthAliasOnly restricts authentication to "aliased" logins only; i.e. those usernames provided by clients which are "mapped" to a real userid by the UserAlias directive. Turning AuthAliasOnly `on' in a particular context will cause proftpd to completely ignore all non-aliased logins for the entire context. If no contexts are available without AuthAliasOnly set to `on', proftpd rejects the client login and sends an appropriate message to syslog.

See also

AnonymousGroup AuthUsingAlias AnonRequirePassword UserAlias

Examples

AuthGroupFile

Name

AuthGroupFile -- Specify alternate group file

Synopsis

AuthGroupFile [`path`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_auth_file

`Compatibility`
: 1.0.3/1.1.1 and later

Description

AuthGroupFile specifies an alternate groups file, having the same format as the system /etc/group file, and if specified is used during authentication and group lookups for directory/access control operations. The path argument should be the full path to the specified file. AuthGroupFile can be configured on a per-VirtualHost basis, so that virtual FTP servers can each have their own authentication database (most often used in conjunction with AuthUserFile).

Note that this file need not reside inside a chroot()ed directory structure for Anonymous or DefaultRoot logins, as it is held open for the duration of client connections.

See also

AuthUserFile

Examples

AuthOrder

Name

AuthOrder -- Configure auth module checking order

Synopsis

AuthOrder [`module-name...`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.8rc1 and later

Description

The AuthOrder directive configures the names of auth modules, and the order in which they will be checked when authenticating a user.

At least one module name must be given; there is no maximum number of modules that can be listed. The listed module names must the full name of the source file, e.g. "mod_auth_unix.c". To see a full list of module names, use "proftpd -l". Do not use "mod_auth.c", as that module is the authentication front end module, and is necessary. Also, do not use "mod_auth_pam.c", as that module does not provide, by itself, all of the information proftpd needs.

Examples

```
# Use only AuthUserFiles when authenticating, and not the system's /etc/passwd
AuthOrder mod_auth_file.c

# If the user's information is not in LDAP, they're not a user to use
# this server.
AuthOrder mod_ldap.c

# Use SQL tables first, then LDAP, for authentication
AuthOrder mod_sql.c mod_ldap.c
```

AuthPAM

Name

AuthPAM -- Enable/Disable PAM authentication

Synopsis

AuthPAM [`on|off`]

`Default`
: on

`Context`
: server config,<VirtualHost>, <Global>

`Module`
: mod_auth_pam

`Compatibility`
: 1.2.0rc1 and later

Description

This directive determines whether PAM is used as an authentication method by ProFTPD. Enabled by default to fit in with the design policy of using PAM as the primary authentication mechanism.

See also

Examples

AuthPAMConfig

Name

AuthPAMConfig -- Select PAM service name

Synopsis

AuthPAMConfig [`service`]

`Default`
: ftp

`Context`
: server config,<VirtualHost>, <Global>

`Module`
: mod_auth_pam

`Compatibility`
: 1.2.0rc1 and later

Description

This directive allows you to specify the PAM service name used in authentication. PAM allows you to specify a service name to use when authenticating. This allows you to configure different PAM service names to be used for different virtual hosts. The directive was renamed from PAMConfig post 1.2.0 pre10.

See also

Examples

```
# Virtual host foobar authenticates differently than the rest

AuthPAMConfig foobar

# This assumes, that you have a PAM service named foobar
# configured in your /etc/pam.conf file or /etc/pam.d directory.
```

AuthUserFile

Name

AuthUserFile -- Specify alternate passwd file

Synopsis

AuthUserFile [`path`]

`Default`
: None

`Context`
: server config,<VirtualHost>, <Global>

`Module`
: mod_auth_file

`Compatibility`
: 1.0.3/1.1.1 and later

Description

AuthUserFile specifies an alternate passwd file, having the same format as the system /etc/passwd file, and if specified is used during authentication and user lookups for directory/access control operations. The path argument should be the full path to the specified file. AuthUserFile can be configured on a per-VirtualHost basis, so that virtual FTP servers can each have their own authentication database (most often used in conjunction with AuthGroupFile).

Note that this file need not reside inside a chroot()ed directory structure for Anonymous or DefaultRoot logins, as it is held open for the duration of client connections.

See also

AuthGroupFile

Examples

AuthUsingAlias

Name

AuthUsingAlias -- Authenticate via Alias-name instead of mapped username

Synopsis

AuthUsingAlias [`on|off`]

`Default`
: AuthUsingAlias off

`Context`
: <Anonymous>

`Module`
: mod_auth

`Compatibility`
: 1.2.0pre9 and later

Description

AuthUsingAlias disables the resolving of mapped usernames for authentication purposes. For example, if you have mapped the username anonymous to the "real" user ftp, the password gets checked against the user "anonymous". When AuthUsingAlias is disabled, the checked username would be "ftp".

See also

AnonymousGroup AuthAliasOnly AnonRequirePassword

Examples

```
An example of an Anonymous configuration using
AuthUsingAlias
# Basic Read-Only Anonymous Configuration.
<Anonymous /home/ftp>
UserAlias             anonymous  nobody
UserAlias             ftp        nobody
AuthAliasOnly         on
<Limit WRITE>
DenyAll
</Limit>
</Anonymous>
# Give Full Read-Write Anonymous Access to certain users
<Anonymous /home/ftp>
AnonRequirePassword   on
AuthAliasOnly         on
AuthUsingAlias        on
```

```
# The list of authorized users.
# user/pass lookup is for each user, not password entry
# of server uid ('nobody' in this example).
UserAlias             fred       nobody
UserAlias             joe        nobody
<Limit ALL>
AllowAll
</Limit>
</Anonymous>
```

Bind

Name

Bind -- Bind the server or Virtualhost to a specific IP address [deprecated]

Synopsis

Bind [`IP address`]

`Default`
: None

`Context`
: server config, <VirtualHost>

`Module`
: mod_core

`Compatibility`
: 1.1.6 - 1.3.0rc1

Description

Cause of too much confusion this directive has been deprecated with ProFTPD 1.3.0rc1. Please take a look at the VirtualHost and DefaultAddress directive. The Bind directive allows additional IP addresses to be bound to a main or VirtualHost configuration. Multiple Bind directives can be used to bind multiple addresses. The address argument should be either a fully qualified domain name or a numeric dotted-quad IP address. Incoming connections destined to an additional address added by Bind are serviced by the context containing the directive. Additionally, if SocketBindTight is set to on, a specific listen connection is created for each additional address.

See also

VirtualHost DefaultAddress

Examples

ByteRatioErrMsg

Name

ByteRatioErrMsg -- Ratio directive

Synopsis

ByteRatioErrMsg [`foo1 foo2 foo3`]

`Default`
: None known

`Context`
: <Directory>, <Anonymous>, <Limit>,.ftpaccess

`Module`
: mod_ratio

`Compatibility`
: at least 1.2.0 and later

Description

The ByteRatioErrMsg directive .... Example: ByteRatioErrMsg

See also

Examples

CapabilitiesEngine

Name

CapabilitiesEngine -- Enable/disable mod_cap

Synopsis

CapabilitiesEngine [`on off`]

`Default`
: CapabilitiesEngine On, if running on a Linux hosts that supports capabilities

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_cap

`Compatibility`
: 1.2.8rc1 and later

Description

The CapabilitiesEngine directive enables or disables the module's runtime capabilities engine. If set to off, this module does no runtime capabilities processing at all. Use this directive to disable the module.

CapabilitiesSet

Name

CapabilitiesSet -- Configure the set of Linux capabilities processed

Synopsis

CapabilitiesSet [`[+/-]capability...`]

`Default`
: CapabilitiesSet +CAP_CHOWN

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_cap

`Compatibility`
: 1.2.8rc1 and later

Description

By default, mod_cap removes all but two capabilities from the session-handling process: CAP_NET_BIND_SERVICE, for binding to ports lower than 1024 (required for active data transfers), and CAP_CHOWN, for allowing a process to change a file's ownership to a different user. The latter capability is only strictly necessary if the UserOwner configuration directive is in use; if not being used, the CAP_CHOWN capability is best removed. The CapabilitiesSet directive is used to manipulate the set of capabilities that mod_cap grants.

To remove a capability, prefix the name with a '-'; to enable a capability, use '+'. At present, this directive only supports one capability: CAP_CHOWN.

Example

<IfModule mod_cap.c> CapabilitiesEngine on CapabilitiesSet -CAP_CHOWN </IfModule>

CDPath

Name

CDPath -- Sets "search paths" for the cd command

Synopsis

CDPath [`directory`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.0pre2 and later

Description

Adds an entry to a search path that is used when changing directories. For example: CDPath /home/public CDPath /var/devel This allows a user to cd into any directory directly under /home/public or /var/devel, provided they have the appropriate rights. So, if /home/public/proftpd exists, cd proftpd will bring the user to that directory, regardless of where they currently are in the directory tree.

See also

Examples

Class

Name

Class -- Define a class of client connections

Synopsis

VirtualHost [`<Class name>`]

`Default`
: None

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 1.2.10rc1 and later

Description

When configuring proftpd, it is sometimes nice, or even necessary, to tag or label a client as belonging to some group, based on that client's IP address or DNS hostname. A "class" is the name for such connection-based groupings in ProFTPD terms. A class is defined to have a name, and as having certain criteria such as IP addresses, IP subnets/masks, and DNS hostnames. A client that connects to the daemon that has matching characteristics is then labeled as belonging to that class.

Within a <Class> section, the From directive is used to list the IP addresses, IP subnet/masks, and DNS names that make up the class.

See also

From

Examples

```
From 192.168.0.0/16
```

This defines a class named "internal"; any client connecting from 192.168.0.0/16 will belong to this class. And if you wanted to define a class for all clients not connecting from 192.168.0.0/16 address space:

```
From !192.168.0.0/16
```

A more complicated class might include matching DNS names as well:

From 1.2.3.4 From proxy.*.com From my.example.com From 5.6.7.8

CommandBufferSize

Name

CommandBufferSize -- Limit the maximum command length

Synopsis

CommandBufferSize [`size`]

`Default`
: 512

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.0pre7 and later

Description

The CommandBufferSize directive controls the maximum command length permitted to be sent to the server. This allows you to effectively control what the longest command the server may accept it, and can help protect the server from various Denial of Service or resource-consumption attacks.

See also

Examples

CreateHome

Name

CreateHome -- Create and populate users' home directories as needed

Synopsis

CreateHome [`off|on [<mode>] [skel <path>] [dirmode <mode>]`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_auth

`Compatibility`
: 1.2.8rc2 and later

Description

The CreateHome directive configures the server to automatically create a user's home directory, if that directory does not exist, during the login process.

The mode parameter is used to configure the absolute mode of the home directory created. If not specified, the module will default to 700.

The optional skel path parameter can be used to configure an /etc/skel-like directory containing account initialization files and directories. The parameter must be the full path to the directory. The directory must not be world-writeable. Files copied from this directory into the new home directory will have the UID and GID of the logging-in user. Note that sockets and FIFOs in the skeleton directory will not be copied; any setuid or setgid bits on files will be removed from the copied files in the target home directory.

The optional dirmode parameter can be used to specify the mode for intermediate directories that may need to be created in order to create the target home directory. By default, the mode for such intermediate directories will be 711. NOTE: using a mode that does not allow for the execute bit to be enabled can cause havoc. You have been warned.

Examples

Use the CreateHome default settings CreateHome on

Specify a skeleton directory CreateHome on skel /etc/ftpd/skel

No skeleton, but make sure that intermediate directories have 755 # permissions. CreateHome on dirmode

755

Skeleton directory, with 700 intermediate directories CreateHome on skel /etc/ftpd/skel dirmode 700

CreateHome

Name

CreateHome -- Create and populate users' home directories as needed

Synopsis

CreateHome [`off | on [<mode>] [skel <path>] [dirmode <mode>]`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_auth

`Compatibility`
: 1.2.8rc2 and later

Description

The CreateHome directive configures the server to automatically create a user's home directory, if that directory does not exist, during the login process.

The mode parameter is used to configure the absolute mode of the home directory created. If not specified, the module will default to 700.

The optional skel path parameter can be used to configure an /etc/skel-like directory containing account initialization files and directories. The parameter must be the full path to the directory. The directory must not be world-writeable. Files copied from this directory into the new home directory will have the UID and GID of the logging-in user. Note that sockets and FIFOs in the skeleton directory will not be copied; any setuid or setgid bits on files will be removed from the copied files in the target home directory.

The optional dirmode parameter can be used to specify the mode for intermediate directories that may need to be created in order to create the target home directory. By default, the mode for such intermediate directories will be 711. NOTE: using a mode that does not allow for the execute bit to be enabled can cause havoc. You have been warned.

Examples

Use the CreateHome default settings CreateHome on

Specify a skeleton directory CreateHome on skel /etc/ftpd/skel

No skeleton, but make sure that intermediate directories have 755 # permissions. CreateHome on dirmode

755

Skeleton directory, with 700 intermediate directories CreateHome on skel /etc/ftpd/skel dirmode 700

CwdRatioMsg

Name

CwdRatioMsg -- Ratio directive

Synopsis

CwdRatioMsg [`foo1 foo2 foo3`]

`Default`
: None known

`Context`
: <Directory>, <Anonymous>, <Limit>,.ftpaccess

`Module`
: mod_ratio

`Compatibility`
: at least 1.2.0 and later

Description

The CwdRatioMsg directive .... Example: CwdRatioMsg

See also

Examples

DebugLevel

Name

DebugLevel -- Set the debugging output level

Synopsis

DebugLevel [`level`]

`Default`
: DebugLevel 0

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.8rc1 and later

Description

The DebugLevel directive configures the debugging level the server will use when logging. The level parameter must be between 0 (lowest) and 10 (highest). This configuration directive will take precedence over any command-line debugging options used.

DefaultAddress

Name

DefaultAddress -- Set the address for the server to listen on

Synopsis

DefaultAddress [`dns-names|ip-addresses seperated with spaces`]

`Default`
: none

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 1.2.7rc1 and later

Description

This directive sets the the address the main server instance will bind to, the default behaviour is to select whatever IP the system reports as being the primary IP.

Starting with ProFTPD 1.3.0rc1 it's possible to use more than one FQDN or IP Address. With this change the old Bind directive has been deprecated.

See also

VirtualHost

Examples

ServerName "Default FTP Server"
Port 21

We want the main server instance to listen on a specific IP
DefaultAddress 192.168.10.30

Since 1.3.0rc1 it's also possible to use the following:
DefaultAddress 192.168.10.30 my.domain.tld

DefaultChdir

Name

DefaultChdir -- Set starting directory for FTP sessions

Synopsis

DefaultChdir [`directory [group-expression]`]

`Default`
: ~

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_auth

`Compatibility`
: 1.2.0pre2 and later

Description

Determines the directory a user is placed in after logging in. By default, the user is put in their home directory. The specified directory can be relative to the user's home directory. NOTE: If the specified directory is not available then DefaultChdir is treated as if it wasn't there in the first place. In particular, in this case the directory a user is placed in after logging in is determined by the other settings in proftpd.conf.

See also

DefaultRoot

Examples

DefaultRoot

Name

DefaultRoot -- Sets default chroot directory

Synopsis

DefaultRoot [`directory [group-expression]`]

`Default`
: DefaultRoot /

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_auth

`Compatibility`
: 0.99.0pl7 and later

Description

The DefaultRoot directive controls the default root directory assigned to a user upon login. If DefaultRoot is set to a directory other than "/", a chroot operation is performed immediately after a client authenticates. This can be used to effectively isolate the client from a portion of the host system filespace. The specified root directory must begin with a / or can be the magic character '~'; meaning that the client is chroot jailed into their home directory.

When the specified chroot directory is a symlink this will be resolved to it's parent first before setting up the chroot. This can have unwanted side effects. For example if a user has write access to the symlink he could modify it so that it points to '/'. Thus the chroot would be the root directory of the server, resulting in insufficient or no restrictions.

If the DefaultRoot directive specifies a directory which disallows access to the logged-in user's home directory, the user's current working directory after login is set to the DefaultRoot instead of their normal home directory. DefaultRoot cannot be used in <Anonymous> configuration blocks, as the <Anonymous> directive explicitly contains a root directory used for Anonymous logins. The special character '~' is replaced with the authenticating user's home directory immediately after login. Note that the default root may be a subdirectory of the home directory, such as "~/anon-ftp".

The optional group-expression argument can be used to restrict the DefaultRoot directive to a unix group, groups or subset of groups. The expression takes the format: [!]group-name1[,[!]group-name2[,...]]. The expression is parsed in a logical boolean AND fashion, such that each member of the expression must evaluate to logically TRUE in order for the DefaultRoot directive to apply. The special character '!' is used to negate group membership.

Care should be taken when using DefaultRoot. Chroot "jails" should not be used as methods for implementing general system security as there are potentially ways that a user can "escape" the jail.

See also

Examples

```
Example of a DefaultRoot configuration:

ServerName "A test ProFTPD Server"
ServerType inetd
User ftp
Group ftp
#
# This causes proftpd to perform a chroot into the authenticating user's directory
# immediately after login.
# Once this happens, the user is unable to "see" higher level directories.
# Because a group-expression is included, only users who are a member of
# the group 'users' and NOT a member of 'staff' will have their default
# root directory set to '~'.
DefaultRoot ~ users,!staff
...
```

DefaultServer

Name

DefaultServer -- Set the default server

Synopsis

DefaultServer [`on|off`]

`Default`
: DefaultServer off

`Context`
: server config,<VirtualHost>

`Module`
: mod_core

`Compatibility`
: 0.99.0pl6 and later

Description

The DefaultServer directive controls which server configuration is used as the default when an incoming connection is destined for an IP address which is neither the host's primary IP address or one of the addresses specified in a <VirtualHost> configuration block. Normally such "unknown" connections are issued a "no server available to service your request" message and disconnected. When DefaultServer is turned on for either the primary server configuration or a virtual server, all unknown destination connections are serviced by the default server. Only a single server configuration can be set to default.

See also

Examples

DefaultTransferMode

Name

DefaultTransferMode -- Set the default method of data transfer

Synopsis

DefaultTransferMode [`ascii|binary`]

`Default`
: DefaultTransferMode ascii

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.0pre9 and later

Description

DefaultTransferMode sets the default transfer mode of the server. By default, carriage-return/linefeed translation will be performed (ASCII mode).

See also

Examples

DeferWelcome

Name

DeferWelcome -- Don't show welcome message until user has authenticated

Synopsis

DeferWelcome [`DeferWelcome on|off`]

`Default`
: DeferWelcome off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The DeferWelcome directive configures a master or virtual server to delay transmitting the ServerName and address to new connections, until a client has successfully authenticated. If enabled, the initial welcome message will be exceedingly generic and will not give away any type of information about the host that the daemon is actively running on. This can be used by security-conscious administrators to limit the amount of "probing" possible from non-trusted networks/hosts.

See also

ServerIdent ServerName

Examples

Define

Name

Define -- Initialises Defines for IfDefine

Synopsis

Define [`parameter-name`]

`Default`
: none

`Context`
: any context

`Module`
: mod_core

`Compatibility`
: 1.2.6rc1 and later

Description

This directive is used to initialise defines for use in conjunction with the IfDefine directive

See also

IfDefine, IfModule

Examples

IfDefine LoadLimiting
IfDefine HighPerformanceSetup

DelayEngine

Name

DelayEngine -- Control the use of mod_delay

Synopsis

DelayEngine [`on|off`]

`Default`
: DelayEngine on

`Context`
: server config

`Module`
: mod_delay

`Compatibility`
: 1.3.0rc1 and later

Description

The DelayEngine directive enables or disables the module's runtime delaying calculations. If it is set to off this module does no delaying. Use this directive to disable the module.

See also

DelayTable

Examples

```
<IfModule mod_delay.c>
  DelayEngine off
</IfModule>
```

DelayTable

Name

DelayTable -- Sets the name and path of the file used as the timing table

Synopsis

DelayTable [`path`]

`Default`
: DelayTable var/proftpd/proftpd.delay

`Context`
: server config

`Module`
: mod_delay

`Compatibility`
: 1.3.0rc1 and later

Description

The DelayTable directive configures a path to a file that mod_delay uses for storing its timing data. The given path must be an absolute path. It is recommended that this file not be on an NFS mounted partition.

Note that timing data is kept across daemon stop/starts. When new <VirtualHost>s are added to the configuration, though, mod_delay will detect that it does not have a suitable DelayTable for the new configuration, and will clear all stored data.

See also

DelayEngine

Examples

DeleteAbortedStores

Name

DeleteAbortedStores -- Enable automatic deletion of partially uploaded HiddenStores files

Synopsis

DeleteAbortedStores [`DeleteAbortedStores on|off`]

`Default`
: off

`Context`
: server, <VirtualHost>, <Directory>, <Anonymous>, <Global>, .ftpaccess

`Module`
: mod_xfer

`Compatibility`
: 1.2.0rc2 and later

Description

The DeleteAbortedStores directive controls whether ProFTPD deletes partially uploaded HiddenStores files if the transfer is stopped via the ABOR command rather than a connection failure.

See also

HiddenStores

Examples

Deny

Name

Deny -- Access control directive

Synopsis

Deny [`Deny ["from"] "all"|"none"|host|network[,host|network[,...]]`

`Default`
: None

`Context`
: <Limit>

`Module`
: mod_core

`Compatibility`
: 0.99.0pl6 and later

Description

The Deny directive is used to create a list of hosts and/or networks which will explicitly be denied access to a given <Limit> context block. The magic keywords "ALL" and "NONE" can be used to indicate that all hosts are denied access, or that no hosts are explicitly denied (respectively). For more information on the syntax and usage of Deny see: Allow and Order.

See also

Allow Order Limit

Examples

DenyAll

Name

DenyAll -- Deny all clients

Synopsis

DenyAll [`DenyAll`]

`Default`
: None

`Context`
: <Directory>, <Anonymous>, <Limit>, .ftpaccess

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The DenyAll directive is analogous to a combination of "order deny,allow <cr> deny from all", with the exception that it has a higher precedence when parsed. It is provided as a convenient method of completely denying access to a directory, anonymous ftp or limit block. Because of its precedence, it should not be intermixed with normal Order/Deny directives. The DenyAll directive can be overridden at a lower level directory by using AllowAll. DenyAll and AllowAll are mutually exclusive.

See also

AllowAll

Examples

DenyClass

Name

DenyClass -- Class based deny rules

Synopsis

DenyClass [`["AND"|"OR"|"regex"] class-expression`]

`Default`
: None

`Context`
: <Limit>

`Module`
: mod_core

`Compatibility`
: 1.2.10rc1 and later

Description

DenyClass specifies a class-expression that is specifically denied access within the context of the <Limit> block it is applied to. class-expression has a similar syntax as that used in AllowGroup, in that it should contain a comma delimited list of classes or "not" classes (by prefixing a class name name with the `!' character) that are to be denied access to the block.

By default, the expression is parsed as a boolean "OR" list, meaning that ANY elements of the expression must evaluate to logically true in order to the explicit deny to apply. In order to treat the expression as a boolean "AND" list, meaning that ALL of the elements must evaluate to logically true, use the optional "AND" keyword. Similarly, to treat the expression as a regular expression, use the "regex" keyword.

See also

AllowUser DenyUser AllowGroup DenyGroup AllowClass

Examples

```
# A regular expression AllowClass directive
AllowClass regex ^known

# An AND-evaluated ClassUser directive
DenyClass AND bad,scanner
```

DenyFilter

Name

DenyFilter -- Regular expression of command arguments to be blocked

Synopsis

DenyFilter [`DenyFilter regular-expression`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>, <Anonymous>, <Directory>, .ftpaccess

`Module`
: mod_core

`Compatibility`
: 1.2.0pre7 and later

Description

Similar to AllowFilter, DenyFilter specifies a regular expression which must not match any of the command arguments. If the regex does match, a "Forbidden command" error is returned to the client. This can be especially useful for forbidding certain command argument combinations from ever reaching ProFTPD.

Notes: The 'PASV' command cannot be blocked using this directive.

See also

AllowFilter

Examples

```
# We don't want to allow any commands with % being sent to the server
DenyFilter "%"
```

DenyGroup

Name

DenyGroup -- Group based deny rules

Synopsis

DenyGroup [`["AND"|"OR"|"regex"] group-expression`]

`Default`
: None

`Context`
: <Limit>

`Module`
: mod_core

`Compatibility`
: 1.1.1 and later

Description

DenyGroup specifies a group-expression that is specifically denied within the context of the <Limit> block it is applied to. group-expression has the same format as that used in DefaultRoot, in that it should contain a comma separated list of groups or "not" groups (by prefixing a group name with the `!' character) that are to be denied access to the block.

By default, the expression is parsed as a boolean "AND" list, meaning that ALL elements of the expression must evaluate to logically true in order to the explicit deny to apply. In order to treat the expression as a boolean "OR" list, meaning that ANY of the elements must evaluate to logically true, use the optional "AND" keyword. Similarly, to treat the expression as a regular expression, use the "regex" keyword.

See also

DenyUser, AllowUser AllowGroup

Examples

```
# An OR-evaluated AllowGroup directive
AllowGroup OR www,doc

# A regular expression DenyGroup directive
DenyGroup regex ^sys
```

DenyUser

Name

DenyUser -- User based deny rules

Synopsis

DenyUser [`["AND"|"OR"|"regex"] user-expression`]

`Default`
: None

`Context`
: <Limit>

`Module`
: mod_core

`Compatibility`
: 1.1.7 and later

Description

DenyUser specifies a user-expression that is specifically denied within the context of the <Limit> block it is applied to. user-expression is a comma delimited list of users or "not" users (by prefixing a user name with the `!' character).

By default, the expression is parsed as a boolean "OR" list, meaning that ANY elements of the expression must evaluate to logically true in order to the explicit deny to apply. In order to treat the expression as a boolean "AND" list, meaning that ALL of the elements must evaluate to logically true, use the optional "AND" keyword. Similarly, to treat the expression as a regular expression, use the "regex" keyword.

See also

DenyGroup, AllowUser AllowGroup

Examples

```
# A regular expression AllowUser directive
AllowUser regex ^ftp

# An AND-evaluated DenyUser directive
DenyUser AND system,test
```

Directory

Name

Directory -- Directory-limited configuration directives

Synopsis

Directory [`<Directory pathname>`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

This directive creates a block of configuration directives which applies only to the specified directory and its sub-directories. The block is ended with </Directory>. Per-directory configuration is enabled during run-time with a "closest" match algorithm, meaning that the <Directory> directive with the closest matching path to the actual pathname of the file or directory in question is used. Per-directory configuration is inherited by all sub-directories until a closer matching <Directory> is encountered, at which time the original per-directory configuration is replaced with the closer match. Note that this does not apply to <Limit> </Limit> blocks, which are inherited by all sub-directories until a <Limit> block is reached in a closer match.

A trailing slash and wildcard ("/*") can be appended to the directory, specifying that the configuration block applies only to the contents (and sub-contents), not to the actual directory itself. Such wildcard matches always take precedence over non-wildcard <Directory> configuration blocks. <Directory> blocks cannot be nested (they are automatically nested at run-time based on their pathnames). Pathnames must always be absolute (except inside <Anonymous>), and should not reference symbolic links. Pathnames inside an <Anonymous> block can be relative, indicating that they are based on the anonymous root directory.

[Notes for ProFTPD 1.1.3 and later only] Pathnames that begin with the special character '~' and do not specify a username immediately after ~ are put into a special deferred mode. When in deferred mode, the directory context is not hashed and sorted into the configuration tree at boot time, but rather this hashing is deferred until a user authenticates, at which time the '~' character is replaced with the user's home directory. This allows a global <Directory> block which applies to all user's home directories, or sub-directories thereof.

See also

Limit

Examples

```
#Default usage of the directory directive
<Directory /users/robroy/private>
  HideNoAccess on
</Directory>

#Example with username-expanding
<Directory ~/anon-ftp>
  <Limit WRITE>
    DenyAll
  </Limit>
</Directory>
```

DirFakeGroup

Name

DirFakeGroup -- Hide real file/directory group

Synopsis

DirFakeGroup [`DirFakeGroup On|Off [groupname]`]

`Default`
: DirFakeGroup Off

`Context`
: server config, <VirtualHost>, <Global>, <Anonymous>, <Directory>, .ftpaccess

`Module`
: mod_ls

`Compatibility`
: 1.1.5

Description

DirFakeGroup can be used to hide the true group of files (including directories, fifos, etc.) in a directory listing. If simply turned On, DirFakeGroup will display all files as being owned by group 'ftp'. Optionally, the groupname argument can be used to specify a specific group other than 'ftp'. "~" can be used as the argument in order to display the primary group name of the current user.

Both DirFakeGroup and DirFakeUser are completely cosmetic; the groupname or username specified don't need to exist on the system, and neither directive affects permissions, real ownership or access control in any way.

See also

DirFakeUser DirFakeMode

Examples

DirFakeMode

Name

DirFakeMode -- Hide real file/directory permissions

Synopsis

DirFakeMode [`DirFakeMode octal-mode`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>, <Anonymous>, <Directory>

`Module`
: mod_ls

`Compatibility`
: 1.1.6

Description

The DirFakeMode directive configures a mode (or permissions) which will be displayed for ALL files and directories in directory listings. For each subset of permissions (user, group, other), the "execute" permission for directories is added in listings if the "read" permission is specified by this directive. As with DirFakeUser, and DirFakeGroup, the "fake" permissions shown in directory listings are cosmetic only, they do not affect real permissions or access control in any way.

See also

DirFakeUser DirFakeGroup

Examples

```
  DirFakeMode 0640

Will result in:

  -rw-r----- ... arbitrary.file
  drwxr-x--- ... arbitrary.directory
```

DirFakeUser

Name

DirFakeUser -- Hide real file/directory owner

Synopsis

DirFakeUser [`DirFakeUser On|Off [username]`]

`Default`
: DirFakeUser Off

`Context`
: server config, <VirtualHost>, <Global>, <Anonymous>, <Directory>, .ftpaccess

`Module`
: mod_ls

`Compatibility`
: 1.1.5

Description

DirFakeUser can be used to hide the true user owners of files (including directories, fifos, etc.) in a directory listing. If simply turned On, DirFakeUser will display all files as being owned by user 'ftp'. Optionally, the username argument can be used to specify a specific user other than 'ftp'. "~" can be used as the argument in order to display the current user's username.

Both DirFakeGroup and DirFakeUser are completely cosmetic; the groupname or username specified don't need to exist on the system, and neither directive affects permissions, real ownership or access control in any way.

See also

DirFakeGroup DirFakeMode

Examples

DisplayChdir

Name

DisplayChdir -- Set the file to display when entering a directory

Synopsis

DisplayChdir [`DisplayChdir filename [ true ]`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>, <Anonymous>, <Directory>

`Module`
: mod_core

`Compatibility`
: 1.3.1rc1 and later

Description

The DisplayFirstChdir directive configures an ASCII text filename which will be displayed to the user everytime he changes into a directory. If you would like to have the old behaviour of DisplayFirstChdir back you've to use the option "true". Then the file will only be displayed on the first time the user changes into the directory or if proftpd detects that its last modification time has changed since the previous CWD into a given directory. If the filename is relative, it is looked for in the new directory that the user has changed into. Note that for anonymous ftp logins (see <Anonymous>), the file must reside inside the chroot()ed file system space. If the file cannot be found or accessed, no error occurs and nothing is logged or displayed to the client.

DisplayChdir, DisplayConnect, DisplayLogin and DisplayQuit support the following "magic cookies" (only in 0.99.0pl10 and later), which are replaced with their respective strings before being displayed to the user.

```
%C      Current working directory

%E      Server admin's e-mail address

%F      Available space on file system, in bytes

%f      Available space on file system, with units

%i      The number of files uploaded (input) in this session

%K      Total number of bytes transferred

%k      Total number of bytes transferred, in units

%L      Local host name

%M      Max number of authenticated clients

%N      Current number of authenticated clients
```

```
%o      The number of files downloaded (output) in this session

%R      Remote host name

%T      Current Time

%t      The number of files transfered (uploaded and downloaded) in this session

%U      Username originally used in login

%u      Username reported by ident protocol

%V      Name of virtual host (if any)

%x      The name of the user's class

%y      Current number of connections from the user's class

%z      Max number of connections from the user's class

%{total_bytes_in} The number of bytes uploaded (input) in this session

%{total_bytes_out} The number of bytes downloaded (output) in this session

%{total_bytes_xfer} The number of bytes transferred (uploaded and downloaded) in this session

%(total_files_in} The number of files uploaded (input) in this session

%(total_files_out} The number of files downloaded (output) in this session

%(total_files_xfer} The number of files transferred (uploaded and downloaded) in this session
```

NOTE: not all of these may have a rational value, depending on the context in which they're used (e.g., %u if ident lookups are off).

See also

DisplayConnect DisplayLogin DisplayQuit

Examples

#Old way in the spirit of DisplayFirstChdir
DisplayChdir /home/ftp/filetodisplay true

DisplayConnect

Name

DisplayConnect -- Sets connect banner file

Synopsis

DisplayConnect [`DisplayConnect filename`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.0pre2 and later

Description

The DisplayConnect directive configures an ASCII text filename which will be displayed to the user when they initially connect but before they login. The filename can be either relative or absolute. In the case of a relative filename, the file is searched for starting in the home directory of the user the server is running as. As this can lead confusion, absolute pathnames are suggested. If the file cannot be found or accessed, no error occurs and nothing is logged or displayed to the client.

See also

DisplayFirstChdir

Examples

DisplayFileTransfer

Name

DisplayFileTransfer -- FIXFIXFIX

Synopsis

DisplayFileTransfer [`"name" limit|regex|ip value`]

`Default`
: FIXFIXFIX

`Context`
: server config, <Global>, <VirtualHost>, <Anonymous>, <Limit>, .ftpaccess

`Module`
: mod_xfer

`Compatibility`
: 1.3.1rc1 and later

Description

FIX FIX FIX

See also

Examples

FIXFIXFIX

FIXFIX

DisplayFirstChdir

Name

DisplayFirstChdir -- Set the file to display when first entering a directory [deprecated]

Synopsis

DisplayFirstChdir [`DisplayFirstChdir filename`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous>, <Directory>, <Global>

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later, magic cookies only in 0.99.0pl10 and later up to 1.3.1rc1

Description

This directive has been deprecated with ProFTPD 1.3.1rc1. Please use DisplayChdir instead.

The DisplayFirstChdir directive configures an ASCII text filename which will be displayed to the user the first time they change into a directory (via CWD) per a given session. The file will also be displayed if proftpd detects that its last modification time has changed since the previous CWD into a given directory. If the filename is relative, it is looked for in the new directory that the user has changed into. Note that for anonymous ftp logins (see <Anonymous>), the file must reside inside the chroot()ed file system space. If the file cannot be found or accessed, no error occurs and nothing is logged or displayed to the client.

DisplayFirstChdir, DisplayConnect, DisplayLogin and DisplayQuit support the following "magic cookies" (only in 0.99.0pl10 and later), which are replaced with their respective strings before being displayed to the user.

```
%C      Current working directory

%E      Server admin's e-mail address

%F      Available space on file system, in bytes

%f      Available space on file system, with units

%i      The number of files uploaded (input) in this session

%K      Total number of bytes transferred

%k      Total number of bytes transferred, in units

%L      Local host name
```

```
%M      Max number of authenticated clients

%N      Current number of authenticated clients

%o      The number of files downloaded (output) in this session

%R      Remote host name

%T      Current Time

%t      The number of files transfered (uploaded and downloaded) in this session

%U      Username originally used in login

%u      Username reported by ident protocol

%V      Name of virtual host (if any)

%x      The name of the user's class

%y      Current number of connections from the user's class

%z      Max number of connections from the user's class

%{total_bytes_in} The number of bytes uploaded (input) in this session

%{total_bytes_out} The number of bytes downloaded (output) in this session

%{total_bytes_xfer} The number of bytes transferred (uploaded and downloaded) in this session

%(total_files_in} The number of files uploaded (input) in this session

%(total_files_out} The number of files downloaded (output) in this session

%(total_files_xfer} The number of files transferred (uploaded and downloaded) in this session
```

NOTE: not all of these may have a rational value, depending on the context in which they're used (e.g., %u if ident lookups are off).

See also

DisplayChdir DisplayConnect DisplayLogin DisplayQuit

Examples

DisplayGoAway

Name

DisplayGoAway -- Set the file to display to a rejected connection

Synopsis

DisplayGoAway [`DisplayGoAway filename`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.0pre8 and later

Description

The DisplayGoAway directive specifies an ASCII text filename which will be displayed to the user if the class they're a member of has too many users logged in and their login request has been denied. DisplayGoAway supports the same "magic cookies" as DisplayFirstChdir.

See also

DisplayFirstChdir

Examples

DisplayLogin

Name

DisplayLogin -- Set the file to display on login

Synopsis

DisplayLogin [`DisplayLogin filename`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The DisplayLogin directive configures an ASCII text filename which will be displayed to the user when they initially login. The filename can be either relative or absolute. In the case of a relative filename, the file is searched for in the initial directory a user is placed in immediately after login (home directory for unix user logins, anonymous-root directory for anonymous logins). Note: that for jailed logins, the file must reside inside the chroot()ed file system space. If the file cannot be found or accessed, no error occurs and nothing is logged or displayed to the client. DisplayLogin supports the same "magic cookies" as DisplayFirstChdir.

See also

DisplayFirstChdir

Examples

DisplayQuit

Name

DisplayQuit -- Set the file to display on quit

Synopsis

DisplayQuit [`DisplayQuit filename`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.0pre8 and later

Description

DisplayQuit configures an ASCII text filename which will be displayed to the user when they quit. The filename can be either relative or absolute. In the case of a relative filename, the file is searched for in current directory a user is in when they logout -- for this reason, a absolute filename is usually preferable. NOTE: for jailed logins, the file must reside inside the chroot()ed file system space. If the file cannot be found or accessed, no error occurs and nothing is logged or displayed to the client. DisplayQuit supports the "magic cookies" listed under DisplayFirstChdir.

See also

DisplayFirstChdir

Examples

DisplayReadme

Name

DisplayReadme -- Enable display of file modification times on a file pattern

Synopsis

DisplayReadme [`DisplayReadme filename or pattern`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_readme

`Compatibility`
: 1.2.0pre8 and later

Description

Module: mod_readme The DisplayReadme directive notifies the user of the last change date of the specified file or pattern. Only a single DisplayReadme directive is allowed per configuration scope. DisplayReadme README Will result in: Please read the file README it was last modified on Sun Oct 17 10:36:14 1999 - 0 days ago Being displayed to the user on a cwd. DisplayReadmePattern README* Will result in: Please read the file README it was last modified on Tue Jan 25 04:47:48 2000 - 0 days ago Please read the file README.first it was last modified on Tue Jan 25 04:48:04 2000 - 0 days ago Being displayed to the user on a cwd.

See also

Examples

ExtendedLog

Name

ExtendedLog -- Specify custom logfiles

Synopsis

ExtendedLog [`filename [[command-classes] format-nickname]`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous> <Global>

`Module`
: mod_log

`Compatibility`
: 1.1.6pl1 and later

Description

The ExtendedLog directive allows customizable logfiles to be generated, either globally or per VirtualHost. The filename argument must contain an absolute pathname to a logfile which will be appended to when proftpd starts; the pathname should not be to a file in a nonexistent directory, to a world-writeable directory, or be a symbolic link (unless AllowLogSymlinks is set to on). Multiple logfiles (potentially with different command classes and formats) can be created. Optionally, the command-classes argument can be used to control which types of commands are logged. If not command classes are specified, proftpd logs all commands by default (passwords are hidden). command-classes is a comma delimited (no whitespace!) list of which commands to log.

The following are valid classes: NONE No commands AUTH Authentication commands (ACCT, PASS, REIN, USER) INFO Informational commands (FEAT, HELP, MDTM, QUIT, PWD, STAT, SIZE, SYST, XPWD) DIRS Directory commands (CDUP, CWD, LIST, MKD, NLST, RMD, XCWD, XCUP, XMKD, XRMD) READ File reading (RETR) WRITE File/directory writing or creation (APPE, MKD, RMD, RNFR, RNTO, STOR, STOU, XMKD, XRMD) MISC Miscellaneous commands (ABOR, ALLO, EPRT, EPSV, MODE, NOOP, OPTS, PASV, PORT, REST, RNFR, RNTO, SITE, SMNT, STRU, TYPE) SEC RFC2228-related security FTP commands ALL All commands (default)

If a format-nickname argument is supplied, ExtendedLog will use the predefined logformat (created by LogFormat). Otherwise, the default format of "%h %l %u %t \"%r\" %s %b" is used.

See also

AllowLogSymlinks, LogFormat, TransferLog

Examples

For example, to log all read and write operations to /var/log/ftp.log (using the default format), you could:

```
ExtendedLog /var/log/ftp.log read,write
```

FileRatioErrMsg

Name

FileRatioErrMsg -- (docs incomplete)

Synopsis

FileRatioErrMsg [`FileRatioErrMsg foo1 foo2 foo3`]

`Default`
: None known

`Context`
: <Directory>, <Anonymous>, <Limit>,.ftpaccess

`Module`
: mod_ratio

`Compatibility`
: at least 1.2.0 and later

Description

The FileRatioErrMsg directive .... Example: FileRatioErrMsg

See also

Examples

Global

Name

Global -- Set some directives to apply across the entire daemon

Synopsis

Global [`<Global>`]

`Default`
: None

`Context`
: server config, <VirtualHost>

`Module`
: mod_core

`Compatibility`
: 1.1.6 and later

Description

The Global configuration block is used to create a set of configuration directives which is applied universally to both the main server configuration and all VirtualHost configurations. Most, but not all other directives can be used inside a Global block.

In addition, multiple <Global> blocks can be created. At runtime, all Global blocks are merged together and finally into each server's configuration. Global blocks are terminated by a matching </Global> directive.

See also

Examples

Group

Name

Group -- Set the group the server normally runs as

Synopsis

Group [`Group groupid`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The Group directive configures which group the server daemon will normally run at. See User for more details.

See also

Examples

GroupOwner

Name

GroupOwner -- Change default group for new files and directories

Synopsis

GroupOwner [`GroupOwner groupname`]

`Default`
: None

`Context`
: <Anonymous>, <Directory>, .ftpaccess

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The GroupOwner directive configures which group all newly created directories and files will be owned by, within the context that GroupOwner is applied to. The group ID of groupname cannot be 0. Note that GroupOwner cannot be used to override the host OS/file system user/group paradigm. If the current user is not a member of the specified group, new files and directories will not be able to be chown()ed to the GroupOwner group. If this happens, file STOR (send file from client to server) and MKD/XMKD (mkdir) operations will succeed normally, however the new directory entries will be owned by the current user's default group (a warning message is also logged) instead of by the desired group. If you also use UserOwner in the same context, this restriction is lifted.

See also

UserOwner

Examples

GroupPassword

Name

GroupPassword -- Set a group-wide password

Synopsis

GroupPassword [`GroupPassword groupid hashed-password`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_auth

`Compatibility`
: 0.99.0pl5 and later

Description

The GroupPassword directive creates a special "group" password which allows all users in the specified group to authenticate using a single password. The group/password supplied is only effective inside the context to which GroupPassword is applied. The hashed-password argument is a standard cleartext password which has been passed through the standard unix crypt() library function. Extreme care should be taken when using GroupPassword, as serious security problems may arise if group membership is not carefully controlled.

See also

UserPassword

Examples

GroupRatio

Name

GroupRatio -- Ratio directive

Synopsis

GroupRatio [`GroupRatio foo1 foo2 foo3`]

`Default`
: None known

`Context`
: <Directory>, <Anonymous>, <Limit>,.ftpaccess

`Module`
: mod_ratio

`Compatibility`
: at least 1.2.0 and later

Description

The GroupRatio directive .... Example: GroupRatio

See also

Examples

HiddenStor

Name

HiddenStor -- Enables more safe file uploads [deprecated]

Synopsis

HiddenStor [`HiddenStor on|off`]

`Default`
: HiddenStor off

`Context`
: <Directory>, <Anonymous>, <VirtualHost>, <Global>

`Module`
: mod_xfer

`Compatibility`
: 1.2.0pre5 up to 1.3.1rc1

Description

This directive has been deprecated with ProFTPD 1.3.1rc1. Please use HiddenStores instead.

This directive is just an alias for HiddenStores and might be removed in the future entirely.

See also

HiddenStores

HiddenStores

Name

HiddenStores -- Enables more safe file uploads

Synopsis

HiddenStores [`HiddenStores on|off`]

`Default`
: HiddenStores off

`Context`
: server config, <Global>, <VirtualHost>, <Anonymous>, >

`Module`
: mod_xfer

`Compatibility`
: 1.2.7rc1 and later

Description

The HiddenStores directive enables two-step file uploads: files are uploaded as ".in.filename." and once the upload is complete, renamed to just "filename". This provides a degree of atomicity and helps prevent 1) incomplete uploads and 2) files being used while they're still in the progress of being uploaded.

Note: if the temporary file name is already in use (e.g., a server crash during upload), it will prevent the file from being uploaded

The REST (Restart STOR) command is automatically blocked when HiddenStores is enabled, with the server returning a 501 error code to the client.

See also

AllowStoreRestart DeleteAbortedStores

HideFiles

Name

HideFiles -- Enable hiding of files based on regular expressions

Synopsis

HideFiles [`[!]regexp|"none" ["user"|"group"|"class" expression]]`

`Default`
: None

`Context`
: <Directory>, .ftpaccess

`Module`
: mod_core

`Compatibility`
: 1.2.7rc1 and later

Description

The HideFiles directive configures a <Directory> section to hide all directory entries, e.g. its files and sub-directories, that match the given regular expression. These files can still be operated on by other FTP commands (DELE, RETR, etc), as constrained by any applicable <Limit>s, but this can be modified using the IgnoreHidden directive. Note that this directive manipulates a file's "hidden-ness", but doesn't do any hiding by itself. A <Limit> section, with IgnoreHidden enabled, does the actual hiding of the files from the <Limit>ed commands.

As <Directory> configurations are inherited by sub-directories, the "none" parameter can be used to disable any inherited file hiding within a sub-directory, usually through the use of a .ftpaccess file.

The optional parameters are used to restrict the rule for hiding files only to specific users. If "user" restriction is given, then expression is a user-expression specifying to which users the rule applies. Similarly for the "group" restriction. For the "class" restriction, the expression is simply the name of connection class for whom the rule will apply.

An unrestricted HideFiles directive and an unrestriected ShowFiles directive cannot be used simultaneously in the same context.

Examples:

```
# Hide configuration and passwd files from view
HideFiles "(\\.conf|passwd)$"

# ...or the same regex, without the quotes
HideFiles (\.conf|passwd)$

# Hide those same files from everyone _except_ a special user
```

```
HideFiles (\.conf|passwd)$ user !tj

# Using the ! prefix to "invert" the regular expression matching,
# allow only .txt and .html files to be seen
HideFiles !(\.txt|\.html)$

# Only let users of the webmaster group see HTML files, but nothing else
HideFiles !(\.htm|\.html)$ group webmaster
```

See Also: HideGroup, HideUser, HideNoAccess

HideGroup

Name

HideGroup -- Enable hiding of files based on group owner

Synopsis

HideGroup [`HideGroup groupid`]

`Default`
: None

`Context`
: <Directory>, <Anonymous>

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The HideGroup directive configures a <Directory> or < Anonymous> block to hide all directory entries owned by the specified group, unless the group is the primary group of the currently logged-in, authenticated user . Normally, hidden directories and files cannot be seen via LIST or NLST commands but can be operated on via other FTP commands (CWD, DELE, RETR, etc). This behavior can be modified via the IgnoreHidden directive.

See also

See Also: HideUser, HideNoAccess, IgnoreHidden

Examples

HideNoAccess

Name

HideNoAccess -- Block the listing of directory entries to which the user has no access permissions

Synopsis

HideNoAccess [`HideNoAccess on|off`]

`Default`
: None

`Context`
: <Directory>,<Anonymous>

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The HideNoAccess directive configures a <Directory> or <Anonymous> block to hide all directory entries in a directory listing (via the LIST or NLST FTP commands) to which the current logged-in, authenticated user has no access. Normal Unix-style permissions always apply, so that although a user may not be able to see a directory entry that has HideNoAccess applied, they will receive a normal "Permission denied" error message when attempting to blindly manipulate the file system object. The directory or file can be made completely invisible to all FTP commands by applying IgnoreHidden in conjunction with HideNoAccess.

See also

See Also: HideUser, HideGroup, IgnoreHidden

Examples

HideUser

Name

HideUser -- Enable hiding of files based on user owner

Synopsis

HideUser [`HideUser userid`]

`Default`
: None

`Context`
: <Directory>, <Anonymous>

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The HideUser directive configures a <Directory> or <Anonymous> block to hide all directory entries owned by the specified user, unless the owning user is the currently logged-in, authenticated user. Normally, hidden directories and files cannot be seen via LIST or NLST commands but can be operated on via other FTP commands (CWD, DELE, RETR, etc). This behavior can be modified via the IgnoreHidden directive.

See also

HideGroup, HideNoAccess, IgnoreHidden

Examples

HostRatio

Name

HostRatio -- Ratio directive

Synopsis

HostRatio [`HostRatio foo1 foo2 foo3`]

`Default`
: None known

`Context`
: <Directory>, <Anonymous>, <Limit>,.ftpaccess

`Module`
: mod_ratio

`Compatibility`
: at least 1.2.0 and later

Description

The HostRatio directive .... Example: HostRatio

See also

Examples

IdentLookups

Name

IdentLookups -- Toggle ident lookups

Synopsis

IdentLookups [`IdentLookups on|off`]

`Default`
: IdentLookups on

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.1.5 and later

Description

Normally, when a client initially connects to proftpd, the ident protocol (RFC1413) is used to attempt to identify the remote username. This can be controlled via the IdentLookups directive.

See also

Examples

IfDefine

Name

IfDefine -- To control the use of sections of the configuration

Synopsis

IfDefine [`[!]define-label`]

`Default`
: none

`Context`
: any

`Module`
: mod_core

`Compatibility`
: 1.2.6rc1 and later

Description

The <IfDefine test>...</IfDefine> section is used to mark directives that are conditional. The directives within an IfDefine section are only processed if the test is true. If the test is false, everything between the start and end markers is ignored.

The test in the <IfDefine> section directive can be one of two forms: 'parameter-name' or '!parameter-name'

In the former case, the directives between the start and end markers are only processed if the parameter named parameter-name is defined. The second format reverses the test, and only processes the directives if parameter-name is not defined.

The parameter-name argument is a define as given on the command line via -Dparameter-name, at the time the server was started.

<IfDefine> sections are nest-able, which can be used to implement simple multiple-parameter tests.

See also

Define, IfModule

Examples

$ proftpd -DDoSomething

```
--[ proftpd.conf ]--
<IfDefine DoSomething>
# do something here
</IfDefine>
--[ end ]--
```

IfModule

Name

IfModule -- Parse a section of config based on module name

Synopsis

IfModule [`[!]module-name`]

`Default`
: none

`Context`
: any

`Module`
: mod_core

`Compatibility`
: 1.2.6rc1 and later

Description

The <IfModule test>...</IfModule> section is used to mark directives that are conditional. The directives within an IfModule section are only processed if the test is true. If the test is false, everything between the start and end markers is ignored.

The test in the <IfModule> section directive can be one of two forms: "module name" or "!module name"

In the former case, the directives between the start and end markers are only processed if the module named module name is compiled in to ProFTPD. The second format reverses the test, and only processes the directives if module name is not compiled in.

The module name argument is a module name as given as the file name of the module, at the time it was compiled. For example, mod_sql.c.

<IfModule> sections are nest-able, which can be used to implement simple multiple-module tests.

See also

Define, IfDefine

Examples

```
<IfModule mod_load.c>
MaxLoad             10 "Access denied, server load too high"
```

</IfModule>

IgnoreHidden

Name

IgnoreHidden -- Treat 'hidden' files as if they don't exist

Synopsis

IgnoreHidden [`IgnoreHidden on|off`]

`Default`
: IgnoreHidden off

`Context`
: <Limit>

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

Normally, files hidden via HideNoAccess, HideUser or HideGroup can be operated on by all FTP commands (assuming Unix file permissions allow access), even though they do not appear in directory listings. Additionally, even when normal file system permissions disallow access, proftpd returns a "Permission denied" error to the client, indicating that the requested object does exist, even if it cannot be acted upon. IgnoreHidden configures a <Limit> block to completely ignore any hidden directory entries for the set of limited FTP commands. This has the effect of returning an error similar to "No such file or directory" when the client attempts to use the limited command upon a hidden directory or file.

See also

Examples

Include

Name

Include -- Load additional configuration directives from a file

Synopsis

Include [`Include file`]

`Default`
: None

`Context`
: server config, <Directory>, <Anonymous>, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.0 and later

Description

This directive allows you to include another configuration file within your current configuration file. The given file argument must be the full path to the file to be included.

See also

Examples

LDAPAliasDereference

Name

LDAPAliasDereference -- Specify how LDAP alias dereferencing is done

Synopsis

LDAPAliasDereference [`never` `find` `search` `always`]

`Default`
: LDAPAliasDereference never

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_ldap

`Compatibility`
: 2.8.16 and later

Description

Should be one of never, always, search, or find to specify that aliases are never dereferenced, always dereferenced, dereferenced when searching, or dereferenced only when locating the base object for the search.

Examples

LDAPAliasDereference always

LDAPAttr

Name

LDAPAttr -- Map LDAP Attributes to something non standard

Synopsis

LDAPAttr [`uid uidNumber gidNumber homeDirectory userPassword loginShell cn memberUid ftpQuota`] [`"NewAttribute"`]

`Default`
`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_ldap

`Compatibility`
: 2.8.13 and later

Description

FIXMEFIXMEFIXME

This dicrective has to be set before any of the LDAPDo* directives.

See also

Examples

FIXFIXFIX

FIXFIX

LDAPAuthBinds

Name

LDAPAuthBinds -- (docs incomplete)

Synopsis

Syntax: LDAPAuthBinds [`on off`]

(docs incomplete)

`Default`
: LDAPAuthBinds off in mod_ldap <= 2.7.6, LDAPAuthBinds on in mod_ldap >= 2.8

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.5 and later

Description

By default, the DN specified by LDAPDNInfo will be used to bind to the LDAP server to obtain user information, including the userPassword attribute. If LDAPAuthBinds is set to on, the DN specified by LDAPDNInfo will be used to fetch all user information except the userPassword attribute. Then, mod_ldap will bind to the LDAP server as the user who is logging in via FTP with the user-supplied password. If this bind succeeds, the user is considered authenticated and is allowed to log in. This method of LDAP authentication has the added benefit of supporting any password encryption scheme that your LDAP server supports.

See also

Examples

LDAPDefaultAuthScheme

Name

LDAPDefaultAuthScheme -- Set the authentication scheme/hash that is used when no leading {hashname} is present.

Synopsis

LDAPDefaultAuthScheme [`crypt clear`]

`Default`
: LDAPDefaultAuthScheme "crypt"

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.0 and later

Description

Specifies the authentication scheme used for passwords with no {prefix} in the LDAP database. For example, if you are using something like userPassword: mypass in your LDAP database, you would want to set LDAPDefaultAuthScheme to clear.

See also

Examples

LDAPDefaultGID

Name

LDAPDefaultGID -- Set the default GID to be assigned to users when no uidNumber attribute is found.

Synopsis

LDAPDefaultGID [`default-gid`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.0 and later

Description

This directive is useful primarily in virtual-user environments common in large-scale ISPs and hosting organizations. If a user does not have a LDAP gidNumber attribute, the LDAPDefaultGID is used. This allows one to have a large number of users in an LDAP database without gidNumber attributes; setting this configuration directive will automatically assign those users a single GID.

See also

Examples

LDAPDefaultUID

Name

LDAPDefaultUID -- Set the default UID to be assigned to users when no uidNumber attribute is found.

Synopsis

LDAPDefaultUID [`default-uid`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.0 and later

Description

This directive is useful primarily in virtual-user environments common in large-scale ISPs and hosting organizations. If a user does not have a LDAP uidNumber attribute, the LDAPDefaultUID is used. This allows one to have a large number of users in an LDAP database without uidNumber attributes; setting this configuration directive will automatically assign those users a single UID.

See also

Examples

LDAPDNInfo

Name

LDAPDNInfo -- Set DN information to be used for initial bind

Synopsis

LDAPDNInfo [`LDAPDNInfo "ldap-dn" "dn-password"`]

`Default`
: LDAPDNInfo "" "" (anonymous bind)

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.0 and later

Description

This directive specifies the LDAP DN and password to use when binding to the LDAP server. If this configuration directive is not specified, anonymous binds are used.

See also

Examples

LDAPDoAuth

Name

LDAPDoAuth -- Enable LDAP authentication

Synopsis

LDAPDoAuth [`on off`] [`"auth-base-dn"`] [`"search-filter-template"`]

`Default`
: LDAPDoAuth off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.0 and later

Description

This configuration directive activates LDAP authentication. The second argument to this directive is the LDAP base DN to use for authentication. The third argument is a template to be used for the search filter; %v will be replaced with the username that is being authenticated. By default, the search filter template "(&(uid=%v)(objectclass=posixAccount))" is used. The uid for the the search filter is taken from the LDAPAttr directive. Search filter templates are only supported in mod_ldap v2.7 and later.

See also

LDAPAttr

Examples

LDAPDoGIDLookups

Name

LDAPDoGIDLookups -- Enable LDAP lookups for user group membership and GIDs in directory listings

Synopsis

LDAPDoGIDLookups [`on` `off`] [`"gid-base-dn"`] [`"cn-filter-template"`] [`"gid-number-filter-template"`] [`"member-uid-filter-template"`]

`Default`
: LDAPDoGIDLookups off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.0 and later

Description

This configuration directive activates LDAP GID-to-name lookups in directory listings. The second argument to this directive is the LDAP base DN to use for GID-to-name lookups. The third through fifth arguments are templates to be used for the search filter; %v will be replaced with the GID that is being looked up.

By default, the search filter templates look like this:

cn_filter: "(cn=%v)(objectclass=posixGroup))", gidnumber_filter: "(gidNumber=%v)(objectclass=posixGroup))", memberuid_filter: "(memberUid=%v)(objectclass=posixGroup))".

The attribute names used in the default search filters are taken from the LDAPAttr directive.

Filter templates are only supported in mod_ldap v2.8.3 and later.

See also

LDAPAttr

Examples

LDAPDoQuotaLookups

Name

LDAPDoQuotaLookups -- Enable LDAP quota limit support

Synopsis

LDAPDoQuotaLookups [`on` `off`] [`"base-dn"`] [`"quota-filter-template"`] [`"default-quota"`]

`Default`
: LDAPDoQuotaLookups off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.8.12 and later

Description

This configuration directive activates LDAP quota lookups. The second argument to this directive is the LDAP base DN to use for quota limit search. The third argument is a template to be used for the search filter; %v will be replaced with the username that is being authenticated. By default, the search filter template "(&(LDAPAttr_uid=%v)(objectclass=posixAccount))" is used. The uid for the the search filter is taken from the LDAPAttr directive Search filter templates are only supported in mod_ldap v2.7 and later.

If specified, the `default-quota argument specifies the quota limits to use if a user does not have a ftpQuota attribute. This argument is formatted the same way as the ftpQuota LDAP attribute.`

See also

LDAPAttr

Examples

LDAPDoUIDLookups

Name

LDAPDoUIDLookups -- Enable LDAP lookups for UIDs in directory listings

Synopsis

LDAPDoUIDLookups [`on` `off`] [`"uid-base-dn"`] [`"uid-filter-template"`]

`Default`
: LDAPDoUIDLookups off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.0 and later

Description

This configuration directive activates LDAP UID-to-name lookups in directory listings. The second argument to this directive is the LDAP base DN to use for UID-to-name lookups. The third argument is a template to be used for the search filter; %v will be replaced with the UID that is being looked up. By default, the search filter template "(&(LDAPAttr_uidNumber=%v)(objectclass=posixAccount))" is used. The uid for the the search filter is taken from the LDAPAttr directive Search filter templates are only supported in mod_ldap v2.7 and later.

See also

LDAPAttr

Examples

LDAPForceDefaultGID

Name

LDAPForceDefaultGID -- Force all LDAP-authenticated users to use the same GID.

Synopsis

Syntax: LDAPForceDefaultGID [`on off`]

`Default`
: LDAPForceDefaultGID off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.8 and later

Description

Even when a LDAPDefaultGID is configured, mod_ldap will allow individual users to have gidNumber attributes that will override this default GID. With LDAPForceDefaultGID enabled, all LDAP-authenticated users are given the default GID; GIDs may not be overridden by gidNumber attributes.

See also

Examples

LDAPForceDefaultUID

Name

LDAPForceDefaultUID -- Force all LDAP-authenticated users to use the same UID.

Synopsis

Syntax: LDAPForceDefaultUID [`on off`]

`Default`
: LDAPForceDefaultUID off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.8 and later

Description

Even when a LDAPDefaultUID is configured, mod_ldap will allow individual users to have uidNumber attributes that will override this default UID. With LDAPForceDefaultUID enabled, all LDAP-authenticated users are given the default UID; UIDs may not be overridden by uidNumber attributes.

See also

Examples

LDAPForceGeneratedHomedir

Name

LDAPForceGeneratedHomedir -- Force all LDAP-authenticated users to use the default HomeDironDemand prefix/suffix.

Synopsis

LDAPForceGeneratedHomedir [`on` `off`] [`directory-mode`]

`Default`
: LDAPForceGeneratedHomedir off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.8.13 and later

Description

Even when a LDAPGenerateHomedirPrefix is configured, mod_ldap will allow individual users to have homeDirectory attributes that will override the default. With LDAPForceHomeDironDemand enabled, all LDAP-authenticated users are given the default prefix and/or suffix; homedirs may not be overridden by LDAP homeDirectory attributes.

See also

LDAPGenerateHomedir LDAPGenerateHomedirPrefix LDAPGenerateHomedirPrefixNoUsername

Examples

LDAPForceHomedirOnDemand

Name

LDAPForceHomedirOnDemand -- Force all LDAP-authenticated users to use the default HomeDironDemand prefix/suffix. [deprecated]

Synopsis

LDAPForceHomedirOnDemand [`on off`] [`directory-mode`]

`Default`
: LDAPForceHomedirOnDemand off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.8.11 and later

Description

This directive has been deprecated with mod_ldap v2.8.13. Please take a look at LDAPForceGenerateHomedir

Even when a LDAPHomeDironDemandPrefix is configured, mod_ldap will allow individual users to have homeDirectory attributes that will override the default. With LDAPForceHomeDironDemand enabled, all LDAP-authenticated users are given the default prefix and/or suffix; homedirs may not be overridden by LDAP homeDirectory attributes.

See also

LDAPForceGenerateHomedir

Examples

LDAPGenerateHomedir

Name

LDAPGenerateHomedir -- Enable the creation of user home directories on demand

Synopsis

LDAPGenerateHomedir [`on` `off`] [`directory-mode`]

`Default`
: LDAPGenerateHomedir off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.8.13 and later

Description

LDAPGenerateHomedir activates on-demand home directory creation. If a user logs in and does not yet have a home directory, a home directory is created automatically.

In mod_ldap <= 2.7.6, the home directory will be owned by the same user and group that ProFTPD runs as (see the User and Group configuration directives). mod_ldap >= 2.8 can create home directories for users with any UID/GID, not just those with the same UID/GID as the main ProFTPD server.

The second argument allows you to specify the mode (default permissions) to use when creating home directories on demand, subject to ProFTPD's umask (see the Umask directive). If no directory mode is specified, the default of 0755 is used. Directory mode setting is only supported in mod_ldap v2.7 or later.

See also

LDAPForceGeneratedHomedir LDAPGenerateHomedirPrefix LDAPGenerateHomedirPrefixNoUsername

Examples

%

LDAPGenerateHomedirPrefix

Name

LDAPGenerateHomedirPrefix -- Enable the creation of user home directories on demand

Synopsis

LDAPGenerateHomedirPrefix [`leading-path`]

`Default`
: LDAPGenerateHomedirPrefix off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.8.13 and later

Description

LDAPGenerateHomedirPrefix enables a prefix to be specified for on-demand home directory creation. This is most useful if mod_ldap is being used to authenticate against an LDAP directory that does not return a homeDirectory attribute, either because it cannot (Microsoft Active Directory, for example) or because you do not wish to extend your existing directory schema.

For example, setting this directive to "/home" and logging in as the user "joe" would result in his home directory being created as "/home/joe". The directory will be created with the mode specified in LDAPGenerateHomedir. To use this directive, LDAPGenerateHomedir must be enabled.

See also

LDAPForceGeneratedHomedir LDAPGenerateHomedir LDAPGenerateHomedirPrefixNoUsername

Examples

LDAPGenerateHomedirPrefixNoUsername

Name

LDAPGenerateHomedirPrefixNoUsername -- (docs incomplete)

Synopsis

LDAPGenerateHomedirPrefixNoUsername [`on` `off`]

`Default`
: (docs incomplete)

`Context`
: server config, <Global>, <VirtualHost>, <Anonymous>, <Limit>, .ftpaccess

`Module`
: mod_ldap

`Compatibility`
: mod_ldap 2.8.13 and later

Description

(docs incomplete)

See also

LDAPForceGeneratedHomedir LDAPGenerateHomedir LDAPGenerateHomedirPrefix

LDAPHomedirOnDemand

Name

LDAPHomedirOnDemand -- Enable the creation of user home directories on demand [deprecated]

Synopsis

LDAPHomedirOnDemand [`on off`] [`directory-mode`]

`Default`
: LDAPHomedirOnDemand off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.0 and later

Description

This directive has been deprecated with mod_ldap v2.8.13. Please take a look at LDAPGenerateHomedir

LDAPHomedirOnDemand activates on-demand home directory creation. If a user logs in and does not yet have a home directory, a home directory is created automatically.

In mod_ldap <= 2.7.6, the home directory will be owned by the same user and group that ProFTPD runs as (see the User and Group configuration directives). mod_ldap >= 2.8 can create home directories for users with any UID/GID, not just those with the same UID/GID as the main ProFTPD server.

The second argument allows you to specify the mode (default permissions) to use when creating home directories on demand, subject to ProFTPD's umask (see the Umask directive). If no directory mode is specified, the default of 0755 is used. Directory mode setting is only supported in mod_ldap v2.7 or later.

See also

LDAPGenerateHomedir

Examples

LDAPHomedirOnDemandPrefix

Name

LDAPHomedirOnDemandPrefix -- Enable the creation of user home directories on demand [deprecated]

Synopsis

LDAPHomedirOnDemandPrefix [`leading-path`]

`Default`
: LDAPHomedirOnDemandPrefix off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.8 and later

Description

This directive has been deprecated with mod_ldap v2.8.13. Please take a look at LDAPGenerateHomedirPrefix

LDAPHomedirOnDemandPrefix enables a prefix to be specified for on-demand home directory creation. This is most useful if mod_ldap is being used to authenticate against an LDAP directory that does not return a homeDirectory attribute, either because it cannot (Microsoft Active Directory, for example) or because you do not wish to extend your existing directory schema.

For example, setting this directive to "/home" and logging in as the user "joe" would result in his home directory being created as "/home/joe". The directory will be created with the mode specified in LDAPHomedirOnDemand. To use this directive, LDAPHomedirOnDemand must be enabled.

See also

LDAPGenerateHomedirPrefix

Examples

LDAPHomedirOnDemandPrefixNoUsername

Name

LDAPHomedirOnDemandPrefixNoUsername -- (docs incomplete)

Synopsis

LDAPHomedirOnDemandPrefixNoUsername [`"name" limit|regex|ip value`]

`Default`
: (docs incomplete)

`Context`
: server config, <Global>, <VirtualHost>, <Anonymous>, <Limit>, .ftpaccess

`Module`
: mod_ldap

`Compatibility`
: 1.2.5rc1 and later

Description

This directive has been deprecated with mod_ldap v2.8.13. Please take a look at LDAPGenerateHomedirPrefixNoUsername

(docs incomplete)

See also

LDAPGenerateHomedirPrefixNoUsername

LDAPHomedirOnDemandSuffix

Name

LDAPHomedirOnDemandSuffix -- Specify an additional directory to be created inside a user's home directory on demand. [deprecated]

Synopsis

LDAPHomedirOnDemandSuffix [`additional-directory1 additional-directory2 additional-directory3`]

`Default`
: LDAPHomedirOnDemandSuffix ""

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.6 and later.

Description

This directive is deprecated and was removed from mod_ldap v2.8.13. It has no replacement option.

to be created within a user's home directory when it is created on demand. For example, if a user's home directory is "/home/user", setting this configuration directive to "public_html" will also create "/home/user/public_html" on demand. In mod_ldap v2.7.6 and earlier, you must also activate LDAPHomedirOnDemand in your configuration.

mod_ldap >= 2.8 supports multiple suffix arguments and does not require LDAPHomedirOnDemand to be enabled.

mod_ldap >= 2.8.11 supports additional mode information; you can add ":octal-mode" to a directory argument to have it created with that mode. For example, LDAPHomedirOnDemandSuffix foo:700 will create the suffix directory foo with the mode 700.

See also

Examples

LDAPNegativeCache

Name

LDAPNegativeCache -- Enable negative caching for LDAP lookups

Synopsis

LDAPNegativeCache [`on` `off`]

`Default`
: LDAPNegativeCache off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v1.1 and later

Description

LDAPNegativeCache specifies whether or not to cache negative responses from the LDAP server when using LDAP for UID/GID lookups. This option is useful if you also use/are in transition from another authentication system; if there are many users in your old authentication system that aren't in the LDAP database, there can be a significant delay when a directory listing is performed as the UIDs not in the LDAP database are repeatedly looked up in an attempt to present usernames instead of UIDs in directory listings. With LDAPNegativeCache set to on, negative ("not found") responses from the LDAP server will be cached and speed will improve on directory listings that contain many users not present in the LDAP database.

See also

Examples

LDAPProtocolVersion

Name

LDAPProtocolVersion -- Set the LDAP protocol version

Synopsis

LDAPProtocolVersion [2 | 3]

`Default`
: 3

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_ldap

`Compatibility`
: 2.8.13 and later

Description

FIX FIX FIX

See also

Examples

FIXFIXFIX

FIXFIX

LDAPQueryTimeout

Name

LDAPQueryTimeout -- Set a timeout for LDAP queries

Synopsis

LDAPQueryTimeout [`timeout-seconds`]

`Default`
: LDAPQueryTimeout default-api-timeout

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.0 and later

Description

Sets the timeout used for LDAP directory queries. The default is the default timeout used by your LDAP API.

See also

Examples

LDAPSearchScope

Name

LDAPSearchScope -- Specify the search scope used in LDAP queries

Synopsis

LDAPSearchScope [`onelevel` `subtree`]

`Default`
: LDAPSearchScope subtree

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.6 and later

Description

Set the scope used for LDAP searches. The default setting, subtree, searches for all entries in the tree from the current level down. Setting this directive to onelevel searches only one level deep in the LDAP tree.

See also

Examples

LDAPServer

Name

LDAPServer -- Specify the LDAP server to use for lookups

Synopsis

LDAPServer [`"hostname1:port1 hostname2:port2"`]

`Default`
: LDAPServer "localhost"

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v1.0 and later

Description

LDAPServer allows you to to specify the hostname(s) and port(s) of the LDAP server(s) to use for LDAP authentication. If no LDAPServer configuration directive is present, the default LDAP servers specified by your LDAP API will be used.

See also

Examples

LDAPUseTLS

Name

LDAPUseTLS -- Enable TLS/SSL connections to the LDAP server.

Synopsis

Syntax: LDAPUseTLS [`on off`]

`Default`
: LDAPUseTLS off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_ldap

`Compatibility`
: mod_ldap v2.8 and later

Description

By default, mod_ldap connects to the LDAP server via a non-encrypted connection. Enabling this option causes mod_ldap to use an encrypted (TLS/SSL) connection to the LDAP server. If a secure connection to the LDAP server fails, mod_ldap will not authenticate users (mod_ldap will *not* fall back to an unsecure connection).

See also

Examples

LeechRatioMsg

Name

LeechRatioMsg -- Sets the 'over ratio' error message

Synopsis

LeechRatioMsg [`LeechRatioMsg foo1 foo2 foo3`]

`Default`
: None known

`Context`
: <Directory>, <Anonymous>, <Limit>,.ftpaccess

`Module`
: mod_ratio

`Compatibility`
: at least 1.2.0 and later

Description

The LeechRatioMsg directive defines the response message sent back to the client upon breaking their quota limits.

See also

Examples

```
LeechRatioMsg "please upload as well as download"
```

Limit

Name

Limit -- Set the commands/actions to be controlled

Synopsis

Limit [`<Limit command|command-group [command2 ..]>`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Directory>, <Anonymous>, <Global>, .ftpaccess

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The Limit configuration block is used to place access restrictions on one or more FTP commands, within a given context. Limits flow downward, so that a Limit configuration in the server config context applies to all <Directory> and <Anonymous> blocks that also reside in the configuration; until it is overridden by a "lower" <Limit> block. Any number of command parameters can be specified, against which the contents of the <Limit> block will be applied. command can be any valid FTP command, but is generally one of the following: CWD (Change Working Directory) Sent by client when changing directories. MKD / XMKD (MaKe Directory) Sent by client to create a new directory. RNFR (ReName FRom), RNTO (ReName TO) Sent as a pair by client to rename a directory entry. DELE (DELEte) Sent by client to delete a file. RMD / XRMD (ReMove Directory) Sent by client to remove a directory. RETR (RETRieve) Transfer a file from the server to the client. STOR (STORe) Transfer a file from the client to the server. In addition, the following command-groups are accepted. They have a lower precedence than real commands, meaning that a real command limit will always be applied instead of the command-group. READ All FTP commands which deal with file reading (directory listing not included): RETR, SITE, SIZE, STAT WRITE All FTP commands which deal with file or directory write/creation/deletion: APPE, DELE, MKD, RMD, RNTO, STOR, XMKD, XRMD DIRS All FTP commands which deal with directory listing: CDUP, CWD, LIST, MDTM, NLST, PWD, RNFR, XCUP, XCWD, XPWD ALL ALL FTP commands (identical to READ WRITE DIRS). Note this group has the lowest precedence of all; it will not override a limit imposed by another command-group (e.g. DIRS). Finally, a special command is allowed which can be used to control login access: LOGIN Connection or login to the server. Applying a <Limit> to this pseudo-command can be used to allow or deny initial connection or login to the context. It has no effect, and is ignored, when used in a context other than server config, <VirtualHost> or <Anonymous> (i.e. using it in a <Directory> context is meaningless). <Limit> command restrictions should not be confused with file/directory access permission. While limits can be used to restrict a command on a certain directory, they cannot be used to override the file permissions inherent to the base operating/file system. The following FTP commands cannot be restricted via <Limit>: ABOR HELP MODE (not implemented, always S) NOOP PASS (use <Limit LOGIN>) PASV PORT QUIT REST (use AllowRetrieveRestart, AllowStoreRestart) STRU (not implemented, always F) SYST TYPE

USER (use <Limit LOGIN>)

See also

See Also: IgnoreHidden

Examples

ListOptions

Name

ListOptions -- Configure options used when listing directories

Synopsis

ListOptions ["options string"][["strict"]]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>, <Directory>, .ftpaccess

`Module`
: mod_ls

`Compatibility`
: 1.2.8rc1 and later

Description

Normally, FTP commands involving directory listings (NLST, LIST and STAT) use the arguments (options) passed by the client to determine what files are displayed and the format they are displayed in. The ListOptions directive can alter the behaviour of such listings by making it such that a certain option (or options) is always in effect, or is always disabled.

In addition to the normal dash-prefixed options that the builtin ls takes, the directive allows for plus-prefixed options. The plus-prefixed options allow for their dash-prefixed equivalents, potentially given by a user, to be disabled, while still allowing other options to function normally.

```
-1 List one file per line

-A List all files except "." and ".."

-a List all files including those whose names start with "."

-C List entries by columns

-d List directory entries instead of directory contents

-F Append file type indicator (one of "*", "/", "=", "@" or "|") to names

-h Print file sizes in human-readable format (e.g. 1K, 234M, 2G)

-L List files pointed to by symlinks

-l Use a long listing format

-n List numeric UIDs/GIDs instead of user/group names

-R List subdirectories recursively
```

```
-r Sort filenames in reverse order

-S Sort by file size

-t Sort by modification time
```

If the optional "strict" keyword is used, then the configured options will override any options given by the user (i.e. the user's options will be ignored). In addition to "strict" the following keywords are supported:

```
maxfiles Sets a maximum limit on the number of files listed in one directory listing

maxdirs Sets a maximum limit on the number of directories listed in one directory listing

maxdepth Sets a maximum recursion depth, if the -R option is allowed
```

See also

Examples

Force directory listings to always show dotfiles ListOptions "-a"

To prevent anyone from doing recursive listings, but still allowing # other user options, use +R to disable any -R option given by users ListOptions "+R"

To allow only the basic listing, no options, always ListOptions "" strict

#limit maximum files given back to 2000 and recurse in to a max #depth of 3 directories ListOptions -a maxfiles 2000 maxdepth 3

LogFormat

Name

LogFormat -- Specify a logging format

Synopsis

LogFormat [`LogFormat nickname "format-string"`]

`Default`
: LogFormat default "%h %l %u %t \"%r\" %s %b"

`Context`
: server config

`Module`
: mod_log

`Compatibility`
: 1.1.6pl1 and later

Description

The LogFormat directive can be used to create a custom logging format for use with the ExtendedLog directive. Once created, the format can be referenced by the specified nickname. The format-string argument can consist of any combination of letters, numbers and symbols. The special character % is used to start a meta-sequence (see below). To insert a literal % character, use %%.

The following meta sequences are available and are replaced as indicated when logging.

```
%a              Remote client IP address
%A              Anonymous username (password given), or UNKNOWN if non-anonymous
%b              Bytes sent for request
%d              Directory name (not full path) for CDUP, CWD, MKD, RMD, XCWD, XCUP, XMKD, XRMD
%D              Directory name (full path) for CDUP, CWD, MKD, RMD, XCWD, XCUP, XMKD, XRMD
%{FOOBAR}e        Contents of environment variable FOOBAR.  Note that the server does not set any
%f                Filename stored or retrieved, absolute path (not chrooted)
%F                Filename stored or retrieved, as the client sees it
%h                Remote client DNS name
%J                Command arguments received from client, e.g. file.txt
%l                Remote username (from ident), or UNKNOWN if ident lookup failed
%L                Local server IP address
%m                Command (method) name received from client, e.g. RETR
%p                Local server port number
%P                Local server process id (pid)
%r                Full command line received from client
%s                Numeric FTP response code (status)
%S                Response message send from the client (avaible since v1.3.1rc1)
%t                Current local time
%{format}t        Current local time formatted (strftime(3) format)
%T                Time taken to transmit/receive file, in seconds
%u                Local authenticated userid
%U                USER name originally sent by the client
```

```
%v                ServerName of server handling session
%V                DNS name of server handling session
%{version}        Print ProFTPD Version
```

See also

ExtendedLog, TransferLog

Examples

LoginPasswordPrompt

Name

LoginPasswordPrompt -- Configure to display the passwort prompt or not

Synopsis

LoginPasswordPrompt [`LoginPasswordPrompt on|off`]

`Default`
: LoginPasswordPrompt on

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_auth

`Compatibility`
: 1.2.0pre1 and later

Description

If set to off, ProFTPd will skip the password request if the login will be denied regardless of password, e.g., if a <Limit LOGIN> directive forbids the connection.

See also

Examples

MasqueradeAddress

Name

MasqueradeAddress -- Configure the server address presented to clients

Synopsis

MasqueradeAddress [`MasqueradeAddress ip-address|dns-hostname`]

`Default`
: none

`Context`
: server config, <VirtualHost>

`Module`
: mod_core

`Compatibility`
: 1.2.2 and later

Description

MasqueradeAddress causes the server to display the network information for the specified IP address or DNS hostname to the client, on the assumption that that IP address or DNS host is acting as a NAT gateway or port forwarder for the server.

See also

Examples

MasqueradeAddress nat-gw.mydomain.com

MaxClients

Name

MaxClients -- Limits the number of users that can connect

Synopsis

MaxClients [`MaxClients number|none [message]`]

`Default`
: MaxClients none

`Context`
: server config, <Anonymous>, <VirtualHost>, <Global>

`Module`
: mod_auth

`Compatibility`
: 0.99.0 and later

Description

The MaxClients directive configures the maximum number of authenticated clients which may be logged into a server or anonymous account. Once this limit is reached, additional clients attempting to authenticate will be disconnected. The special value none may be supplied which removes all maximum connection limits from the applicable configuration context. Additionally, an optional message argument may be used which will be displayed to a client attempting to exceed the maximum value; immediately before disconnection. The message argument is parsed for the magic string "%m", which is replaced with the configured maximum value. If message is not supplied, a system-wide default message is used. Example: MaxClients 5 "Sorry, the maximum number of allowed users are already connected (%m)" Results in: 530 Sorry, the maximum number of allowed users are already connected (5)

See also

Examples

MaxClientsPerClass

Name

MaxClientsPerClass -- Limit the number of connections per class

Synopsis

MaxClientsPerClass [`MaxClientsPerClass name number|"none" [message]`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_auth

`Compatibility`
: 1.2.10rc1 and later

Description

The MaxClientsPerClass directive configures the maximum number of clients that may be connected at any given time from the same Class. The optional argument message may be used which will be displayed to a client attempting to exceed the maximum value. If message is not supplied, a default message of "Sorry, the maximum number of clients (%m) from your class are already connected."

See also

MaxClients, MaxClientsPerHost MaxClientsPerUser MaxHostsPerUser

Examples

```
MaxClientsPerClass foo 1 "Only one such client at a time."
Results in: 530 Only one such client at a time.
```

MaxClientsPerHost

Name

MaxClientsPerHost -- Limits the connections per client machine

Synopsis

MaxClientsPerHost [`MaxClientsPerHost number|none [message]`]

`Default`
: MaxClientsPerHost none

`Context`
: server config, <Anonymous>, <VirtualHost>, <Global>

`Module`
: mod_auth

`Compatibility`
: 1.1.7 and later

Description

The MaxClientsPerHost directive configures the maximum number of clients allowed to connect per host. The optional argument message may be used which will be displayed to a client attempting to exceed the maximum value. If message is not supplied, a default message of "Sorry, the maximum number clients (%m) from your host are already connected." is used.

See also

MaxClients, MaxHostsPerUser

Examples

```
MaxClientsPerHost 1 "Sorry, you may not connect more than one time."
Results in: 530 Sorry, you may not connect more than one time.
```

MaxClientsPerUser

Name

MaxClientsPerUser -- Limit the number of connections per userid

Synopsis

MaxClientsPerUser [`MaxClientsPerUser number|none [message]`]

`Default`
: MaxClientsPerUser none

`Context`
: server config, <VirtualHost>, <Global>, <Anonymous>

`Module`
: mod_auth

`Compatibility`
: 1.2.7rc1 and later

Description

The MaxClientsPerUser directive configures the maximum number of clients that may be connected at any given time using the same user name. The optional argument message may be used which will be displayed to a client attempting to exceed the maximum value. If message is not supplied, a default message of "Sorry, the maximum number of clients (%m) for this user already connected."

See also

MaxClients, MaxClientsPerHost MaxHostsPerUser

Examples

```
MaxClientsPerUser 1 "Only one such user at a time."
Results in: 530 Only one such user at a time.
```

MaxConnectionRate

Name

MaxConnectionRate -- Maximum TCP socket connection rate

Synopsis

MaxConnectionRate [`connections per second`]

`Default`
: none

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 1.2.7rc1 and later

Description

Set the maxiumum rate at which new TCP connections are accepted, this applies to the entire server, therefore too low a value on a high traffic server can result in all VirtualHosts being made unavailable due to normal traffic levels.

The value is the number of connections in a given second at which the block comes into effect, thus a value of "1" will result in all connections being blocked.

See also

Examples

MaxConnectionRate 4

MaxConnectionsPerHost

Name

MaxConnectionsPerHost -- Limits the unauthenticated connections per client machine

Synopsis

MaxConnectionsPerHost [`MaxConnectionsPerHost number|none [message]`]

`Default`
: MaxConnectionsPerHost none

`Context`
: server config, <Anonymous>, <VirtualHost>, <Global>

`Module`
: mod_auth

`Compatibility`
: 1.2.11rc1 and later

Description

The MaxConnectionsPerHost directive configures the maximum number of unauthenticated clients allowed to connect per host. The optional argument message may be used which will be displayed to a client attempting to exceed the maximum value. If message is not supplied, a default message of "Sorry, the maximum number of connections (%m) from your host are already connected." is used.

See also

MaxClients, MaxClientsPerHost, MaxHostsPerUser

Examples

```
MaxConnectionsPerHost 1 "Sorry, you may not connect more than one time."
Results in: 530 Sorry, you may not connect more than one time.
```

MaxHostsPerUser

Name

MaxHostsPerUser -- Limit the number of connections per userid

Synopsis

MaxHostsPerUser [`MaxHostsPerUser number|none [message]`]

`Default`
: MaxHostsPerUser none

`Context`
: server config, <Anonymous>, <VirtualHost>, <Global>

`Module`
: mod_auth

`Compatibility`
: 1.2.4 and later

Description

The MaxHostsPerUser directive configures the maximum number of times different hosts, using a given login, can connect at any given time. The optional argument message may be used which will be displayed to a client attempting to exceed the maximum value. If message is not supplied, a default message of "Sorry, the maximum number of hosts (%m) for this user already connected."

See also

MaxClients, MaxClientsPerHost

Examples

```
MaxHostsPerUser 1 "Sorry, you may not connect more than one time."
Results in: 530 Sorry, you may not connect more than one time.
```

MaxInstances

Name

MaxInstances -- Sets the maximum number of child processes to be spawned

Synopsis

MaxInstances [`MaxInstances number`]

`Default`
: MaxInstances none

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 1.1.6pl1

Description

The MaxInstances directive configures the maximum number of child processes that may be spawned by a parent proftpd process in standalone mode. The directive has no effect when used on a server running in inetd mode. Because each child proftpd process represents a single client connection, this directive also controls the maximum number of simultaneous connections allowed. Additional connections beyond the configured limit are syslog'd and silently disconnected. The MaxInstances directive can be used to prevent undesirable denial-of-service attacks (repeatedly connecting to the ftp port, causing proftpd to fork-bomb). By default, no limit is placed on the number of child processes that may run at one time.

See also

Examples

MaxLoginAttempts

Name

MaxLoginAttempts -- Sets how many password attempts are allowed before disconnection

Synopsis

MaxLoginAttempts [`MaxLoginAttempts number`]

`Default`
: MaxLoginAttempts 3

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_auth

`Compatibility`
: 0.99.0 and later

Description

The MaxLoginAttempts directive configures the maximum number of times a client may attempt to authenticate to the server during a given connection. After the number of attempts exceeds this value, the user is disconnected and an appropriate message is logged via the syslog mechanism.

See also

Examples

MaxRetrieveFileSize

Name

MaxRetrieveFileSize -- Restrict size of downloaded files

Synopsis

MaxRetrieveFileSize [`number|"*" units ["user"|"group"|"class" expression]`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>, <Anonymous>, <Directory>, .ftpaccess

`Module`
: mod_xfer

`Compatibility`
: 1.2.7rc1 and later

Description

When downloading files to clients (eg serving a RETR request), the server will check for any configured limit against the size of the file being requested, and abort any transfers if the requested file's size exceeds the configured limit.

A single "*" argument configures unlimited file sizes, and is used primarily to override any inherited restrictions from higher contexts. The given number is the number of bytes for the limit, and is followed by a units specifier of (case-insensitive) "Gb" (Gigabytes), "Mb" (Megabytes), "Kb" (Kilobytes), or "B" (bytes). The given number of bytes is multiplied by the appropriate factor.

The optional parameters are used to restrict the file size limits only to specific users. If the "user" restriction is given, then expression is a user-expression specifying to which users the rule applies. Similarly for the "group" restriction. For the "class" restriction, the expression is simply the name of connection class for whom the rule will apply. If no matching user, group, or class expression is found for the current user (in that order), then a limit with no expression (i.e. no "user", "group", or "class" identifier) is applied.

See Also: MaxStoreFileSize

See also

Examples

```
# Restrict downloads to only 1 gigabyte
MaxRetrieveFileSize 1 Gb
```

```
# Restrict downloads for user fred, but allow unlimited download size for
# everyone else
MaxStoreFileSize 50 Kb user fred
MaxStoreFileSize *
```

MaxStoreFileSize

Name

MaxStoreFileSize -- Restrict size of uploaded files

Synopsis

MaxStoreFileSize [`number|"*" units ["user"|"group"|"class" expression]`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>, <Anonymous>, <Directory>, .ftpaccess

`Module`
: mod_xfer

`Compatibility`
: 1.2.7rc1 and later

Description

When uploading files from a client (eg serving a STOR request), the server will check for any configured limit against the size of the file being sent, and abort any transfers if/when the given file's size exceeds the configured limit.

A single "*" argument configures unlimited file sizes, and is used primarily to override any inherited restrictions from higher contexts. The given number is the number of bytes for the limit, and is followed by a units specifier of (case-insensitive) "Gb" (Gigabytes), "Mb" (Megabytes), "Kb" (Kilobytes), or "B" (bytes). The given number of bytes is multiplied by the appropriate factor.

The optional parameters are used to restrict the file size limits only to specific users. If the "user" restriction is given, then expression is a user-expression specifying to which users the rule applies. Similarly for the "group" restriction. For the "class" restriction, the expression is simply the name of connection class for whom the rule will apply. If no matching user, group, or class expression is found for the current user (in that order), then a limit with no expression (ie no "user", "group", or "class" identifier) is applied.

See Also: MaxRetrieveFileSize

See also

Examples

```
# Restrict upload to only 3 megabytes
MaxStoreFileSize 3 Mb
```

```
# Restrict anonymous uploads to 50k, but allow unlimited upload size for
# everyone else
MaxStoreFileSize 50 Kb user anonymous
MaxStoreFileSize *
```

MultilineRFC2228

Name

MultilineRFC2228 -- Enable RFC2228 multiline response mode

Synopsis

MultilineRFC2228 [`MultilineRFC2228 on|off`]

`Default`
: MultilineRFC2228 off

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 1.2.0pre3 and later

Description

By default, proftpd sends multiline responses as per RFC 959, i.e.: 200-First line More lines... 200 Last line RFC 2228 specifies that "6xy" response codes will be sent as follows: 600-First line 600-More lines... 600 Last line Note that 2228 ONLY specifies this for response codes starting with '6'. Enabling this directive causes ALL responses to be sent in this format, which may be more compatible with certain web browsers and clients. Also note that this is NOT the same as wu-ftpd's multiline responses, which do not comply with any RFC. Using this method of multilines is more likely to be compatible with all clients, although it isn't strictly RFC, and is thus not enabled by default.

See also

Examples

Order

Name

Order -- Configures the precedence of the Limit directives

Synopsis

Order [`Order allow,deny|deny,allow`]

`Default`
: Order allow,deny

`Context`
: <Limit>

`Module`
: mod_core

`Compatibility`
: 0.99.0pl6 and later

Description

The Order directive configures the order in which Allow and Deny directives are checked inside of a <Limit> block. Because Allow directives are permissive, and Deny directives restrictive, the order in which they are examined can significantly alter the way security functions. If the default setting of allow,deny is used, "allowed" access permissions are checked first. If an Allow directive explicitly allows access to the <Limit> context, access is granted and any Deny directives are never checked. If Allow did not explicitly permit access, Deny directives are checked. If any Deny directive applies, access is explicitly denied. Otherwise, access is granted. When deny,allow is used, "deny" access restrictions are checked first. If any restriction applies, access is denied immediately. If nothing is denied, Allow permissions are checked. If an Allow explicitly permits access, access to the entire context is permitted; otherwise access is implicitly denied. For clarification, the following illustrates the steps used when checking Allow/Deny access: Order allow,deny Check Allow directives. If one or more apply, exit with result: ALLOW Check Deny directives. If one or more apply, exit with result: DENY Exit with default implicit ALLOW Order deny,allow Check Deny directives. If one or more apply, exit with result: DENY Check Allow directives. If one or more apply, exit with result: ALLOW Exit with default implicit: DENY

See also

Examples

PassivePorts

Name

PassivePorts -- Specify the ftp-data port range to be used

Synopsis

PassivePorts [`PassivePorts min-pasv-port max-pasv-port`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.0rc2 and later

Description

PassivePorts restricts the range of ports from which the server will select when sent the PASV command from a client. The server will randomly choose a number from within the specified range until an open port is found. Should no open ports be found within the given range, the server will default to a normal kernel-assigned port, and a message logged.

The port range selected must be in the non-privileged range (eg. greater than or equal to 1024); it is STRONGLY RECOMMENDED that the chosen range be large enough to handle many simultaneous passive connections (for example, 49152-65534, the IANA-registered ephemeral port range).

See also

Examples

```
# Use the IANA registered ephemeral port range
PassivePorts 49152 65534
```

PathAllowFilter

Name

PathAllowFilter -- Only allow new files which match a specified pattern

Synopsis

PathAllowFilter [`PathAllowFilter regular-expression`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>, <Directory>, .ftpaccess

`Module`
: mod_core

`Compatibility`
: 1.1.7 and later

Description

PathAllowFilter allows the configuration of a regular expression that must be matched for all newly uploaded (stored) files. The regular expression is applied against the entire pathname specified by the client, so care must be taken when creating a proper regex. Paths that fail the regex match result in a "Forbidden filename" error being returned to the client. If the regular-expression argument contains whitespace, it must be enclosed in quotes.

See also

PathDenyFilter

Examples

```
# Only allow a-z 0-9 . - _ in file names,
PathAllowFilter ^[a-z0-9._-]+$

# as above but with upper case characters as well
PathAllowFilter ^[A-Za-z0-9._-]+$
```

PathDenyFilter

Name

PathDenyFilter -- Disallow new files which match a specified pattern

Synopsis

PathDenyFilter [`PathDenyFilter regular-expression`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>, <Directory>, .ftpaccess

`Module`
: mod_core

`Compatibility`
: 1.1.7 and later

Description

Similar to PathAllowFilter, PathDenyFilter specifies a regular expression which must not match any uploaded pathnames. If the regex does match, a "Forbidden filename" error is returned to the client. This can be especially useful for forbidding .ftpaccess or .htaccess files.

See also

PathAllowFilter

Examples

```
# We don't want .ftpaccess or .htaccess files to be uploaded
PathDenyFilter "(\\.ftpaccess|\\.htaccess)$"
```

PersistentPasswd

Name

PersistentPasswd -- Sets handling of unix auth files

Synopsis

PersistentPasswd [`PersistentPasswd on|off`]

`Default`
: Platform dependent

`Context`
: server config

`Module`
: mod_auth_unix

`Compatibility`
: 1.1.5 and later

Description

The PersistentPasswd directive controls how proftpd handles authentication, user/group lookups, and user/group to name mapping. If set to "on", proftpd will attempt to open the system-wide /etc/passwd, /etc/group (and /etc/shadow, potentially) files itself, holding them open even during a chroot()ed login. Note that /etc/shadow is never held open, for security reasons). On some platforms, you must turn this option on, as the libc functions are incapable of accessing these databases from inside of a chroot(). At configure-time, the configuration script will attempt to detect whether or not you need this support, and make it the default. However, such "guessing" may fail, and you will have to manually enable or disable the feature. If you cannot see user or group names when performing a directory listing inside an anonymous chrooted login, this indicates you must enable the directive. Use of the AuthUserFile or AuthGroupFile directives will force partial support for persistent user or group database files, regardless of PersistentPasswd's setting.

Note: NIS/NIS+ and NSS users will most likely want to disable this feature, regardless of proftpd's detected configuration defaults. Failure to disable this will make your NIS/NIS+ maps and NSS lookups not work! On certain systems, you may also need to compile ProFTPD with the --enable-autoshadow option in order to authenticate both users from NIS maps or NSS lookups, and local users.

See also

Examples

PidFile

Name

PidFile -- Set the filepath to hold the pid of the master server

Synopsis

PidFile [`PidFile filename`]

`Default`
: none

`Context`
: server config, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.0rc2 and later

Description

The PidFile directive sets the file to which the server records the process id of the daemon. The filename should be relative to the system root, ie /var/run/proftpd/pidfile. The PidFile is only used in standalone mode. It is often useful to be able to send the server a signal, so that it closes and then reopens its ErrorLog and TransferLog, and re-reads its configuration files. This is done by sending a SIGHUP (kill -1) signal to the process id of the master daemon listed in the PidFile.

See also

Examples

Port

Name

Port -- Set the port for the control socket

Synopsis

Port [`Port port-number`]

`Default`
: Port 21

`Context`
: server config, <VirtualHost>

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The Port directive configures the TCP port which proftpd will listen on while running in standalone mode. It has no effect when used upon a server running in inetd mode (see ServerType). The directive can be used in conjunction with <VirtualHost> in order to run a virtual server on the same IP address as the master server, but listening on a different port.

For any server, either <VirtualHost> or server config, setting Port 0 effectively turns off that server.

See also

Examples

RadiusAcctServer

Name

RadiusAcctServer -- Setup RADIUS accounting details

Synopsis

RadiusAcctServer [`server[:port] shared-secret [timeout]`]

`Default`
: none

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_radius

`Compatibility`
: 1.2.7rc1 and later

Description

The RadiusAcctServer is used to specify a RADIUS server to be used for accounting. The server parameter may be either an IP address or a DNS hostname. If not specified, the port used will be the IANA-registered 1813. The optional timeout parameter is used to tell mod_radius how long to wait for a response from the server; it defaults to 30 seconds.

Multiple RadiusAcctServers may be configured; each will be tried, in order of appearance in the configuration file, until that server times out or mod_radius receives a response.

If no RadiusAcctServers are configured, mod_radius will not use RADIUS for accounting.

See also

RadiusAuthServer

RadiusAuthServer

Name

RadiusAuthServer -- Setup RADIUS authenticator details

Synopsis

RadiusAuthServer [`server[:port] shared-secret [timeout]`]

`Default`
: none

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_radius

`Compatibility`
: 1.2.7rc1 and later

Description

The RadiusAcctServer is used to specify a RADIUS server to be used for accounting. The server parameter may be either an IP address or a DNS hostname. If not specified, the port used will be the IANA-registered 1813. The optional timeout parameter is used to tell mod_radius how long to wait for a response from the server; it defaults to 30 seconds.

Multiple RadiusAcctServers may be configured; each will be tried, in order of appearance in the configuration file, until that server times out or mod_radius receives a response.

If no RadiusAcctServers are configured, mod_radius will not use RADIUS for accounting.

See also

RadiusAuthServer

RadiusEngine

Name

RadiusEngine -- Enable RADIUS support

Synopsis

RadiusEngine [`on|off`]

`Default`
: off

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_radius

`Compatibility`
: 1.2.7rc1 and later

Description

The RadiusEngine directive enables or disables the module's runtime RADIUS engine. If it is set to off this module does no RADIUS authentication or accounting at all. Use this directive to disable the module instead of commenting out all mod_radius directives.

See also

RadiusLog

Name

RadiusLog -- Specify the logfile for reporting / debugging

Synopsis

RadiusLog [`"file"|none`]

`Default`
: none

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_radius

`Compatibility`
: 1.2.7rc1 and later

Description

The RadiusLog directive is used to a specify a log file for mod_radius reporting and debugging, and can be done a per-server basis. The file parameter must be the full path to the file to use for logging. Note that this path must not be to a world-writeable directory and, unless AllowLogSymlinks is explicitly set to on (generally a bad idea), the path must not be a symbolic link.

If file is "none", no logging will be done at all; this setting can be used to override a RadiusLog setting inherited from a <Global> context.

See also

RadiusRealm

Name

RadiusRealm -- Setup the authentication realm

Synopsis

RadiusRealm [`realm`]

`Default`
: none

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_radius

`Compatibility`
: 1.2.7rc1 and later

Description

The RadiusRealm directive configures a realm string that will be added to the username in the constructed RADIUS packets.

See also

Examples

RadiusRealm .castaglia.org

RadiusUserInfo

Name

RadiusUserInfo -- Configure login information via RADIUS

Synopsis

RadiusUserInfo [`uid gid home shell [suppl-group-names suppl-group-ids]`]

`Default`
: none

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_radius

`Compatibility`
: 1.2.7rc1 and later

Description

The RadiusUserInfo directive is used to configure login information used for every user authenticated via RADIUS. The optional suppl-group-names and suppl-group-ids parameters are used to specify supplemental group membership for each user; the number of names and IDs must match if these parameters are used.

In order to support RADIUS servers that may use custom attributes in their Access-Accept response packets to supply user information back to the RADIUS client (mod_radius in this case), this directive allows the following syntax for some of its parameters:

```
$(attribute-id:default-value)
```

where the enclosing $() signals that the parameter is to be supplied by the RADIUS server, attribute-id is the custom attribute ID for which to search in the response packet, and default-value is the value to use in case the requested attribute is not present in the response packet. This syntax is not supported for the suppl-group-names or suppl-group-ids parameters.

If RadiusUserInfo is not used, mod_radius will perform pure "yes/no" authentication only, in the style of PAM. The information that would have been configured via this directive will be pulled from other sources (e.g. /etc/passwd, AuthUserFiles, MySQL tables, etc).

See also

RatioFile

Name

RatioFile -- Ratio directive

Synopsis

RatioFile [`RatioFile foo1 foo2 foo3`]

`Default`
: None known

`Context`
: <Directory>, <Anonymous>, <Limit>,.ftpaccess

`Module`
: mod_ratio

`Compatibility`
: at least 1.2.0 and later

Description

The RatioFile directive .... Example: RatioFile

See also

Examples

Ratios

Name

Ratios -- (docs incomplete)

Synopsis

Ratios [`Ratios foo1 foo2 foo3`]

`Default`
: None known

`Context`
: <Directory>, <Anonymous>, <Limit>,.ftpaccess

`Module`
: mod_ratio

`Compatibility`
: at least 1.2.0 and later

Description

The Ratios directive .... Example: Ratios

See also

Examples

RatioTempFile

Name

RatioTempFile -- Ratio directive

Synopsis

RatioTempFile [`RatioTempFile foo1 foo2 foo3`]

`Default`
: None known

`Context`
: <Directory>, <Anonymous>, <Limit>,.ftpaccess

`Module`
: mod_ratio

`Compatibility`
: at least 1.2.0 and later

Description

The RatioTempFile directive .... Example: RatioTempFile

See also

Examples

RequireValidShell

Name

RequireValidShell -- Allow connections based on /etc/shells

Synopsis

RequireValidShell [`RequireValidShell on|off`]

`Default`
: RequireValidShell on

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_auth

`Compatibility`
: 0.99.0 and later

Description

The RequireValidShell directive configures the server, virtual host or anonymous login to allow or deny logins which do not have a shell binary listed in /etc/shells. By default, proftpd disallows logins if the user's default shell is not listed in /etc/shells. If /etc/shells cannot be found, all default shells are assumed to be valid.

See also

Examples

RewriteCondition

Name

RewriteCondition -- Define a rule condition

Synopsis

RewriteCondition [`condition pattern`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>, <Anonymous>, <Directory>

`Module`
: mod_rewrite

`Compatibility`
: 1.2.6rc1 and later

Description

The RewriteCondition directive defines a rule condition. Precede a RewriteRule directive with one or more RewriteCondition directives. The following rewriting rule is only used if its pattern matches the current state of the FTP command and if these additional conditions apply too.

Condition is a string which can contain the following expanded constructs in addition to plain text:

- **RewriteRule backreferences**

  These are backreferences of the form:

  $N

  (0 <= N <= 9) which provide access to the grouped parts (parentheses!) of the pattern from the corresponding RewriteRule directive (the one following the current bunch of RewriteCondition directives). Note that $0 will refer back to the entire original string being matched.
- **RewriteCondition backreferences**

  These are backreferences of the form:

  ***%*N**

  (0 <= N <= 9) which provide access to the grouped parts (parentheses!) of the pattern from the previous RewriteCondition attached to this RewriteRule.
- **RewriteMap expansions:**

  These are expansions of the form:

${map-name:lookup-key|default-value}

See the documentation for RewriteMap for more details.

- **Variable substitutions:**

  These are substitutions of the form:

  - **%a** client IP address
  - **%c** name of Class for current session
  - **%f** filename
  - **%F** transfer path, as seen by the client (only useful for upload/download commands)
  - **%g** primary group of authenticated user
  - **%G** supplemental groups of authenticated user
  - **%h** client DNS name
  - **%m** FTP command
  - **%p** port of server handling the session
  - **%u** name of authenticated user
  - **%U** name of user sent by client via USER
  - **%v** ServerName of server handling the session

Pattern is the condition pattern, i.e., a regular expression which is applied to the current instance of the condition, i.e., condition is evaluated and then matched against pattern. You can prefix the pattern string with a '!' character (exclamation mark) to specify a non-matching pattern.

See also

RewriteRule RewriteMap

Examples

RewriteEngine

Name

RewriteEngine -- Enable/disable mod_rewrite

Synopsis

RewriteEngine [`on|off`]

`Default`
: off

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_rewrite

`Compatibility`
: 1.2.6rc1 and later

Description

The RewriteEngine directive enables or disables the module's runtime rewriting engine. If it is set to off this module does no parsing or rewriting at all. Use this directive to disable the module instead of commenting out all mod_rewrite directives.

See also

RewriteLock

Name

RewriteLock -- Set the filename for synchronization lockfile

Synopsis

RewriteLock [`filename`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_rewrite

`Compatibility`
: 1.2.6rc1 and later

Description

The RewriteLock directive sets the filename for a synchronization lockfile which mod_rewrite needs to communicate with RewriteMaps of type fifo. Set file to a local absolute path (not on a NFS-mounted device) when you want to use a rewriting FIFO. It is not required for other types of rewriting maps.

See also

RewriteLog

Name

RewriteLog -- Specify a log file for mod_rewrite reporting

Synopsis

RewriteLog [`file|"none"`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_rewrite

`Compatibility`
: 1.2.6rc1 and later

Description

The RewriteLog directive is used to a specify a log file for mod_rewrite reporting and debugging, and can be done a per-server basis. The file parameter must be the full path to the file to use for logging. Note that this path must **not** be to a world-writeable directory and, unless AllowLogSymlinks is explicitly set to on (generally a bad idea), the path must **not** be a symbolic link. In general, this directive should only be used for debugging your mod_rewrite configuration, and should be removed once debugging is completed; **do not use this directive in a production configuration.**

If file is "none", no logging will be done at all; this setting can be used to override a RewriteLog setting inherited from a <Global> context.

See also

RewriteMap

Name

RewriteMap -- Define a rewrite map

Synopsis

RewriteMap [`map-name map-type:map-soure`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_rewrite

`Compatibility`
: 1.2.6rc1 and later

Description

The RewriteMap directive defines a rewriting map which can be used inside rule substitution strings by the mapping-functions to insert/substitute fields through a key lookup. The source of this lookup can be of various types.

The map-name is the name of the map and will be used to specify a mapping-function for the substitution strings of a rewriting rule via one of the following constructs:

${ map-name : lookup-key }

${ map-name : lookup-key | default-value

When such a construct occurs the map map-name is consulted and the key lookup-key is resolved. If the key is found, the map-function construct is substituted by subst-value. If the key is not found then it is substituted by default-value or by the empty string if no default-value was specified.

The following combinations for map-type and map-src can be used:

- **Standard Plain Text**

  map-type: txt, map-src: Unix filesystem path to valid regular file.

  This is the standard rewriting map feature where the map-src is a plain ASCII file containing either blank lines, comment lines (starting with a '#' character) or pairs like the following - one per line.

  matching-key subst-value

Example 1-1. Example Usermap

```
# ------------------------------------------
# usermap.txt -- map for rewriting user names
# ------------------------------------------

Dave.Admin      dave       # The Uber-admin
root            anonymous  # no one should be logging in as root anyway
```

And, to configure this map to be used:

```
RewriteMap real-to-user txt:/path/to/file/usermap.txt
```

- **FIFO/Named Pipe**

map-type: fifo, map-src: Unix filesystem path to valid FIFO.

For this rewriting map, map-src is a FIFO (a.k.a. named pipe). To create it, you can use the mkfifo(1) command. An external program that opens the FIFO for reading and writing **must** be started before proftpd is started. This program can communicate with the rewriting engine via the FIFO. For each mapping lookup, it can read the key to lookup as a newline-terminated string from the FIFO. It then has to write back to the FIFO the looked-up value as a newline-terminated string, or just simply newline character (denoting an empty string) if there is no corresponding value for the given key).

An example program which will implement a 1:1 mapping (i.e., key == value) could be:

Example 1-2. Example FIFO/Named Pipe 1:1 mapping

```
#!/usr/bin/perl
    use strict;

    use File::Basename qw(basename);
    use Getopt::Long;
    use IO::Handle;
    use IO::Select;

    my $default_delay = 0.5;
    my $program = basename($0);
    my %opts = ();

    GetOptions(\%opts, 'delay=f', 'fifo=s', 'help', 'verbose');

    usage() if $opts{'help'};

    my $delay = $opts{'delay'} ? $opts{'delay'} : $default_delay;

    die "$program: missing required --fifo parameter\n" unless $opts{'fifo'};
    my $fifo = $opts{'fifo'};

    my $verbose = $opts{'verbose'} ? 1 : 0;

    open(my $fifo_fh, "+> $fifo") or die "$program: unable to open $fifo: $!\n";

    # Instantiate a Select object for knowing when to read from and write to
    # the FIFO.
    my $sel = IO::Select->new();
```

```
while (1) {

  # Blocking select() for reading.
  $sel->add($fifo_fh);

  print STDERR "$program: selecting for reading\n" if $verbose;
  my ($rfh) = $sel->can_read();

  my $key = <$rfh>;
  print STDERR "$program: read '$key'\n" if $verbose;

  # Lookup a value for the given key.
  my $value = lookup_value($key);

  # Clear the Select object's filehandles.
  $sel->remove();

  print $fifo_fh "$value\n" if $verbose;
  $fifo_fh->flush();

  print STDERR "$program: wrote '$value'\n" if $verbose;

  # Wait for the buffer's byte to be cleared before reading again.
  wait_fifo($fifo_fh);
}

close($fifo_fh);
print STDOUT "$program: done\n" if $verbose;

exit 0;

# ------------------------------------------------------------------------
sub lookup_value {
  my ($key) = @_;

  # NOTE: do something to obtain a value for the given key here.
  chomp(my $value = $key);

  return $value;
}

# ------------------------------------------------------------------------
sub usage {
  print STDOUT <<END_OF_USAGE;

usage: $program [options]

  --delay         Configure the buffer check delay.
                  The default is $default_delay seconds.

  --fifo          Configure the path to the FIFO.  Required.

  --help          Displays this message.

  --verbose       Enables verbose output while $program runs.

END_OF_USAGE

  exit 0;
}
```

```
# ------------------------------------------------------------------------
sub wait_fifo {
  my ($fh) = @_;

  # Now we get tricky.  Use ioctl(2) to poll the number of bytes to
  # be read from the FIFO filehandle.  When the number drops to zero,
  # it means that the data we just wrote has been read from the buffer
  # by some other process, so we can go back to the top of this loop.
  # Otherwise, if this program loops faster than the reader/writer on
  # the other end of the FIFO, we'd end up reading the data we just
  # wrote.  Quite annoying, actually.
  #
  # Note: this value must be manually extracted from the system header files
  # using the following program:
  #
  # -------- fionread.c -------------------
  #  #include <sys/ioctl.h>
  #
  #  int main(int argc, char *argv[]) {
  #   printf("%#08x\n", FIONREAD);
  #   return 0;
  # }
  # ---------------------------------------
  #
  # > cc -o fionread fionread.c
  # > ./fionread

  my $FIONREAD = 0x00541b;

  my $size = pack('L', 0);
  ioctl($fh, $FIONREAD, $size) or die "$program: unable to use ioctl: $!\n";
  $size = unpack('L', $size);

  while ($size != 0) {
    print STDERR "$program: waiting for buffer to be read\n" if $verbose;
    select(undef, undef, undef, $delay);

    $size = pack('L', 0);
    ioctl($fh, $FIONREAD, $size) or die "$program: unable to use ioctl: $!\n";
    $size = unpack('L', $size);
  }
}
```

To make use of this example script, simply implement your lookup code in the lookup_value() subroutine. Be very careful with such scripts, though:

1. "Keep it simple, stupid" (KISS), because if this program hangs it will hang proftpd when the rule occurs. Well, keep it as simple as possible...
2. Avoid one common mistake: avoid buffered I/O if possible. This can cause a deadloop. If necessary, be sure to flush the filehandle before reading, and after writing.
3. Use the RewriteLock directive to define a lockfile mod_rewrite can use to synchronize the communication to the FIFO program. By default no such synchronization takes place.

- **Internal Function**

map-type: int, map-src: Internal mod_rewrite function.

Here the map-src is a mod_rewrite built-in function. Currently you cannot create your own, but the following functions already exist:

- **toupper**

  Converts the looked up key to all upper case.
- **tolower**

  Converts the looked up key to all lower case.
- **unescape**

  Translates hex-encodings in the looked up key back to special characters.
- **utf8trans**

  Translates UTF-8 encodings in the lookup up key into Latin-1 characters.

The RewriteMap directive can occur more than once. For each mapping-function use one RewriteMap directive to declare its rewriting map name.

Note: For plain text files the looked-up keys are cached in-core until the mtime of the text map file changes or the server does a restart. This way you can have map-functions in rules which are used for **every** request. This is no problem, because the parsing of the text files only happens once!

See also

RewriteCondition

RewriteRule

Name

RewriteRule -- Define a rewrite rule

Synopsis

RewriteRule [`pattern substitution`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>, <Anonymous>, <Directory>

`Module`
: mod_rewrite

`Compatibility`
: 1.2.6rc1 and later

Description

The RewriteRule directive is the real rewriting workhorse. The configuration directive can occur more than once. Each directive defines a single rewriting rule. The order of definition of these rules is important, because this order is used when applying the rules at run-time.

Pattern can be POSIX regular expression which gets applied to the current FTP command argument(s).

Some hints about the syntax of regular expressions:

- **Text:**

```
.           Any single character
[chars]     Character class: one of chars
[^chars]    Character class: none of chars
text1|text2 Alternative: text1 or text2
```

- **Quantifiers:**

```
?           0 or 1 of the preceding text
*           0 or N of the preceding text (N > 0)
+           1 or N of the preceding text (N > 1)
```

- **Grouping:**

```
(text)       Grouping of text
             (either to set the borders of an alternative or
             for making backreferences where the Nth group can
             be used on the RHS of a RewriteRule with $N)
```

- **Anchors:**

```
^           Start of line anchor
$           End of line anchor
```

- **Escaping:**

```
\char       Escape that particular char
            (for instance to specify the chars ".[]()" etc.)
```

For more information about regular expressions have a look at your local regex(3) manpage. If you are interested in more detailed information about regular expressions and their variants (POSIX regex, Perl regex, etc.) have a look at the following dedicated book on this topic:

Mastering Regular Expressions Jeffrey E.F. Friedl Nutshell Handbook Series O'Reilly & Associates, Inc. 1997 ISBN 1-56592-257-3

Additionally in mod_rewrite the NOT character ('!') is a possible pattern prefix. This gives you the ability to negate a pattern; to say, for instance: "if the current argument(s) does NOT match this pattern". This can be used for exceptional cases, where it is easier to match the negative pattern, or as a last default rule.

Notice: When using the NOT character to negate a pattern you cannot have grouped wildcard parts in the pattern. This is impossible because when the pattern does NOT match, there are no contents for the groups. In consequence, if negated patterns are used, you cannot use $N in the substitution string.

Substitution of a rewriting rule is the string which is substituted for (or replaces) the original argument(s) for which pattern matched. Beside plain text you can use:

1. $N backreferences to the RewriteRule pattern
2. %N backreferences to the last matched RewriteCondition pattern
3. variables as in RewriteCondition test strings
4. map function calls (${map-name:lookup-key|default-value})

Backreferences are $**N** (**N**=0..9) identifiers which will be replaced by the contents of the **N**th group of the matched pattern. The variables are the same as for the condition of a RewriteCondition directive, with two additions:

- %**P** process ID
- %**t** Unix time since the epoch, in seconds

The map functions come from the RewriteMap directive and are explained there. These four types of variables are expanded in the order of the above list.

All of the rewriting rules are applied to substitution. The command argument(s) is completely replaced by the substitution.

See also

RewriteCondition RewriteMap

Examples

RLimitCPU

Name

RLimitCPU -- Configure the maximum CPU time in seconds used by a process

Synopsis

RLimitCPU [`RLimitCPU ["daemon"|"session"|"none"] soft-limit|"max" [hard-limit|"max"]`]

`Default`
: System defaults

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 1.2.1rc1 and later

Description

RLimitCPU takes from one to three parameters. The first parameter may be one of "daemon" (applies the limit only to the daemon process), "session" (applies the limit only to child processes handling each FTP session), or "none" (disables any possibly inherited limits). Note that if "daemon" is used, the directive may then only occur in the "server config" context. If none of these keywords are used, the limit is assumed to apply to both daemon and session processes. After any potential keyword, the resource limit must be set. The next parameter is also optional, and sets the maximum resource limit. Either limit parameter can be a number, or "max" to indicate to the server that the limit should be set to the maximum allowed by the operating system.

CPU resource limits are expressed in seconds per process.

See Also:

RLimitMemory, RLimitOpenFiles

Examples

RLimitMemory

Name

RLimitMemory -- Configure the maximum memory in bytes used by a process

Synopsis

RLimitMemory [`RLimitMemory ["daemon"|"session"|"none"] soft-limit[units]|"max" [hard-limit[units]|"max"]`]

`Default`
: None

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 1.2.1rc1 and later

Description

RLimitMemory takes from one to three parameters. The first parameter may be one of "daemon" (applies the limit only to the daemon process), "session" (applies the limit only to child processes handling each FTP session), or "none" (disables any possibly inherited limits). Note that if "daemon" is used, the directive may then only occur in the "server config" context. If none of these keywords are used, the limit is assumed to apply to both daemon and session processes. After any potential keyword, the resource limit must be set. The next parameter is also optional, and sets the maximum resource limit. Either limit parameter can be a number, or "max" to indicate to the server that the limit should be set to the maximum allowed by the operating system.

Memory resource limits are expressed in bytes per process. An optional case-insensitive units specifier may follow the number of bytes given: G (Gigabytes), M (Megabytes), K (Kilobytes), or B (bytes). If the units specifier is used, the given number of bytes is multiplied by the appropriate factor.

See Also

RLimitCPU, RLimitOpenFiles

RLimitOpenFiles

Name

RLimitOpenFiles -- Configure the maximum number of open files used by a process

Synopsis

RLimitOpenFiles [`RLimitOpenFiles ["daemon"|"session"|"none"] soft-limit|"max" [hard-limit|"max"]`]

`Default`
: None

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 1.2.1rc1 and later

Description

RLimitOpenFiles takes from one to three parameters. The first parameter may be one of "daemon" (applies the limit only to the daemon process), "session" (applies the limit only to child processes handling each FTP session), or "none" (disables any possibly inherited limits). Note that if "daemon" is used, the directive may then only occur in the "server config" context. If none of these keywords are used, the limit is assumed to apply to both daemon and session processes. After any potential keyword, the resource limit must be set. The next parameter is also optional, and sets the maximum resource limit. Either limit parameter can be a number, or "max" to indicate to the server that the limit should be set to the maximum allowed by the operating system.

File resource limits are expressed in number of files per process.

See Also:

RLimitCPU, RLimitMemory

RootLogin

Name

RootLogin -- Permit root user logins

Synopsis

RootLogin [`RootLogin on|off`]

`Default`
: RootLogin off

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_auth

`Compatibility`
: 1.1.5 and later

Description

Normally, proftpd disallows root logins under any circumstance. If a client attempts to login as root, using the correct password, a special security message is sent to syslog. When the RootLogin directive is turned On, the root user may authenticate just as any other user could (assuming no other access control measures deny access); however the root login security message is still sysloged. Obviously, extreme care should be taken when using this directive.

The use of RootLogin in the Anonymous context is only valid when the User / Group defined in the Anonymous block is set to 'root'

See also

Examples

RootRevoke

Name

RootRevoke -- Drop root privileges completely

Synopsis

RootRevoke [`RootRevoke on|off`]

`Default`
: RootRevoke off

`Context`
: server config, <VirtualHost>, <Global>, <Anonymous>

`Module`
: mod_auth

`Compatibility`
: 1.2.9rc1 and later

Description

The RootRevoke directive causes all root privileges to be dropped once a user is authenticated. This will also cause active transfers to be disabled, if the server is listening on a port less than 1025. Note that this only affects active transfers; passive transfers will not be blocked.

See also

Examples

SaveRatios

Name

SaveRatios -- FIXME FIXME

Synopsis

SaveRatios [`SaveRatios foo1 foo2 foo3`]

`Default`
: None known

`Context`
: <Directory>, <Anonymous>, <Limit>,.ftpaccess

`Module`
: mod_ratio

`Compatibility`
: at least 1.2.0 and later

Description

The SaveRatios directive .... Example: SaveRatios

See also

Examples

ScoreboardFile

Name

ScoreboardFile -- Sets the name and path of the scoreboard file

Synopsis

ScoreboardFile [`path`]

`Default`
: ScoreboardFile /usr/local/var/proftpd.scoreboard

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 1.2.7rc1 and later

Description

The ScoreboardFile directive sets the path to the file where the daemon will store its run-time "scoreboard" session information. This file is necessary for MaxClients to work properly, as well as other utilities (such as ftpwho and ftpcount). Note that the directory containing the scoreboard cannot be world-writable.

This directive deprecates ScoreboardPath.

See also

Examples

ScoreboardFile /var/run/proftpd.scoreboard

ServerAdmin

Name

ServerAdmin -- Set the address for the server admin

Synopsis

ServerAdmin [`ServerAdmin "admin-email-address"`]

`Default`
: ServerAdmin root@[ServerName]

`Context`
: server config, <VirtualHost>

`Module`
: mod_core

`Compatibility`
: 0.99.0pl10 and later

Description

The ServerAdmin directive sets the email address of the administrator for the server or virtualhost. This address is displayed in magic cookie replacements (see DisplayLogin and DisplayFirstChdir).

See also

Examples

ServerIdent

Name

ServerIdent -- Set the message displayed on connect

Synopsis

ServerIdent [`ServerIdent off|on [identification string]`]

`Default`
: ServerIdent on "ProFTPD [version] Server (server name) [hostname]"

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.0pre2 and later

Description

The ServerIdent directive sets the default message displayed when a new client connects. Setting this to off displays "[hostname] FTP server ready." If set to on, the directive can take an optional string argument, which will be displayed instead of the default text. Sites desiring to give out minimal information will probably want a setting like ServerIdent on "FTP Server ready.", which won't even reveal the hostname.

See also

Examples

ServerIdent on "Welcome to ftp.linux.co.uk"

ServerLog

Name

ServerLog -- Configure logs on a per-server basis

Synopsis

ServerLog [`path`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_log

`Compatibility`
: 1.2.8rc1 and later

Description

The ServerLog directive disables the daemon's use of the syslog mechanism and instead redirects all logging output for the server to the specified filename. The filename argument must contain an absolute path. Use of this directive overrides any facility set by the SyslogFacility directive, as well as overriding any configured SystemLog.

ServerName

Name

ServerName -- Configure the name displayed to connecting users

Synopsis

ServerName [`ServerName "name"`]

`Default`
: ServerName "ProFTPD Server [version]"

`Context`
: server config, <VirtualHost>

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The ServerName directive configures the string that will be displayed to a user connecting to the server (or virtual server if the directive is located in a <VirtualHost> block). See Also: <VirtualHost>

See also

Examples

ServerType

Name

ServerType -- Set the mode proftpd runs in

Synopsis

ServerType [`ServerType type-identifier`]

`Default`
: ServerType standalone

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The ServerType directive configures the server daemon's operating mode. The type-identifier can be one of two values: inetd The daemon will expect to be run from the inetd "super server." New connections are passed from inetd to proftpd and serviced immediately. standalone The daemon starts and begins listening to the configured port for incoming connections. New connections result in spawned child processes dedicated to servicing all requests from the newly connected client.

See also

Examples

SetEnv

Name

SetEnv -- (docs incomplete)

Synopsis

SetEnv [`key value`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.10rc1 and later

Description

(docs incomplete)

See also

Examples

(docs incomplete)

ShowSymlinks

Name

ShowSymlinks -- Toggle the display of symlinks

Synopsis

ShowSymlinks [`ShowSymlinks on|off`]

`Default`
: (versions 1.1.5 and beyond) ShowSymlinks On

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_ls

`Compatibility`

Description

Compatibility: 0.99.0pl6 and later Symbolic links (if supported on the host OS and filesystem) can be either shown in directory listings (including the target of the link) or can be "hidden" (proftpd dereferences symlinks and reports the target's permissions and ownership). The default behavior is to show all symbolic links when normal users are logged in, and hide them for anonymous sessions. If a symbolic link cannot be dereferenced for any reason (permissions, target does not exist, etc) and ShowSymlinks is off, proftpd displays the link as a directory entry of type 'l' (link) with the ownership and permissions of the actual link. Under ProFTPD versions 1.1.5 and higher, the default behavior in regard to ShowSymlinks has been changed so that symbolic links are always displayed as such (in all cases), unless ShowSymlinks off is explicitly set.

See also

Examples

SocketBindTight

Name

SocketBindTight -- Controls how TCP/IP sockets are created

Synopsis

SocketBindTight [`SocketBindTight on|off`]

`Default`
: SocketBindTight off

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 0.99.0pl6 and later

Description

The SocketBindTight directive controls how proftpd creates and binds its initial tcp listen sockets in standalone mode (see ServerType). The directive has no effect upon servers running in inetd mode, because listen sockets are not needed or created. When SocketBindTight is set to off (the default), a single listening socket is created for each port that the server must listen on, regardless of the number of IP addresses being used by <VirtualHost> configurations. This has the benefit of typically requiring a relatively small number of file descriptors for the master daemon process, even if a large number of virtual servers are configured. If SocketBindTight is set to on, a listen socket is created and bound to a specific IP address for the master server and all configured virtual servers. This allows for situations where an administrator may wish to have a particular port be used by both proftpd (on one IP address) and another daemon (on a different IP address). The drawback is that considerably more file descriptors will be required if a large number of virtual servers must be supported. Example: Two servers have been configured (one master and one virtual), with the IP addresses 10.0.0.1 and 10.0.0.2, respectively. The 10.0.0.1 server runs on port 21, while 10.0.0.2 runs on port 2001. SocketBindTight off #default # proftpd creates two sockets, both bound to ALL available addresses. # one socket listens on port 21, the other on 2001. Because each socket is # bound to all available addresses, no other daemon or user process will be # allowed to bind to ports 21 or 2001. ... SocketBindTight on # proftpd creates two sockets again, however one is bound to 10.0.0.1, port 21 # and the other to 10.0.0.2, port 2001. Because these sockets are "tightly" # bound to IP addresses, port 21 can be reused on any address OTHER than # 10.0.0.1, and visa-versa with 10.0.0.2, port 2001. One side-effect of setting SocketBindTight to on is that connections to non-bound addresses will result in a "connection refused" message rather than the typical "500 Sorry, no server available to handle request on xxx.xxx.xxx.xxx.", due to the fact that no listen socket has been bound to the particular address/port pair. This may or may not be aesthetically desirable, depending on your circumstances.

See also

Examples

SocketOptions

Name

SocketOptions -- Tune socket-level options

Synopsis

SocketOptions [`[maxseg <size>] [rcvbuf <size>] [sndbuf <size>]`]

`Default`
: None

`Context`
: "server config", <VirtualHost>

`Module`
: mod_core

`Compatibility`
: 1.2.9rc1 and later

Description

The rcvbuf and sndbuf parameters are used for setting the TCP send/receive window sizes. The maxseg parameter is used for setting a MSS (Maximum Segment Size) via setsockopt(2)'s TCP_MAXSEG option. If the MSS is larger than the interface's MTU, it is ignored and has no effect.

If the send/receive window size is increased, it is helpful for performance to increase the internal buffer size. See the --enable-buffer-size argument to ./configure.

SQLAuthenticate

Name

SQLAuthenticate -- Specify authentication methods and what to authenticate

Synopsis

SQLAuthenticate {on | off}

or

SQLAuthenticate [users] [groups] [userset [fast]] [groupset [fast]]

`Default`
: SQLAuthenticate `on`

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.5rc1 and later

Description

The SQLAuthenticate directive configures mod_sql's authentication behavior, controlling whether to provide user and/or group information during authentication, and how that provisioning is performed. The parameters may appear in any order.

The available parameter values are:

- **on**

  Shorthand for SQLAuthenticate users groups userset groupset.
- **off**

  Disables all mod_sql authentication functions.
- **users**

  If present, mod_sql will do user lookups. If not present, mod_sql will do no user lookups at all, including the {set|get|end}pwent() calls (see below).
- **groups**

  If present, mod_sql will do group lookups. If not present, mod_sql will do no group lookups at all, including the {set|get|end}grent() calls (see below).
- **userset[fast]**

If present, mod_sql will process the potentially expensive {set|getlend}pwent() calls. If not present, mod_sql will not process these calls. Adding the suffix "fast" tells mod_sql to process the users as a single large query, rather than making a query per user. This may significantly reduce the number of queries against the database at the expense of increased memory use. This parameter will have no effect if "users" is not specified.

- **groupset[fast]**

  If present, mod_sql will process the potentially expensive {set|getlend}grent() calls. If not present, mod_sql will not process these calls. Adding the suffix "fast" tells mod_sql to process the groups as a single large query, rather than making a query per group. This may significantly reduce the number of queries against the database at the expense of increased memory use. This parameter will have no effect if "groups" is not specified.

The SQLLog and SQLShowInfo directives will always be processed by mod_sql. The SQLAuthenticate directive only affects the user and group lookup/authentication portions of the module.

Turning off (i.e. by not including) the userset or groupset parameters affects the functionality of mod_sql. Not allowing these lookups may remove the ability to control access or control functionality by group membership, depending on your other authentication handlers and the data available to them. At the same time, choosing not to do these lookups may dramatically speed login for many large sites.

The "fast" suffix is not appropriate for every site. Normally, mod_sql will retrieve a list of users and groups, and get information from the database on a per-user or per-group basis. This is query intensive: it requires (nn + 1) queries, where n is the number of users or groups to lookup. By choosing "fast" lookups, mod_sql will make a single SELECT query to get information from the database.

In exchange for the radical reduction in the number of queries, the single query will increase the memory consumption of the process; all group or user information will be read at once rather than in discrete chunks.

Group Table Structure

Normally **mod_sql** allows multiple group members per row, and multiple rows per group. If you use the "fast" option for groupset, you **must** use only one row per group. For example, normally mod_sql treats the following three tables in exactly the same way:

```
|-------------------------------------------------|
| GROUPNAME | GID | MEMBERS                       |
|-------------------------------------------------|
| group1    | 1000 | naomi                        |
| group1    | 1000 | priscilla                    |
| group1    | 1000 | gertrude                     |
|-------------------------------------------------|

|-------------------------------------------------|
| GROUPNAME | GID | MEMBERS                       |
|-------------------------------------------------|
| group1    | 1000 | naomi, priscilla             |
| group1    | 1000 | gertrude                     |
|-------------------------------------------------|
```

```
|-------------------------------------------------|
|  GROUPNAME  |  GID  |  MEMBERS                  |
|-------------------------------------------------|
| group1      | 1000  | naomi, priscilla, gertrude |
|-------------------------------------------------|
```

If you use the "fast" option, mod_sql assumes that all entries are structured like the last example.

See also

SQLUserInfo SQLGroupInfo

Examples

SQLAuthTypes

Name

SQLAuthTypes -- Specify the allowed authentication types and their check order

Synopsis

SQLAuthTypes [`[OpenSSL]`][`[Crypt]`][`[Backend]`][`[Plaintext]`][`[Empty]`]

`Default`
: none

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.0 and later

Description

This directive deprecates 'SQLEmptyPasswords', 'SQLScrambledPasswords', 'SQLSSLHashedPasswords', 'SQLPlaintextPasswords', and 'SQLEncryptedPasswords'.

The SQLAuthTypes directive specifies which authentication method are to be allowed, and their order of use. **You must specify at least one authentication method.**

The current supported authentication methods are:

- **Backend**

  Allows database-specific backend passwords. Not all backend databases support this option. For example, MySQL datatabases use this option to authenticate MySQL 'PASSWORD()' encrypted passwords. The Postgres backend, however, does nothing.**Caveat** : if your MySQL activity log is world-readable, the user password **will be visible** . You have been warned.
- **Crypt**

  Allows passwords in the database to be of Unix crypt(3) form.
- **Empty**

  Allows empty passwords in the database, which match against **any** password the user may give. The database field must be a truly empty string; NULL values are not acceptable as empty passwords. **Be very careful if using this authentication method.**
- **OpenSSL**

  Allows passwords in the database to be of the form '{digest-name}hashed-value', where hashed-value is the base64-encoded digest of the passsword. Only available if you define HAVE_OPENSSL when

you compile proftpd and you link with OpenSSL's libcrypto library.

- **Plaintext**

  Allows passwords in the database to be in plaintext.

See also

Examples

SQLAuthTypes Crypt Empty

configures mod_sql to first attempt to verify the password using the Unix crypt(3) function, then, if that fails, determine if the password in the database is empty (thus matching any given password). If all of the configured authentication methods fail, mod_sql will fail to authenticate the user.

SQLBackend

Name

SQLBackend -- Set the SQL backend module

Synopsis

SQLBackend [`backend`]

`Default`
: Depends

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.3.0rc1 and later

Description

In 1.3.0rc1, the mod_sql module gained the ability to be compiled with multiple backend modules supported, e.g. to have both mod_sql_mysql and mod_sql_postgres usable in the same proftpd daemon. The SQLBackend directive configures which of these different database backends should be used.

If there is only one backend module compiled in, the SQLBackend directive is not needed. If there are multiple backend modules compiled and no SQLBackend directive is specified, then mod_sql will default to using the first backend module listed. For instance, if you configured proftpd using a configure command such as: ./configure --with-modules=mod_sql:mod_sql_postgres:mod_sql_mysql ... then mod_sql would default to using mod_sql_postgres as the backend module to use.

You might have multiple <VirtualHost> sections which use different SQL backends. Use "mysql" for the mod_sql_mysql module, and "postgres" for the mod_sql_postgres module.

See also

Examples

```
<VirtualHost 1.2.3.4>
  SQLBackend mysql
  ...
</VirtualHost>

<VirtualHost 5.6.7.8>
```

```
  SQLBackend postgres
  ...
</VirtualHost>
```

SQLConnectInfo

Name

SQLConnectInfo -- Specify connection information for the backend

Synopsis

SQLConnectInfo [`connection-info`][`[username]`][`[password]`][`[policy]`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.0 and later

Description

This directive deprecates 'MySQLInfo', 'PostgresInfo', and 'PostgresPort'.

The SQLConnectInfo directive configures the information necessary to connect to the backend database. The connection-info parameter specifies the database, host, port, and other backend-specific information. The optional username and password parameters specify a username and password to use when connecting to the database. Both default to NULL, which the backend will treat in some backend-specific manner. If you specify a password, you **must** specify a username. If no SQLConnectInfo directive is specified, mod_sql will disable itself.

Any given database backend has the opportunity, though not necessarily the responsibility, to check for syntax errors in the connection-info field at server startup, but you should not expect semantic errors (i.e., cannot connect to the database) to be caught until mod_sql attempts to connect for a given host.

A given database connection is governed by a connection policy that specifies when a connection should be opened and when it should be closed. There are three options:

- **PERSESSION**

  Open a database connection at the start of the session and close the database connection at the end of the session.
- number (**TIMED**)

  Timed database connections that close themselves after number seconds of inactivity.

If a connection policy is not specified, if the policy is not a number or is a number less than 1, or if the policy is the string "PERSESSION", the PERSESSION policy will be used.

If the connection policy is any number greater than 0, it specifies the number of seconds that a connection will be held open without activity. After that many seconds of database inactivity, the connection to the database will be closed. As soon as database activity starts again, the connection will be opened and the timer will restart.

The MySQL and Postgres backends' connection-info is expected to be of the form:

database[@hostname][:port]

hostname will default to a backend-specific hostname (which happens to be 'localhost' for both the MySQL and Postgres backends), and port will default to a backend-specific default port (3306 for the MySQL backend, 5432 for the Postgres backend).

From the MySQL documentation:

the value of host may be either a hostname or an IP address. If host is NULL or the string "localhost", a connection to the local host is assumed. If the OS supports sockets (Unix) or named pipes (Windows), they are used instead of TCP/IP to connect to the server.

From the PostgreSQL documentation:

If [the hostname] begins with a slash, it specifies Unix-domain communication rather than TCP/IP communication; the value is the name of the directory in which the socket file is stored. The default is to connect to a Unix-domain socket in /tmp.

If you plan to use the TIMED connection policy, consider the effect of directives such as DefaultRoot on local socket communication: once a user has been chroot()ed, the local socket file will probably not be available within the chroot directory tree, and attempts to reopen communication will fail. One way around this may be to use hardlinks within the user's directory tree. PERSESSION connections are not affected by this because the database will be opened prior to the chroot() call, and held open for the life of the session. Network communications are not affected by this problem. For example, while localhost would not work for MySQL since the MySQL client library will try to use socket communications for that host, 127.0.0.1 will work (as long as your database is setup to accept these connections).

See also

Examples

```
# Connect to the database 'ftpusers' via the default port at host
# 'foo.com'.  Use a NULL username and NULL password when connecting.
# A connection policy of PERSESSION is used.
SQLConnectInfo ftpusers@foo.com

# Connect to the database 'ftpusers' via port 3000 at host 'localhost'.
# Use the username 'admin' and a NULL password when connecting.
# A connection policy of PERSESSION is used.
SQLConnectInfo ftpusers:3000 admin
```

```
# Connect to the database 'ftpusers' via port 3000 at host 'foo.com'.
# Use the username 'admin' and password 'mypassword' when connecting.
# A connection policy of PERSESSION is used.
SQLConnectInfo ftpusers@foo.com:3000 admin mypassword

# Connect to the database 'ftpusers' via port 3000 at host 'foo.com'.
# Use a username of 'admin' and a password of 'mypassword' when
# connecting.  A 30 second timer of connection inactivity is activated.
SQLConnectInfo ftpusers@foo.com:3000 admin mypassword 30
```

Backends may require different information in the connection-info field; check your backend module for more detailed information.

SQLDefaultGID

Name

SQLDefaultGID -- Configure the default GID for users

Synopsis

SQLDefaultGID [`defaultgid`]

`Default`
: 65533

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.0 and later

Description

Sets the default GID for users. Must be greater than SQLMinID.

See also

SQLMinID SQLMinUserGID

SQLDefaultHomedir

Name

SQLDefaultHomedir -- Configure the default homedir

Synopsis

SQLDefaultHomedir [`path`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.5rc1 and later

Description

The SQLDefaultHomedir directive configures a default home directory for all users authenticated with this module, overriding any (deprecated) SQLHomedirField directive. If no home directory is set with either directive, authentication fails. This directive does not change the data retrieved from the database: if you specify a home directory field to SQLUserInfo, that field's data will be returned as the user's home directory, whether that data is a legal directory, or an empty string, or NULL.

See also

SQLUserInfo

Examples

SQLDefaultUID

Name

SQLDefaultUID -- Configure the default UID for users

Synopsis

SQLDefaultUID [`defaultuid`]

`Default`
: 65533

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.0 and later

Description

Sets the default UID for users. Must be greater than SQLMinID.

See also

SQLMinID SQLMinUserUID

SQLEngine

Name

SQLEngine -- Configure how mod_sql will operate

Synopsis

SQLEngine [`on|off|auth|log`]

`Default`
: SQLEngine on

`Context`
: server config, <Global>, <VirtualHost>, <Anonymous>

`Module`
: mod_sql

`Compatibility`
: 1.3.0rc1 and later

Description

The SQLEngine directive is used to specify how mod_sql will operate. By default, SQLEngine is on, and mod_sql will operate as normal. Setting SQLEngine to off will effectively disable the module.

In addition to on and off, SQLEngine accepts two other values: auth and log. If you wish to use mod_sql for authentication and not for logging (via SQLLog), use auth. Conversely, to do only SQLLog-type logging, and no authentication, use log.

This directive can be used to have <Anonymous> sections that do not use mod_sql (see the example below).

See also

Examples

```
<Anonymous ~ftp>
  ...
  SQLEngine off
  ...
</Anonymous>
```

SQLGroupInfo

Name

SQLGroupInfo -- Configure the group table and fields that hold group information

Synopsis

SQLGroupInfo [`group-table group-name gid members`]

`Default`
: "groups groupname gid members"

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.5rc1 and later

Description

The SQLGroupInfo directive configures the group table and fields that hold group information. The parameters for this directive are described below:

- grouptable

  Specifies the name of the table that holds group information.
- groupname

  Specifies the field in the group table that holds the group name.
- gid

  Specifies the field in the group table that holds the group's GID.
- members

  Specifies the field in the group table that holds the group members.

If you need to change any of these field names from the default, you need to specify all of them.

See also

Examples

SQLGroupWhereClause

Name

SQLGroupWhereClause -- Configure a WHERE clause for every group query

Synopsis

SQLGroupWhereClause [`where-clause`]

`Default`
: off

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.5rc1 and later

Description

The directive is used to configure a WHERE clause that is added to every group query. The WHERE clause must contain all relevant punctuation, and must not contain a leading "and".

Starting with ProFTPD 1.3.1rc1 the SQLGroupWhereClause also supports the variables supported by SQLNamedQuery except for the "%{n}" variable

See also

SQLNamedQuery

Examples

As an example of a possible use for this directive, imagine if your group table included a "LoginAllowed" field:

```
SQLGroupWhereClause "LoginAllowed = 'true'"
```

would be appended to every group-related query as the string:

```
" WHERE (LoginAllowed = 'true')"
```

SQLHomedirOnDemand

Name

SQLHomedirOnDemand -- Have mod_sql create home directories as needed [deprecated]

Synopsis

SQLHomedirOnDemand [`on|off`]

`Default`
: off

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.0 up to 1.3.1rc1

Description

This directive has been deprecated with ProFTPD 1.3.1rc1. Please use CreateHome instead.

The SQLHomedirOnDemand directive configures mod_sql to automatically create a user's home directory if that directory does not exist during the login process.

SQLLog

Name

SQLLog -- Log information to a database table

Synopsis

SQLLog [`cmd-set query-name ["IGNORE_ERRORS"]`

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.5rc1 and later

Description

This directive is used to log information to a database table. Multiple SQLLog directives can be in effect for any command; for example, a user changing directories can trigger multiple logging statements.

The first parameter to SQLLog, the cmd-set, is a comma-separated (no spaces) list of FTP commands for which this log command will trigger. The list of commands is too long to list in entirety; commands include CWD, DELE, HELP, LIST, MKD, MODE, NLST, PASS, PASV, PORT and many more. For the complete list check the FTP RFCs. Normally mod_sql will log events after they have completed successfully; in the case of the QUIT command, mod_sql logs prior to the server's processing of the command. (Note, however, that the client may not issue a QUIT before logging out; in this case, use a command of EXIT rather than QUIT. EXIT is not a real FTP command, but it is used here to provide a means for having SQLLog work whenever a session ends.)

FTP commands in the command set will only be logged if they complete successfully. Prefixing any command with "ERR_" will cause logging to occur only if there was an error in the command's processing. To log both errors and successful completion of a given command X, therefore, you'll need both "X" and "ERR_X" in your cmd-set.

The special command "*" matches all FTP commands, while "ERR_*" matches all errors.

The second parameter is the name of a query defined by a SQLNamedQuery directive. The query must be an UPDATE, INSERT, or FREEFORM type query; explicit SELECT queries will not be processed.

The third parameter is optional. If you add "IGNORE_ERRORS" as the third parameter, SQLLog will not check for errors in the processing of the named query. Any value for this parameter other than the string "IGNORE_ERRORS" (case-insensitive) will not cause errors to be ignored.

Normally, SQLLog directives are considered important enough that errors in their processing will cause mod_sql to abort the client session. References to non-existent named queries will not abort the client session, but may result in database corruption (in the sense that the expected database UPDATE or INSERT will not occur). Check your directives carefully.

See also

Examples

```
SQLLog PASS updatecount
SQLNamedQuery updatecount UPDATE "count=count+1 WHERE userid='%u'" users
```

together, these replicate the deprecated "SQLLoginCountField count" directive; if the current user was "joe", this would translate into the query "UPDATE users SET count=count+1 WHERE userid='joe'". This query would run whenever a user was first authenticated.

```
SQLLog CWD updatedir
SQLNamedQuery updatedir UPDATE "cwd='%d' where userid='%u'" users
```

together these replicate the logging side of the deprecated "SQLLogDirs cwd" directive; if the current user was "joe" and the current working directory were /tmp, this would translate into the query "UPDATE users SET cwd='/tmp' WHERE userid='joe'". This query would run whenever a user changed directories.

```
SQLLog RETR,STOR insertfileinfo
SQLNamedQuery insertfileinfo INSERT "'%f', %b, '%u@%v', now()" filehistory
```

would log the name of any file stored or retrieved, the number of bytes transferred, the user and host doing the transfer, and the time of transfer (at least in MySQL). This would translate into a query like: "INSERT INTO filehistory VALUES ('somefile', 12345, 'joe@joe.org', '21-05-2001 20:01:00')"

SQLLogFile

Name

SQLLogFile -- Specify a log file for mod_sql reporting and debugging

Synopsis

SQLLogFile [`file`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.8rc2 and later

Description

The SQLLogFile directive is used to a specify a log file for mod_sql reporting and debugging, and can be done a per-server basis. The file parameter must be the full path to the file to use for logging. Note that this path must not be to a world-writeable directory and, unless AllowLogSymlinks is explicitly set to on (generally a bad idea), the path must not be a symbolic link.

If file is "none", no logging will be done at all; this setting can be used to override a SQLLogFile setting inherited from a ;ltgt& context.

See also

Examples

SQLMinID

Name

SQLMinID -- Set SQLMinUserGID and SQLMinUserID in one place

Synopsis

SQLMinID [`minimum-id`]

`Default`
: 999

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.0 and later

Description

SQLMinID is a quick way of setting both SQLMinUserGID and SQLMinUserUID. These values are checked whenever retrieving a user's GID or UID.

See also

SQLMinUserGID SQLMinUserUID

SQLMinUserGID

Name

SQLMinUserGID -- Set a minimum GID

Synopsis

SQLMinUserGID [`minimum-gid`]

`Default`
: 999

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.5rc1 and later

Description

SQLMinUserGID is checked whenever retrieving a user's GID. If the retrieved value for GID is less than the value of SQLMinUserGID, it is reported as the value of SQLDefaultGID.

See also

Examples

SQLMinUserUID

Name

SQLMinUserUID -- Set a minimum UID

Synopsis

SQLMinUserUID [`minimum-uid`]

`Default`
: 999

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.5rc1 and later

Description

SQLMinUserUID is checked whenever retrieving a user's UID. If the retrieved value for UID is less than the value of SQLMinUserUID, it is reported as the value of SQLDefaultUID.

See also

Examples

SQLNamedQuery

Name

SQLNamedQuery -- Specify a query and an identifier for SQLShowInfo and SQLLog

Synopsis

SQLNamedQuery [`"name" limit|regex|ip value`]

`Default`
: (docs incomplete)

`Context`
: server config, <Global>, <VirtualHost>, <Anonymous>, <Limit>, .ftpaccess

`Module`
: mod_sql

`Compatibility`
: 1.2.5rc1 and later

Description

SQLNamedQuery specifies a query and an identifier (name) for later use by SQLShowInfo and SQLLog.

It is strongly recommended that you read documentation on the LogFormat and ExtendedLog directives, as the meta-sequences available to SQLNamedQuery are largely equivalent.

The first parameter, name, should be unique across all named queries and must not contain spaces. The result of re-using a name is undefined.

The second parameter, type, is the type of query, either "SELECT", "UPDATE", "INSERT", or "FREEFORM". See the note below for information on FREEFORM type queries.

The third parameter is the substance of the database query itself; this should match the form of the second parameter. The meta-sequences accepted are exactly equivalent to the LogFormat directive except the following are not accepted:

- %{FOOBAR}e

  For LogFormat, this logs the content of environment variable "FOOBAR". This is not bavailable in mod_sql.
- %{format}t and %t

  These two meta-sequences logged the local server time; they are not available in mod_sql. Your database undoubtedly provides another way to get the time; for example, MySQL provides the now() function.

and the following is in addition to the LogFormat meta-sequences:

- %d

  The current working directory or "-" if none.
- %{n}

  This meta-sequence is used internally by mod_sql and other third-party modules and patches to pass information to the database. Using this meta-sequence in anything other than an INSERT or UPDATE query is an error, and using this meta-sequence unless directed to by a third-party module or patch is also an error.
- %{env:VAR}

  Starting with ProFTPD 1.3.1rc1 the SQLNamedQuery directive is able to make use of environment variables in the format "%{env:VAR}". The value of the environment variable VAR will be substituted into the SQL statement.

The correct form of a query will be built from the directive arguments, except in the case of FREEFORM queries which will be sent directly to the database. The examples below show the way queries are built from the arguments.

The fourth parameter, table, is only necessary for UPDATE or INSERT type queries, but is required for those types.

Note: FREEFORM queries are a necessary evil; the simplistic query semantics of the UPDATE, INSERT, and SELECT type queries do not sufficiently expose the capabilities of most backend databases. At the same time, using a FREEFORM query makes it impossible for mod_sql to check whether the query type is appropriate, making sure that a SELECT query is not used in a SQLLog directive, for instance. Wherever possible, it is recommended that a specific query type be used.

See also

SQLShowInfo SQLLog LogFormat ExtendedLog

Examples

SQLNamedQuery count SELECT "count from users where userid='%u'"

creates a query named "count" which could be used by SQLShowInfo to inform a user of their login count. The actual query would look something like "SELECT count FROM users WHERE userid='matilda'" for user "matilda".

SQLNamedQuery updatecount UPDATE "count=count+1 WHERE userid='%u'" users

creates a query named "updatecount" which could be used by SQLLog to update a user login counter in the table users. The actual query would look something like "UPDATE users SET count=count+1 WHERE

userid='persephone'" for user "persephone".

SQLNamedQuery accesslog INSERT "now(), '%u'" accesslog

creates a query named "accesslog" which could be used by SQLLog to track access times by clients. The actual query would look something like "INSERT INTO accesslog VALUES (now(), 'pandora')" for user "pandora". Note that this may be too simplistic for your table structure, since most databases require data for all columns to be provided in an INSERT statement of this form. See the following FREEFORM query for an example of something which may suit your needs better.

SQLNamedQuery accesslog FREEFORM "INSERT INTO accesslog(date, user) VALUES (now(), '%u')"

creates a query named "accesslog" which could be used by SQLLog to track access times by clients. The actual query would look something like "INSERT INTO accesslog(date, user) VALUES (now(), 'tilda')" for user "tilda".

SQLNegativeCache

Name

SQLNegativeCache -- Enable negative caching for SQL lookups

Synopsis

SQLNegativeCache [`on off`]

`Default`
: SQLNegativeCache off

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_sql

`Compatibility`
: mod_sql v4.10 and later

Description

SQLNegativeCache specifies whether or not to cache negative responses from SQL lookups when using SQL for UID/GID lookups. Depending on your SQL tables, there can be a significant delay when a directory listing is performed as the UIDs not in the SQL database are repeatedly looked up in an attempt to present usernames instead of UIDs in directory listings. With SQLNegativeCache set to on, negative ("not found") responses from SQL queries will be cached and speed will improve on directory listings that contain many users not present in the SQL database.

See also

Examples

SQLRatios

Name

SQLRatios -- (docs incomplete)

Synopsis

SQLRatios [`"name" limit|regex|ip value`]

`Default`
: None

`Context`
: server config, <Global>

`Module`
: mod_sql

`Compatibility`
: 1.2.5rc1 and later

Description

mod_ratio is currently lacking a module maintainer. This directive is left over and not officially supported.

See also

Examples

(docs incomplete)

SQLRatioStats

Name

SQLRatioStats -- (docs incomplete)

Synopsis

SQLRatioStats [`"name" limit|regex|ip value`]

`Default`
: None

`Context`
: server config, <Global>

`Module`
: mod_sql

`Compatibility`
: 1.2.5rc1 and later

Description

mod_ratio is currently lacking a module maintainer. This directive is left over and not officially supported.

See also

Examples

(docs incomplete)

SQLShowInfo

Name

SQLShowInfo -- Create a message to be sent to the user after any successful command

Synopsis

SQLShowInfo [`cmd-set numeric query-string`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.5rc1 and later

Description

This directive creates a message to be sent to the user after any successful command.

The first parameter, the cmd-set, is a comma separated (no spaces) list of FTP commands for which this log command will trigger. The list of commands is too long to list in entirety; commands include: CWD, DELE, HELP, LIST, MKD, MODE, NLST, PASS, PASV, PORT and many more. For the complete list check the FTP RFCs.

FTP commands in the command set will only be triggered if they complete successfully. Prefixing any command with "ERR_" will show information only if there was an error in command processing. To send a message on both errors and successfull completion of a given command X, therefore, you'll need both "X" and "ERR_X" in your cmd-set.

The special command "*" matches all FTP commands, while "ERR_*" matches all errors.

The second parameter, numeric, specifies the numeric value of the message returned to the FTP client. Do not choose a number blindly: message numbers may be parsed by clients. In most cases you will want to use 214, the "Help message" numeric. It specifies that the information is only meant to be human readable.

The third parameter, query-string, is exactly equivalent to the query-string parameter to the SQLLog directive, with one addition:

- %{name}

  The first return value from the SQLNamedQuery identified by "name". There is currently no way to retrieve more than one value from the database at a time.

Any references to non-existent named queries, non-SELECT or -FREEFORM type queries, or references to queries which return a NULL first value, will be replaced with the string "{null}".

See also

Examples

```
SQLNamedQuery count SELECT "count from users where userid='%u'"
SQLShowInfo PASS "230" "You've logged on %{count} times, %u"
```

As long as the information is in the database, these two directives specify that the user will be greeted with their login count each time they successfully login. Note the use of the "230" numeric, which means "User logged in, proceed". "230" is appropriate in this case because the message will be sent immediately after their password has been accepted and the session has started.

SQLUserInfo

Name

SQLUserInfo -- Configure the user table and fields that hold user information

Synopsis

SQLUserInfo [`user-table user-name passwd uid gid home-dir shell`]

`Default`
: "users userid passwd uid gid homedir shell"

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.5rc1 and later

Description

The SQLUserInfo directive configures the user table and fields that hold user information. If you need to change any of these field names from the default, you must specify all of them, whether NULL or not. The parameters are described below:

- usertable

  Specifies the name of the table that holds user information.
- username

  Specifies the field in the user table that holds the username.
- passwd

  Specifies the field in the user table that holds the user's password.
- uid

  Specifies the field in the user table that holds the user's UID. When a UID is retrieved from the database it is checked against the value of SQLMinUserUID. If the field name is specified as "NULL" the database will not be queried for this value and the user's UID will be set to the value of SQLDefaultUID.
- gid

  Specifies the field in the user table that holds the user's GID. When a GID is retrieved from the database it is checked against the value of SQLMinUserGID. If the field name is specified as "NULL" the database will not be queried for this value and the user's GID will be set to the value of SQLDefaultGID.
- homedir

Specifies the field in the user table that holds the user's home directory. If the fieldname is specified as "NULL" the database will not be queried for this value and the user's home directory will be set to the value of SQLDefaultHomedir. If no home directory is set with either directive, user authentication will be automatically turned off.

- shell

  Specifies the field in the user table that holds the user's shell. If the fieldname is specified as "NULL" the database will not be queried and the shell will be reported as an empty string ("").

As of 1.2.9rc1, the SQLUserInfo directive accepts an alternate syntax:

```
SQLUserInfo custom:/name
```

where name refers to a configured SELECT SQLNamedQuery. This named query must return one row, and return the following columns, in this order: username, passwd, uid, gid, homedir, shell. The configured query may make use of the variables mentioned in the SQLLog description. This syntax allows the administrator a flexible way of constructing queries as needed. Note that if you want use the given USER name, you should use the %U variable, not %u; the latter requires the locally authenticated user name, which is exactly what SQLUserInfo is meant to provide.

See also

SQLLog SQLNamedQuery

Examples

SQLUserWhereClause

Name

SQLUserWhereClause -- Configure a WHERE clause for every user query<

Synopsis

SQLUserWhereClause [`where-clause`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_sql

`Compatibility`
: 1.2.5rc1 and later

Description

The directive is used to configure a WHERE clause that is added to every user query. The WHERE clause must contain all relevant punctuation, and must not contain a leading "and".

Starting with ProFTPD 1.3.1rc1 the SQLUserWhereClause also supports the variables supported by SQLNamedQuery except for the "%{n}" variable

See also

SQLNamedQuery

Examples

As an example of a possible use for this directive, imagine if your user table included a "LoginAllowed" field:

```
SQLUserWhereClause "LoginAllowed = 'true'"
```

would be appended to every user-related query as the string:

```
" WHERE (LoginAllowed = 'true')"
```

StoreUniquePrefix

Name

StoreUniquePrefix -- Set the prefix to be added to uniquely generated filenames

Synopsis

StoreUniquePrefix [`"prefix"`]

`Default`
: none

`Context`
: server config, <Global>, <VirtualHost>, <Global>, <Anonymous>, <Directory> .ftpaccess

`Module`
: mod_xfer

`Compatibility`
: 1.2.6rc1 and later

Description

The StoreUniquePrefix is used to configure a prefix for the generated unique random filenames used for the STOU FTP command. The last six characters of the filename will be random. Slashes are not allowed in the prefix string.

All valid filename characters are allowed except '/'

See also

Examples

StoreUniquePrefix "Wibble"

SyslogFacility

Name

SyslogFacility -- Set the facility level used for logging

Synopsis

SyslogFacility [`SyslogFacility facility-level`]

`Default`
: None

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 1.1.6 and later

Description

Proftpd logs its activity via the Unix syslog mechanism, which allows for several different general classifications of logging messages, known as "facilities." Normally, all authentication related messages are logged with the AUTHPRIV (or AUTH) facility [intended to be secure, and never seen by unwanted eyes], while normal operational messages are logged with the DAEMON facility. The SyslogFacility directive allows ALL logging messages to be directed to a different facility than the default. When this directive is used, ALL logging is done with the specified facility, both authentication (secure) and otherwise. The facility-level argument must be one of the following: AUTH (or AUTHPRIV), CRON, DAEMON, KERN, LPR, MAIL, NEWS, USER, UUCP, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6 or LOCAL7. See Also: SystemLog

See also

Examples

SyslogLevel

Name

SyslogLevel -- Set the verbosity level of system logging

Synopsis

SyslogLevel [`SyslogLevel emerg|alert|crit|error|warn|notice|info|debug`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.0rc2+cvs and later

Description

SyslogLevel adjusts the verbosity of the messages recorded in the error logs. The following levels are available, in order of decreasing significance: Level Description emerg Emergencies - system is unusable. alert Action must be taken immediately. crit Critical Conditions. error Error conditions. warn Warning conditions. notice Normal but significant condition. info Informational. debug Debug-level messages When a particular level is specified, messages from all other levels of higher significance will be reported as well. E.g., when SyslogLevel info is specified, then messages with log levels of notice and warn will also be posted. Using a level of at least crit is recommended.

See also

Examples

SystemLog

Name

SystemLog -- Redirect syslogging to a file

Synopsis

SystemLog [`SystemLog filename|NONE`]

`Default`
: None

`Context`
: server config

`Module`
: mod_log

`Compatibility`
: 1.1.6pl1 and later

Description

The SystemLog directive disables proftpd's use of the syslog mechanism and instead redirects all logging output to the specified filename. The filename argument should contain an absolute path, and should not be to a file in a nonexistent directory, in a world-writeable directory, or be a symbolic link (unless AllowLogSymlinks is set to on). Use of this directive overrides any facility set by the SyslogFacility directive. Additionally, the special keyword NONE can be used which disables all syslog style logging for the entire configuration.

See also

AllowLogSymlinks

Examples

TCPAccessFiles

Name

TCPAccessFiles -- Sets the access files to use

Synopsis

TCPAccessFiles [`allow-filename deny-filename`]

`Default`
: none

`Context`
: server config, <VirtualHost>, <Global>, <Anonymous>

`Module`
: mod_wrap

`Compatibility`
: 1.2.1 and later

Description

TCPAccessFiles specifies two files, an allow and a deny file, each of which contain the IP addresses, networks or name-based masks to be allowed or denied connections to the server. The files have the same format as the standard tcpwrappers hosts.allow/deny files.

Both file names are required. Also, the paths to both files must be the full path, with two exceptions: if the path starts with ~/, the check of that path will be delayed until a user requests a connection, at which time the path will be resolved to that user's home directory; or if the path starts with ~user/, where user is some system user. In this latter case, mod_wrap will attempt to resolve and verify the given user's home directory on start-up.

The service name for which mod_wrap will look in the indicated access files is proftpd by default; this can be configured via the TCPServiceName directive. There is a built-in precedence to the TCPAccessFiles, TCPGroupAccessFiles, and TCPUserAccessFiles directives, if all are used. mod_wrap will look for applicable TCPUserAccessFiles for the connecting user first. If no applicable TCPUserAccessFiles is found, mod_wrap will search for TCPGroupAccessFiles which pertain to the connecting user. If not found, mod_wrap will then look for the server-wide TCPAccessFiles directive. This allows for access control to be set on a per-server basis, and allow for per-user or per-group access control to be handled without interfering with the server access rules.

See also

TCPGroupAccessFiles, TCPServiceName, TCPUserAccessFiles

Examples

server-wide access files TCPAccessFiles /etc/ftpd.allow /etc/ftpd.deny # per-user access files, which are to be found in the user's home directory TCPAccessFiles ~/my.allow ~/my.deny

TCPAccessSyslogLevels

Name

TCPAccessSyslogLevels -- Sets the logging levels for mod_wrap

Synopsis

TCPAccessSyslogLevels [`allow-level deny-level`]

`Default`
: TCPAccessSyslogLevels info warn

`Context`
: server config, <VirtualHost>, <Global>, <Anonymous>

`Module`
: mod_wrap

`Compatibility`
: 1.2.1 and later

Description

ProFTPD can log when a connection is allowed, or denied, as the result of rules in the files specified in TCPAccessFiles, to the Unix syslog mechanism. A discussion on the syslog levels which can be used is given in the SyslogLevel directive.

The allow-level parameter sets the syslog level at which allowed connections are logged; the deny-level parameter sets the syslog level for denied connections.

See also

SyslogLevel

Examples

TCPAccessSyslogLevels debug warn

tcpBackLog

Name

tcpBackLog -- Control the tcp backlog in standalone mode

Synopsis

tcpBackLog [`tcpBackLog backlog-size`]

`Default`
: tcpBackLog 5

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The tcpBackLog directive controls the tcp "backlog queue" when listening for connections in standalone mode (see ServerType). It has no affect upon servers in inetd mode. When a tcp connection is established by the tcp/ip stack inside the kernel, there is a short period of time between the actual establishment of the connection and the acceptance of the connection by a user-space program. The duration of this latency period is widely variable, and can depend upon several factors (hardware, system load, etc). During this period tcp connections cannot be accepted, as the port that was previously "listening" has become filled with the new connection. Under heavy connection load this can result in occasional (or even frequent!) "connection refused" messages returned to the incoming client, even when there is a service available to handle requests. To eliminate this problem, most modern tcp/ip stacks implement a "backlog queue" which is simply a pre-allocation of resources necessary to handle backlog-size connections during the latency period. The larger the backlog queue, the more connections can be established in a very short time period. The trade-off, of course, is kernel memory and/or other kernel resources. Generally it is not necessary to use a tcpBackLog directive, unless you intend to service a large number of virtual hosts (see <VirtualHost>), or have a consistently heavy system load. If you begin to notice or hear of "connection refused" messages from remote clients, try setting a slightly higher value to this directive.

See also

Examples

TCPGroupAccessFiles

Name

TCPGroupAccessFiles -- Sets the access files to use

Synopsis

TCPGroupAccessFiles [`group-expression allow-filename deny-filename`]

`Default`
: none

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_wrap

`Compatibility`
: 1.2.1 and later

Description

TCPGroupAccessFiles allows for access control files, the same types of files required by TCPAccessFiles, to be applied to select groups. The given group-expression is a logical AND expression, which means that the connecting user must be a member of all the groups listed for this directive to apply. Group names may be negated with a ! prefix.

The rules for the filename paths are the same as for TCPAccessFiles settings.

See also

TCPAccessFiles, TCPUserAccessFiles

Examples

every member of group wheel must connect from restricted locations TCPGroupAccessFiles wheel /etc/ftpd-strict.allow /etc/ftpd-strict.deny # everyone else gets the standard access rules TCPGroupAccessFiles !wheel /etc/hosts.allow /etc/hosts.deny

tcpNoDelay

Name

tcpNoDelay -- Control the use of TCP_NODELAY

Synopsis

tcpNoDelay [`tcpNoDelay on|off`]

`Default`
: tcpNoDelay on

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.0pre3a and later

Description

The tcpNoDelay directive controls the use of the TCP_NODELAY socket option (which disables the Nagle algorithm). ProFTPd uses TCP_NODELAY by default, which usually is a benefit but this can occasionally lead to problems with some clients, so tcpNoDelay is provided as a way to disable this option. You will not normally need to use this directive but if you have clients reporting unusually slow connections, try setting this to off.

See also

Examples

TCPServiceName

Name

TCPServiceName -- Configures the name proftpd will use with mod_wrap

Synopsis

TCPServiceName [`name`]

`Default`
: TCPServiceName proftpd

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_wrap

`Compatibility`
: 1.2.1 and later

Description

TCPServiceName is used to configure the name of the service under which mod_wrap will check the allow/deny files. By default, this is the name of the program started, i.e. "proftpd". However, some administrators may want to use a different, more generic service name, such as "ftpd"; use this directive for such needs.

See also

TCPUserAccessFiles

Name

TCPUserAccessFiles -- Sets the access files to use

Synopsis

TCPUserAccessFiles [`user-expression allow-filename deny-filename`]

`Default`
: none

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_wrap

`Compatibility`
: 1.2.1 and later

Description

TCPUserAccessFiles allows for access control files, the same types of files required by TCPAccessFiles, to be applied to select users. The given user-expression is a logical AND expression. Listing multiple users in a user-expression does not make much sense; however, this type of AND evaluation allows for expressions such as "everyone except this user" with the use of the ! negation prefix.

The rules for the filename paths are the same as for TCPAccessFiles settings.

See also

TCPAccessFiles, TCPGroupAccessFiles

Examples

user admin might be allowed to connect from anywhere TCPUserAccessFiles admin /etc/ftpd-anywhere.allow /etc/ftpd-anywhere.deny # while every other user has to connect from LAN addresses TCPUserAccessFiles !admin /etc/ftpd-lan.allow /etc/ftpd-lan.deny

TimeoutIdle

Name

TimeoutIdle -- Sets the idle connection timeout

Synopsis

TimeoutIdle [`TimeoutIdle seconds`]

`Default`
: TimeoutIdle 600

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The TimeoutIdle directive configures the maximum number of seconds that proftpd will allow clients to stay connected without receiving any data on either the control or data connection. If data is received on either connection, the idle timer is reset. Setting TimeoutIdle to 0 disables the idle timer completely (clients can stay connected for ever, without sending data). This is generally a bad idea as a "hung" tcp connection which is never properly disconnected (the remote network may have become disconnected from the Internet, etc) will cause a child server to never exit (at least not for a considerable period of time) until manually killed See Also: TimeoutLogin, TimeoutNoTransfer

See also

Examples

TimeoutLinger

Name

TimeoutLinger -- Sets the timeout used for lingering closes

Synopsis

TimeoutLinger [`TimeoutLinger seconds`]

`Default`
: TimeoutLinger 180

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.10rc2 and later

Description

The TimeoutLinger directive configures the maximum number of seconds that proftpd will wait (or "linger") when closing a data connection. Once the data connection is closed, proftpd will send a message on the control connection indicating the closure. This delay is necessary for properly handling some FTP clients.

If the client aborts a transfer and there is a long delay, this lingering close is the most likely culprit. So if you encounter this delay, set TimeoutLinger to a low number to remove the delay.

See also

Examples

TimeoutLogin

Name

TimeoutLogin -- Sets the login timeout

Synopsis

TimeoutLogin [`TimeoutLogin seconds`]

`Default`
: TimeoutLogin 300

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_auth

`Compatibility`
: 0.99.0 and later

Description

The TimeoutLogin directive configures the maximum number of seconds a client is allowed to spend authenticating. The login timer is not reset when a client transmits data, and is only removed once a client has transmitted an acceptable USER/PASS command combination. See Also: TimeoutIdle, TimeoutNoTransfer

See also

Examples

TimeoutNoTransfer

Name

TimeoutNoTransfer -- Sets the connection without transfer timeout

Synopsis

TimeoutNoTransfer [`TimeoutNoTransfer seconds`]

`Default`
: TimeoutNoTransfer 300

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_xfer

`Compatibility`
: 0.99.0 and later

Description

The TimeoutNoTransfer directive configures the maximum number of seconds a client is allowed to spend connected, after authentication, without issuing a command which results in creating an active or passive data connection (i.e. sending/receiving a file, or receiving a directory listing). See Also: TimeoutIdle, TimeoutLogin

See also

Examples

TimeoutSession

Name

TimeoutSession -- Sets a timeout for an entire session

Synopsis

TimeoutSession [`seconds ["user"|"group"|"class" expression]`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>, <Anonymous>

`Module`
: mod_auth

`Compatibility`
: 1.2.6rc1 and later

Description

The TimeoutSession directive sets the maximum number of seconds a control connection between the proftpd server and an FTP client can exist after the client has successfully authenticated. If the seconds argument is set to 0, sessions are allowed to last indefinitely (the default).

The optional parameters are used to restrict the session time limit only to specific users. If "user" restriction is given, then expression is a user-expression specifying to which users the time limit applies. Similarly for the "group" restriction. For the "class" restriction, the expression is simply the name of connection class for whom the time limit will apply. Note that use of the "user" or "group" classifiers within an <Anonymous> context will not make much sense.

Example: # set a draconian session time limit TimeoutSession 60 # set session time limits for everyone except a few privileged users TimeoutSession 300 user !bob,!dave,!jenni

See also

Examples

Kick the user off after 60 minutes
TimeoutSession 3600

TimeoutStalled

Name

TimeoutStalled -- Sets the timeout on stalled downloads

Synopsis

TimeoutStalled [`TimeoutStalled seconds`]

`Default`
: TimeoutStalled 3600

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_xfer

`Compatibility`
: 1.1.6 and later

Description

The TimeoutStalled directive sets the maximum number of seconds a data connection between the proftpd server and an FTP client can exist but have no actual data transferred (i.e. "stalled"). If the seconds argument is set to 0, data transfers are allowed to stall indefinitely.

See also

Examples

TimesGMT

Name

TimesGMT -- Toggle time display between GMT and local

Synopsis

TimesGMT [`TimesGMT on|off`]

`Default`
: (versions 1.2.0pre9 and beyond) on

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_core

`Compatibility`

Description

Compatibility: 1.2.0pre9 and later The TimesGMT option causes the server to report all ls and MDTM times in GMT and not local time.

See also

Examples

TLSCACertificateFile

Name

TLSCACertificateFile -- Define a CA certificate used to verify your client certificates

Synopsis

TLSCACertificateFile [`CA certificate filename`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSCACertificateFile directive configures one file where you can assemble the certificates of Certification Authorities (CA) for your clients. The CA certificates in the file are then used to verify client certificates, if presented. Such a file is merely the concatenation of the various PEM-encoded CA certificates, in order of preference. This directive can be used in addition to, or as an alternative for, TLSCACertificatePath.

If neither TLSCACertificateFile nor TLSCACertificatePath are specified, the following message will appear in the TLSLog:

using default OpenSSL verification locations (see $SSL_CERT_DIR)

This means that the SSL_CERT_DIR environment variable, if set, will be used to determine the location of a CA certificate directory, to be used when verifying clients.

See also

TLSCACertificatePath

Examples

TLSCACertificateFile /etc/ftpd/ca-bundle.pem

TLSCACertificatePath

Name

TLSCACertificatePath -- Define a path to the CAs used to verify your client certificates

Synopsis

TLSCACertificatePath [`Path to your CA certificates`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSCACertificatePath directive sets the directory for the certificates of Certification Authorities (CAs) for your clients. These are used to verify the client certificates presented. This directive may be used in addition to, or as alternative for, TLSCACertificateFile.

The files in the configured directory have to be PEM-encoded, and are accessed through hash filenames. This means one cannot simply place the CA certificates there: one also has to create symbolic links named hash-value.N. The c_rehash utility that comes with OpenSSL can be used to create the necessary symlinks.

If neither TLSCACertificateFile nor TLSCACertificatePath are specified, the following message will appear in the TLSLog:

```
  using default OpenSSL verification locations (see $SSL_CERT_DIR)
[1]
```

This means that the SSL_CERT_DIR environment variable, if set, will be used to determine the location of a CA certificate directory, to be used when verifying clients.

See also

TLSCACertificateFile

Examples

TLSCACertificatePath /etc/ftpd/ca/

TLSCARevocationFile

Name

TLSCARevocationFile -- Define a file with your CA revocation certifcates

Synopsis

TLSCARevocationFile [`CA revocation filename`]

`Default`
: Define a file holding your Certificate Revocation Lists

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSCARevocationFile directive configures one file that can contain the Certificate Revocation Lists (CRL) of Certification Authorities (CA) for your clients. These CRLs are used during the verification of client certificates, if presented. Such a file is merely the concatenation of the various PEM-encoded CRL files, in order of preference. This directive can be used in addition to, or as an alternative for, TLSCARevocationPath.

See also

TLSCARevocationPath

Examples

TLSCARevocationFile /etc/ftpd/ca-crl-bundle.pem

TLSCARevocationPath

Name

TLSCARevocationPath -- Define a path to your CA revocation certificates

Synopsis

TLSCARevocationPath [`Path to a directory with CA revocation certificates`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSCARevocationPath directive sets the directory for the Certificate Revocation Lists (CRL) of Certification Authorities (CAs) for your clients. These are used during the verification of client certificates, if presented. This directive may be used in addition to, or as alternative for, TLSCARevocationFile.

The files in the configured directory have to be PEM-encoded, and are accessed through hash filenames. This means one cannot simply place the CRLs there: one also has to create symbolic links named hash-value.N. The c_rehash utility that comes with OpenSSL can be used to create the necessary symlinks.

See also

TLSCARevocationFile

Examples

TLSCARevocationPath /etc/ftpd/crl/

TLSCertificateChainFile

Name

TLSCertificateChainFile -- Define an all in one certification file

Synopsis

TLSCertificateChainFile [`TLSCertificateChainFile filename`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSCertificateChainFile directive sets the optional all-in-one file where you can assemble the certificates of Certification Authorities (CA) which form the certificate chain of the server certificate. This starts with the issuing CA certificate of the server certificate and can range up to the root CA certificate. Such a file is simply the concatenation of the various PEM-encoded CA Certificate files in certificate chain order. This server certificate chain is sent to the client, in addition to the server's certificate.

If TLSCertificateChainFile is not used, and TLSCACertificatePath is used, the certificate chain is built from the certificates in that path. TLSCertificateChainFile should be used as an alternative to TLSCACertificatePath for explicitly constructing the server certificate chain. It is especially useful to avoid conflicts with CA certificates when using client authentication. For although placing a CA certificate of the server certificate chain into the TLSCACertificatePath has the same effect for the certificate chain construction, it has the side-effect that client certificates issued by this same CA certificate are also accepted on client authentication. This is usually not what one expects.

Be careful: providing the certificate chain works only if you are using a single (either RSA or DSA) based server certificate. If you are using a coupled RSA+DSA certificate pair, this will work only if actually both certificates use the same certificate chain. Otherwise, clients will become confused.

See also

TLSCACertificateFile TLSCACertificatePath

Examples

TLSCertificateChainFile /etc/ftpd/client-ca-list.pem

TLSCipherSuite

Name

TLSCipherSuite -- Define a cipher list

Synopsis

TLSCipherSuite [`cipher-list`]

`Default`
: ALL:!ADH

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

How to put together a cipher list parameter:

Key Exchange Algorithms:
"kRSA" RSA key exchange
"kDHr" Diffie-Hellman key exchange (key from RSA cert)
"kDHd" Diffie-Hellman key exchange (key from DSA cert)
"kEDH' Ephemeral Diffie-Hellman key exchange (temporary key)

Authentication Algorithm:
"aNULL" No authentication
"aRSA" RSA authentication
"aDSS" DSS authentication
"aDH" Diffie-Hellman authentication

Cipher Encoding Algorithm:
"eNULL" No encodiing
"DES" DES encoding
"3DES" Triple DES encoding
"RC4" RC4 encoding
"RC2" RC2 encoding
"IDEA" IDEA encoding

MAC Digest Algorithm:
"MD5" MD5 hash function
"SHA1" SHA1 hash function
"SHA" SHA hash function (should not be used)

```
Aliases:
  "ALL"       all ciphers
  "SSLv2"     all SSL version 2.0 ciphers (should not be used)
  "SSLv3"     all SSL version 3.0 ciphers
  "EXP"       all export ciphers (40-bit)
  "EXPORT56"  all export ciphers (56-bit)
  "LOW"       all low strength ciphers (no export)
  "MEDIUM"    all ciphers with 128-bit encryption
  "HIGH"      all ciphers using greater than 128-bit encryption
  "RSA"       all ciphers using RSA key exchange
  "DH"        all ciphers using Diffie-Hellman key exchange
  "EDH"       all ciphers using Ephemeral Diffie-Hellman key exchange
  "ADH"       all ciphers using Anonymous Diffie-Hellman key exchange
  "DSS"       all ciphers using DSS authentication
  "NULL"      all ciphers using no encryption
```

Each item in the list may include a prefix modifier:

```
  "+"        move cipher(s) to the current location in the list
  "-"        remove cipher(s) from the list (may be added again by a
             subsequent list entry)
  "!"        kill cipher from the list (it may not be added again by a
             subsequent list entry)
```

If no modifier is specified the entry is added to the list at the current position. "+" may also be used to combine tags to

The OpenSSL command

```
openssl ciphers -v <list of ciphers>
```

may be used to list all of the ciphers and the order described by a specific .

See also

Examples

For example, all available ciphers not including ADH key exchange:

```
ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP
```

All algorithms including ADH and export but excluding patented algorithms:

```
HIGH:MEDIUM:LOW:EXPORT56:EXP:ADH:!kRSA:!aRSA:!RC4:!RC2:!IDEA
```

TLSDHParamFile

Name

TLSDHParamFile -- Define a file used in Diffie-Hellman key exchange

Synopsis

TLSDHParamFile [`Absolute path to the Diffie-Hellman param file`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSDHParamFile directive is used to configure a file that mod_tls will use when engaging in a Diffie-Hellman key exchange. Such a key exchange can be computationally intensive, in terms for parameter generation; to help speed up the process, the parameters used may be generated in advance, and stored in a file. The dhparam utility that comes with OpenSSL may be used to generate an appropriate file for this directive. The file parameter must be an absolute path.

See also

Examples

TLSDSACertificateFile

Name

TLSDSACertificateFile -- Point to the file containing the DSA certificate

Synopsis

TLSDSACertificateFile [`TLSDSACertificateFile filename`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSDSACertificateFile directive points to the PEM-encoded file containing the DSA certificate file for the server and optionally also the corresponding DSA private key file.

If the contained private key is encrypted, the administrator will be prompted for the passphrase when the daemon starts up, and when the daemon is restarted.

See also

TLSDSACertificateKeyFile

Examples

TLSDSACertificateKeyFile /etc/ftpd/server-dsa-key.pem

TLSDSACertificateKeyFile

Name

TLSDSACertificateKeyFile -- Point to the file containing the private DSA key

Synopsis

TLSDSACertificateKeyFile [`TLSDSACertificateKeyFile filename`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSDSACertificateKeyFile directive points to the PEM-encoded private key file for the server. If the private key is not combined with the certificate in the TLSDSACertificateFile, use this additional directive to point to the file with the standalone private key. When TLSDSACertificateFile is used and the file contains both the certificate and the private key, this directive need not be used. However, this practice is strongly discouraged. Instead we recommend you to separate the certificate and the private key.

If the contained private key is encrypted, the administrator will be prompted for the passphrase when the daemon starts up, and when the daemon is restarted.

See also

TLSDSACertificateKeyFile

Examples

TLSDSACertificateKeyFile /etc/ftpd/server-dsa-key.pem

TLSEngine

Name

TLSEngine -- Enable TLS/SSL connections

Synopsis

TLSEngine [`[ on off ]`]

`Default`
: off

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSEngine directive toggles the use of the SSL/TLS protocol engine (e.g. mod_tls). This is usually used inside a <VirtualHost> section to enable SSL/TLS sessions for a particular virtual host. By default mod_tls is disabled for both the main server and all configured virtual hosts.

See also

Examples

TLSLog

Name

TLSLog -- Specify a logfile for mod_tls's reporting on a per-server basis

Synopsis

TLSLog [`TLSLog filename`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSLog directive is used to specify a log file for mod_tls's reporting on a per-server basis. The file parameter given must be the full path to the file to use for logging.

See also

Examples

TLSOptions

Name

TLSOptions -- Configure optional behaviour of mod_tls

Synopsis

TLSOptions [`[ AllowDotLogin ] [ Allow PerUser ] [ ExportCertData ] [ NoCertRequest ] [ StdEnvVars ] [ dNSNameRequired ] [ iPAddressRquired ]`**]**

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSOptions directive is used to configure various optional behavior of mod_tls. The currently implemented options are:

- AllowDotLogin

  By default, mod_tls still requires that a user supply a password for authentication, even if a valid client certificate is presented. If this option is enabled, mod_tls will check in the user's home directory for a .tlslogin file, which should contain one or more PEM-encoded certificates. If the certificate presented by the client, if any, matches a certificate in this .tlslogin file, the user will be considered authenticated. The server will still prompt for a password, and if the user's .tlslogin does not exist, or does not contain the client's certificate, then the server will fallback to using the password for authentication.
- AllowPerUser

  This option affects how mod_tls evaluates any TLSRequired directives. Usually mod_tls will reject any FTP commands, when TLSRequired on or TLSRequired ctrl is in effect, if the client has not successfully negotiated a SSL/TLS handshake. The FTPS specification requires that the SSL/TLS handshake occur, via the AUTH FTP command, before the USER and PASS commands. This means that mod_tls does not know the identity of the connecting client when enforcing TLSRequired. If this AllowPerUser is used, mod_tls will wait until after the PASS command has been processed to enforce any TLSRequired settings.

  Important: if AllowPerUser is used, even if TLSRequired on or TLSRequired ctrl are in effect, it will be possible for the connecting client to send usernames and passsword unprotected before mod_tls rejects the connection. This results in a slightly weaker security policy enforcement; please consider

carefully if this tradeoff is acceptable for your site.

- ExportCertData

  Sets the following environment variables, if applicable. Note that doing so increases the memory size of the process quite a bit:

Table 1-1. Enviroment variables

TLS_SERVER_CERT	Server certificate, PEM-encoded
TLS_CLIENT_CERT	CLient certificate, PEM-encoded
TLS_CLIENT_CERT_CHAINn	PEM-encoded certificates in client certificate chain

- NoCertRequest

  Some FTP clients are known to be buggy when handling a server's certificate request. This option causes the server not to include such a request during an SSL handshake.

- StdEnvVars

  Sets the following environment variables, if applicable. These environment variables are then avaiable for use, such as in LogFormats. Note that doing so increases the memory size of the process quite a bit: increases the memory size of the process quite a bit:

Table 1-2. Enviroment variables

FTPS	Present if FTP over SSL/TLS is being used
TLS_PROTOCOL	SSL protocol version (e.g. SSLv3, TLSv1)
TLS_SESSION_ID	Hex-encoded SSL session ID
TLS_CIPHER	Cipher specification name
TLS_CIPHER_EXPORT	Present if cipher is an export cipher
TLS_CIPHER_KEYSIZE_POSSIBLE	Number of cipher bits possible
TLS_CIPHER_KEYSIZE_USED	Number of cipher bits used
TLS_LIBRARY_VERSION	OpenSSL version
TLS_CLIENT_M_VERSION	Client certificate version
TLS_CLIENT_M_SERIAL	Client certificate serial number
TLS_CLIENT_S_DN	Subject DN of client certificate
TLS_CLIENT_S_DN_x509	Component of client certificate's Subject DN, where x509 is a component of a X509 DN: C,CN,D,I,G,L,O,OU,S,ST,T,UID,Email
TLS_CLIENT_I_DN	Issuer DN of client certificate
TLS_CLIENT_I_DN_x509	Component of client certificate's Issuer DN, where x509 is a component of a X509 DN: C,CN,D,I,G,L,O,OU,S,ST,T,UID,Email
TLS_CLIENT_V_START	Start time of client certificate validity
TLS_CLIENT_V_END	End time of client certificate validity
TLS_CLIENT_A_SIG	Client certificate's signature algorithm

TLS_CLIENT_A_KEY	Client certificate's public key algorithm
TLS_CLIENT_CERT	Client certificate, PEM-encoded
TLS_CLIENT_CERT_CHAINn	PEM-encoded certificates in client certificate chain
TLS_SERVER_M_VERSION	Server certificate version
TLS_SERVER_M_SERIAL	Server certificate serial number
TLS_SERVER_S_DN	Subject DN of server certificate
TLS_SERVER_S_DN_x509	Component of server certificate's Subject DN, where x509 is a component of a X509 DN: C,CN,D,I,G,L,O,OU,S,ST,T,UID,Email
TLS_SERVER_I_DN	Issuer DN of server certificate
TLS_SERVER_I_DN_x509	Component of server certificate's Issuer DN, where x509 is a component of a X509 DN: C,CN,D,I,G,L,O,OU,S,ST,T,UID,Email
TLS_SERVER_V_START	Start time of server certificate validity
TLS_SERVER_V_END	End time of server certificate validity
TLS_SERVER_A_SIG	Server certificate's signature algorithm
TLS_SERVER_A_KEY	Server certificate's public key algorithm
TLS_SERVER_CERT	Server certificate, PEM-encoded

- dNSNameRequired

  This option will cause mod_tls to perform checks on a client's certificate once the SSL handshake has been completed: the client's certificate will be searched for the subjectAltName X509v3 extension, and, in that extension, the dNSName value will be looked up. Unless a dNSName value is present, and the value matches the DNS name to which the client's IP address resolves, the SSL session is closed. This check is only performed during SSL handshakes on the control channel. Note that if UseReverseDNS is off, this option is automatically disabled.
- iPAddressRequired

  This option will cause mod_tls to perform checks on a client's certificate once the SSL handshake has been completed: the client's certificate will be searched for the subjectAltName X509v3 extension, and, in that extension, the iPAddress value will be looked up. Unless an iPAddress value is present, and the value matches the IP address of the client, the SSL session is closed. This check is only performed during SSL handshakes on the control channel.

See also

Examples

TLSOptions iPAddressRequired StdEnvVars

TLSPassPhraseProvider

Name

TLSPassPhraseProvider -- FIXFIXFIX

Synopsis

TLSPassPhraseProvider [`"name" limit|regex|ip value`]

`Default`
: FIXFIXFIX

`Context`
: server config, <Global>, <VirtualHost>, <Anonymous>, <Limit>, .ftpaccess

`Module`
: mod_tls

`Compatibility`
: 1.3.1rc1 and later

Description

FIX FIX FIX

See also

Examples

FIXFIXFIX

FIXFIX

TLSProtocol

Name

TLSProtocol -- Define the SSL/TLS protocol version mod_tls should use

Synopsis

TLSProtocol [`[ SSLv23 SSLv3 TLSv1 ]`]

`Default`
: SSLv23

`Context`
: server config

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSProtocol directive is used to configure the SSL/TLS protocol versions that mod_tls should use when establishing SSL/TLS sessions. Clients can then only connect using the configured protocol.

Since the protocol version used by mod_tls is set only once, when the daemon starts, the TLSProtocol directive is only allowed in the "server config" context.

The allowed protocols are:

SSLv23 Compatibility mode, used to allow both SSLv3 and TLSv1

SSLv3 Allow only SSLv3

TLSv1 Allow only TLSv1

All use of SSLv2 is disabled. SSLv2 should not be used.

See also

Examples

TLSRandomSeed

Name

TLSRandomSeed -- Define a file for PRNG seeding

Synopsis

TLSRandomSeed [`Absolute path to the file`]

`Default`
: openssl-dir /.rnd

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSRandomSeed directive configures the file that mod_tls will use for seeding the PRNG. seed must be an absolute path.

When the daemon shuts down, any random data left will be written out to the random seed file, so that that data may be used for seeding when the daemon is started again.

See also

Examples

TLSRandomSeed /etc/ftpd/server.rnd

TLSRenegotiate

Name

TLSRenegotiate -- Configure SSL renegotiations

Synopsis

TLSRenegotiate [`["ctrl" secs] ["data" Kbytes] ["timeout" secs]|["required" on|off]|"none"`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSRenegotiate directive is used to configure when SSL renegotiations are to occur. Renegotiations, and thus this directive, are only supported by mod_tls if the version of OpenSSL installed is 0.9.7 or greater.

If supported, renegotiations will occur on control channels that have been established for four hours by default, and on data channels that have transferred over one gigabyte of data by default. When renegotiations are requested, the client is given a timeout of 30 seconds, by default, to perform the renegotiation. To change the default control channel renegotiation timeout, use ctrl followed by a number, greater than zero, in seconds. Use data followed by a number, greater than zero, of kilobytes to change the default data channel renegotiation threshhold. The timeout parameter, followed by a positive number of seconds, is used to change the length of time given to a client to complete a requested renegotiation, after which the SSL session will be shutdown. By default, mod_tls will require that the client comply with the requested renegotiation within the TLSRenegotiate timeout. If, however, the client is unwilling or unable to do so, and the daemon needs to support these clients, set required to off. Doing so will cause renegotiations to be requested, but not required.

By default, mod_tls will perform renegotiations if supported, on the control channel after 4 hours, and on the data channel after one gigabyte of transferred data. The default timeout for a renegotiation is 30 seconds.

Use none to disable all renegotiation requirements.

See also

Examples

```
# Change renegotiations to occur on control channels after 1 hour
TLSRenegotiate ctrl 3600

# Change renegotiations to occur on data channels after 500 MB
TLSRenegotiate data 512000

# Change renegotiations so that they are not required, only requested
TLSRenegotiate required off

# Change only the timeout for renegotiations to be 5 minutes
TLSRenegotiate timeout 300

# Change all of the above renegotiation threshholds using one directive
TLSRenegotiate ctrl 3600 data 512000 required off timeout 300

# To disable renegotiations entirely
TLSRenegotiate none
```

TLSRequired

Name

TLSRequired -- Require SSL/TLS on the control and/or data channel

Synopsis

TLSRequired [`on | off | ctrl | data | auth | auth+data`]

`Default`
: off

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

: 1.3.1rc1 and later provide the auth and auth+data options

Description

The TLSRequired directive is used to define a basic security policy, one that dictates whether the control channel, or data channel, or both, of an FTP session must occur over SSL/TLS.

The "on" parameter enables SSL/TLS requirements on both control and data channels; "off" disables the requirements on both channels. Use "ctrl" and "data" to require SSL/TLS on either channel individually.

The "auth" parameter requires that SSL/TLS be used on the control channel, but only for authentication. To use this setting and require SSL/TLS for data transfers, use the "auth+data" parameter.

This "auth+data" parameter allows a very specific security policy: authentication via the USER/PASS commands must be protected via SSL/TLS, as must the data channel, but after authenticating, the client can request that protection be removed from the control channel. This policy allows clients to use the CCC (Clear Command Channel) command, which in turn enables SSL/TLS protected data transfers that are operate better with firewalls that monitor the FTP control channel.

See also

Examples

```
# Require SSL/TLS on the control channel, so that passwords are not sent
# in the clear.
```

```
TLSRequired ctrl

# Require SSL/TLS on both channels.
TLSRequired on

# Allow the client to use the CCC command to remove SSL/TLS from the
# control channel, but only after authentication has been performed.
# Still enforce the policy of using SSL/TLS for data transfers.
#
# Note that if we did not need to protect data transfers, we would
# set 'TLSRequired auth' instead of using 'TLSRequired auth+data'.
TLSRequired auth+data
```

TLSRSACertificateFile

Name

TLSRSACertificateFile -- Point to the file containing the RSA certificate

Synopsis

TLSRSACertificateFile [`TLSRSACertificateFile filename`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSRSACertificateFile directive points to the PEM-encoded file containing the RSA certificate file for the server and optionally also the corresponding RSA private key file.

If the contained private key is encrypted, the administrator will be prompted for the passphrase when the daemon starts up, and when the daemon is restarted.

See also

TLSRSACertificateKeyFile

Examples

```
TLSRSACertificateFile /etc/ftpd/server-rsa-cert.pem
```

TLSRSACertificateKeyFile

Name

TLSRSACertificateKeyFile -- Point to the file containing the private RSA key

Synopsis

TLSRSACertificateKeyFile [`TLSRSACertificateKeyFile filename`]

`Default`
: None

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSRSACertificateKeyFile directive points to the PEM-encoded private key file for the server. If the private key is not combined with the certificate in the TLSRSACertificateFile, use this additional directive to point to the file with the standalone private key. When TLSRSACertificateFile is used and the file contains both the certificate and the private key, this directive need not be used. However, this practice is strongly discouraged. Instead we recommend you to separate the certificate and the private key.

If the contained private key is encrypted, the administrator will be prompted for the passphrase when the daemon starts up, and when the daemon is restarted.

See also

Examples

TLSRSACertificateKeyFile /etc/ftpd/server-rsa-key.pem

TLSVerifyClient

Name

TLSVerifyClient -- Configure how to candle certificates presented by clients --

Synopsis

TLSVerifyClient [`on off`]

`Default`
: off

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSVerifyClient directive configures how mod_tls handles certificates presented by clients. If off, the module will accept the certificate and establish an SSL/TLS session, but will not verify the certificate. If on, the module will verify a client's certificate and, furthermore, will fail all SSL handshake attempts unless the client presents a certificate when the server requests one. Note that the server can be configured to not request a client certificate via the TLSOptions directive's "NoCertRequest" parameter.

See also

Examples

TLSVerifyDepth

Name

TLSVerifyDepth -- Define how deeply mod_tls should verify a client certificate

Synopsis

TLSVerifyDepth [`depth`]

`Default`
: 9

`Context`
: server config, <Global>, <VirtualHost>

`Module`
: mod_tls

`Compatibility`
: 1.2.7rc1 and later

Description

The TLSVerifyDepth directive sets how deeply mod_tls should verify before deciding that the client does not have a valid certificate. The depth actually is the maximum number of intermediate certificate issuers, i.e. the number of CA certificates which are allowed to be followed while verifying the client certificate. A depth of 0 means that only self-signed client certificates are accepted, a depth of 1 means the client certificate can be self-signed or has to be signed by a CA which is directly known to the server (i.e. the CA's certificate is under TLSCACertificatePath), etc.

See also

Examples

```
TLSVerifyDepth 10
```

TransferLog

Name

TransferLog -- Specify the path to the transfer log

Synopsis

TransferLog [`TransferLog filename|NONE`]

`Default`
: TransferLog /var/log/xferlog

`Context`
: server config, <Anonymous>, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.1.4 and later

Description

The TransferLog directive configures the full path to the "wu-ftpd style" file transfer log. Separate log files can be created for each Anonymous and/or VirtualHost. Additionally, the special keyword NONE can be used, which disables wu-ftpd style transfer logging for the context in which the directive is used (only applicable to version 1.1.7 and later). See Also: ExtendedLog, LogFormat

See also

Examples

TransferRate

Name

TransferRate -- Configure upload, download transfer rates

Synopsis

TransferRate [`cmds`][`kilobytes-per-sec[:free-bytes]`][`["user"|"group"|"class" expression]`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>, <Anonymous>, <Directory>, .ftpaccess

`Module`
: mod_xfer

`Compatibility`
: 1.2.8rc1 and later

Description

The TransferRate directive is used to set transfer rates limits on the transfer of data. This directive allows for transfer rates to be set in a wide variety of contexts, on a per-command basis, and for certain subsets of users. Note that this limit only applies to a single connection, and not to the overall transfer rate of the server.

The cmds parameter may be an comma-separated list of any of the following commands: APPE, RETR, STOR, and STOU.

The kilobytes-per-sec parameter is the actual transfer rate to be applied.

The free-bytes parameter, if configured, allows that many bytes to be transferred before the rate controls are applied. This allows for clients transferring small files to be unthrottled, but for larger files, such as MP3s and ISO images, to be throttled.

The optional parameters are used to restrict the application of the rate controls only to specific users. If the "user" restriction is given, then expression is a user-expression specifying to which users the rate applies. Similarly for the "group" restriction. For the "class" restriction, the expression is simply the name of connection class for whom the rate will apply.

Examples

Limit downloads for everyone except the special group of users TransferRate RETR 1.5 group !special-users

Limit uploads (and appends!) to the prolific users in the # lotsofuploadfiles.net domain. This presumes that a Class has been defined # for that domain, and that that Class has been named "uploaders". Let them # upload small files without throttling, though. TransferRate APPE,STOR 8.0:1024 class uploaders

Umask

Name

Umask -- Set the default Umask

Synopsis

Umask [`Umask file octal-mask [directory octal-mask]`]

`Default`
: None

`Context`
: server config, <Anonymous>, <VirtualHost>, <Directory>, <Global>, .ftpaccess

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

Umask sets the mask applied to newly created file and directory permissions within a given context. By default, the Umask in the server configuration, <VirtualHost> or <Anonymous> block is used, unless overridden by a "per-directory" Umask setting. Any arguments supplied must be an octal number, in the format 0xxx. An optional second argument can specify a Umask to be used when creating directories. If a second argument isn't specified, directories are created using the default Umask in the first argument. For more information on umasks, consult your operating system documentation/man pages.

Proftpd will not create files that have the execution bit turned on, this is a security driven design decision. The permissions of the uploaded file can be changed by issuing a SITE CHMOD command can be used to change the mode of the uploaded file. Syntax of the command is: SITE CHMOD <mode> <file>.

See also

Examples

UnsetEnv

Name

UnsetEnv -- (docs incomplete)

Synopsis

UnsetEnv [`key`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.2.10rc1 and later

Description

(docs incomplete)

See also

Examples

(docs incomplete)

UseFtpUsers

Name

UseFtpUsers -- Block based on /etc/ftpusers

Synopsis

UseFtpUsers [`UseFtpUsers on|off`]

`Default`
: UseFtpUsers on

`Context`
: server config, <Anonymous>, <VirtualHost>, <Global>

`Module`
: mod_auth

`Compatibility`
: 0.99.0 and later

Description

Legacy FTP servers generally check a special authorization file (typically /etc/ftpusers) when a client attempts to authenticate. If the user's name is found in this file, FTP access is denied. For compatibility sake, proftpd defaults to checking this file during authentication. This behavior can be suppressed using the UseFtpUsers configuration directive.

See also

Examples

UseGlobbing

Name

UseGlobbing -- Toggles use of glob() functionality

Synopsis

UseGlobbing [`on|off`]

`Default`
: UseGlobbing on

`Context`
: server config, <VirtualHost>, <Global>, <Anonymous>

`Module`
: mod_ls

`Compatibility`
: 1.2.5rc1 and later

Description

The UseGlobbing directive controls use of glob() functionality, which is needed for supporting wildcard characters such as *.

See also

UseIPv6

Name

UseIPv6 -- Disable IPv6 support

Synopsis

UseIPv6 [`"on"|"off"`]

`Default`
: UseIPv6 on

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 1.3.1rc1 and later

Description

This directive enables or disables the IPv6 support within proftpd. It's also possible to control this behaviour with command-line options.

-4, --ipv4 Support IPv4 functionality only
-6, --ipv6 Support IPv6 functionality

See also

Examples

proftpd -4

Start Proftpd only with IPv4 functionality enabled.

User

Name

User -- Set the user the daemon will run as

Synopsis

User [`User userid`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The User directive configures which user the proftpd daemon will normally run as. By default, proftpd runs as root which is considered undesirable in all but the most trustful network configurations. The User directive used in conjunction with the Group directive instructs the daemon to switch to the specified user and group as quickly as possible after startup. On some unix variants, the daemon will occasionally switch back to root in order to accomplish a task which requires super-user access. Once the task is completed, root privileges are relinquished and the server continues to run as the specified user and group. When applied to a <VirtualServer> block, proftpd will run as the specified user/group on connections destined for the virtual server's address or port. If either User or Group is applied to an <Anonymous> block, proftpd will establish an anonymous login when a user attempts to login with the specified userid, as well as permanently switching to the corresponding uid/gid (matching the User/Group parameters found in the anonymous block) after login. Note: When an authorized unix user is authenticated and logs in, all former privileges are released, the daemon switches permanently to the logged in user's uid/gid, and is never again capable of switching back to root or any other user/group.

See also

Examples

UserAlias

Name

UserAlias -- Alias a username to a system user

Synopsis

UserAlias [`login-user real-user`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_auth

`Compatibility`
: 0.99.0 and later

Description

ProFTPD requires a real username/uid when authenticating users as provided by PAM, AuthUserFile or another authentication mechanism. There are however times when additional aliases are required but it is undesirable to provide additional login accounts.

UserAlias provides a mechanism to do this, a typical and common example is within Anonymous configuration blocks. It is normal for the server to use 'ftp' as the primary authentication user, however it is common practice for users to login using "anonymous". This is achieved by adding the following to the config file.

See also

Examples

UserAlias anonymous ftp

UserDirRoot

Name

UserDirRoot -- Set the chroot directory to a subdirectory of the anonymous server

Synopsis

UserDirRoot [`UserDirRoot on|off`]

`Default`
: off

`Context`
: <Anonymous>

`Module`
: mod_auth

`Compatibility`
: 1.2.0pre2 and later

Description

When set to true, the chroot base directory becomes a subdirectory of the anonymous ftp directory, based on the username of the current user. For example, assuming user "foo" is aliased to "ftp", logging in as "foo" causes proftpd to run as real user ftp, but to chroot into ~ftp/foo instead of just ~ftp.

See also

Examples

UseReverseDNS

Name

UseReverseDNS -- Toggle rDNS lookups

Synopsis

UseReverseDNS [`UseReverseDNS on|off`]

`Default`
: UseReverseDNS on

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 1.1.7 and later

Description

Normally, incoming active mode data connections and outgoing passive mode data connections have a reverse DNS lookup performed on the remote host's IP address. In a chroot environment (such as <Anonymous> or DefaultRoot), the /etc/hosts file cannot be checked and the only possible resolution is via DNS. If for some reason, DNS is not available or improperly configured this can result in proftpd blocking ("stalling") until the libc resolver code times out. Disabling this directive prevents proftpd from attempting to reverse-lookup data connection IP addresses.

See also

Examples

UserOwner

Name

UserOwner -- Set the user ownership of new files / directories

Synopsis

UserOwner [`UserOwner username`]

`Default`
: None

`Context`
: <Anonymous>, <Directory>

`Module`
: mod_core

`Compatibility`
: 1.2pre11 and later

Description

The UserOwner directive configures which user all newly created directories and files will be owned by, within the context that UserOwner is applied to. The user ID of username cannot be 0 (root). Where it is used, the GroupOwner directive is not restricted to groups that the current user is a member of.

See also

GroupOwner

Examples

UserPassword

Name

UserPassword -- Creates a hardcoded username/password pair

Synopsis

UserPassword [`UserPassword userid hashed-password`]

`Default`
: None

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_auth

`Compatibility`
: 0.99.0pl5 and later

Description

The UserPassword directive creates a password for a particular user which overrides the user's normal password in /etc/passwd (or /etc/shadow). The override is only effective inside the context to which UserPassword is applied. The hashed-password argument is a cleartext string which has been passed through the standard unix crypt() function. Do NOT use a cleartext password. This can be useful when combined with UserAlias to provide multiple logins to an Anonymous FTP site. See Also: GroupPassword

See also

Examples

UserRatio

Name

UserRatio -- Ratio directive

Synopsis

UserRatio [`UserRatio foo1 foo2 foo3`]

`Default`
: None known

`Context`
: <Directory>, <Anonymous>, <Limit>,.ftpaccess

`Module`
: mod_ratio

`Compatibility`
: at least 1.2.0 and later

Description

The UserRatio directive .... Example: UserRatio

See also

Examples

UseSendfile

Name

UseSendfile -- Toggles use of sendfile() functionality

Synopsis

UseSendfile [`on|off`]

`Default`
: UseSendfile on

`Context`
: server config, <VirtualHost>, <Global>

`Module`
: mod_xfer

`Compatibility`
: 1.3.0rc1 and later

Description

The UseSendfile directive controls use of sendfile functionality, which is an optimization for sending files to clients. Use of sendfile functionality avoids separate read and send operations, and buffer allocations. But on some platforms or within some filesystems, it is better to disable this feature to avoid operational problems:

```
*  Some platforms may have broken sendfile support that the build system
   did not detect, especially if the binaries were built on another box
   and moved to such a machine with broken sendfile support.

* On Linux the use of sendfile triggers TCP-checksum offloading bugs on
  certain networking cards when using IPv6.

* With a network-mounted directories (e.g. NFS or SMB), the kernel may be
  unable to serve the network file through its own cache.
```

Note that if sendfile support is enabled, tools like ftpwho and ftptop will not show the transfer rate for downloads. These tools work by reading the ScoreboardFile, and the ScoreboardFile is updated periodically during uploads and downloads. However, when sendfile support is used, the ScoreboardFile does not have a chance to be updated. This is only true for downloads; the tools will continue to show the transfer rate for uploads.

UseUTF8

Name

UseUTF8 -- FIXFIXFIX

Synopsis

UseUTF8 [`"name" limit|regex|ip value`]

`Default`
: FIXFIXFIX

`Context`
: server config, <Global>, <VirtualHost>, <Anonymous>, <Limit>, .ftpaccess

`Module`
: mod_core

`Compatibility`
: 1.3.1rc1 and later

Description

FIX FIX FIX

See also

Examples

FIXFIXFIX

FIXFIX

VirtualHost

Name

VirtualHost -- Define a virtual ftp server

Synopsis

VirtualHost [`<VirtualHost addresses seperated by spaces>`]

`Default`
: None

`Context`
: server config

`Module`
: mod_core

`Compatibility`
: 0.99.0 and later

Description

The VirtualHost configuration block is used to create an independent set of configuration directives that apply to a particular hostname or IP address. It is often used in conjunction with system level IP aliasing or dummy network interfaces in order to establish one or more "virtual" servers which all run on the same physical machine. The block is terminated with a </VirtualHost> directive. By utilizing the Port directive inside a VirtualHost block, it is possible to create a virtual server which uses the same address as the master server, but listens on a separate tcp port (incompatible with ServerType inetd). When proftpd starts, virtual server connections are handled in one of two ways, depending on the ServerType setting: inetd The daemon examines the destination address and port of the incoming connection handed off from inetd. If the connection matches one of the configured virtual hosts, the connection is serviced based on the appropriate configuration. If no virtual host matches, and the main server does not match, the client is informed that no server is available to service their requests and disconnected. standalone After parsing the configuration file, the daemon begins listening for connections on all configured ports, spawning child processes as necessary to handle connections for either the main server or any virtual servers. Because of the method that the daemon uses to listen for connections when in standalone mode, it is possible to support an exceedingly large number of virtual servers, potentially exceeding the number of per-process file descriptors. This is due to the fact that a single file descriptor is used to listen to each configured port, regardless of the number of addresses being monitored. Note that it may be necessary to increase the tcpBackLog value on heavily loaded servers in order to avoid kernel rejected client connections ("Connection refused").

Starting with ProFTPD 1.3.0rc1 it's possible to use more than one FQDN or IP Address. With this change the old Bind directive has been deprecated.

See also

DefaultAddress

Examples

<VirtualHost host1.domain.com host2.domain.com> ... </VirtualHost>

WtmpLog

Name

WtmpLog -- Toggle logging to wtmp

Synopsis

WtmpLog [`WtmpLog on|off|NONE`]

`Default`
: WtmpLog on

`Context`
: server config, <VirtualHost>, <Anonymous>, <Global>

`Module`
: mod_core

`Compatibility`
: 1.1.7 and later

Description

The WtmpLog directive controls proftpd's logging of ftp connections to the host system's wtmp file (used by such commands as `last'). By default, all connections are logged via wtmp. Please report any corrections or additions via http://bugs.proftpd.net/

See also

Examples

Chapter 2. List of modules

mod_auth

Name

mod_auth -- Authentication module

Synopsis

mod_auth

Description

FIXME FIXME FIXME

See also

AccessDenyMsg AccessGrantMsg AnonRejectPasswords AnonRequirePassword AuthAliasOnly AuthGroupFile AuthPAM AuthPAMConfig AuthUserFile AuthUsingAlias CreateHome #CreateHome# DefaultChdir DefaultRoot GroupPassword LoginPasswordPrompt MaxClients MaxClientsPerClass MaxClientsPerHost MaxClientsPerUser MaxConnectionsPerHost MaxHostsPerUser MaxLoginAttempts PersistentPasswd RequireValidShell RootLogin RootRevoke TimeoutLogin TimeoutSession UseFtpUsers UserAlias UserDirRoot UserPassword

CapabilitiesEngineCapabilitiesSet

mod_core

Name

mod_core -- Core module

Synopsis

mod_core

Description

This module provides all the core functionality ProFTPD needs to function, this module must be compiled in.

See also

Allow AllowAll AllowClass AllowFilter AllowForeignAddress AllowGroup AllowOverride AllowRetrieveRestart AllowStoreRestart AllowUser Anonymous AnonymousGroup AuthOrder Bind CDPath Class CommandBufferSize DebugLevel DefaultAddress DefaultServer DefaultTransferMode DeferWelcome Define Deny DenyAll DenyClass DenyFilter DenyGroup DenyUser Directory DisplayChdir DisplayConnect DisplayFirstChdir DisplayGoAway DisplayLogin DisplayQuit Global Group GroupOwner HideFiles HideGroup HideNoAccess HideUser IdentLookups IfDefine IfModule IgnoreHidden Include Limit MasqueradeAddress MaxConnectionRate MaxInstances MultilineRFC2228 Order PassivePorts PathAllowFilter PathDenyFilter PidFile Port RLimitCPU RLimitMemory RLimitOpenFiles ScoreboardFile ServerAdmin ServerIdent ServerName ServerType SetEnv SocketBindTight SocketOptions SyslogFacility SyslogLevel tcpBackLog tcpNoDelay TimeoutIdle TimeoutLinger TimesGMT TransferLog Umask UnsetEnv UseIPv6 User UseReverseDNS UserOwner UseUTF8 VirtualHost WtmpLog

mod_delay

Name

mod_tls -- Prevent information leak through timing attacks

Synopsis

mod_delay

Description

When proftpd processes the USER and PASS FTP commands from a client, it has to perform checks against configured ACLs, look up user and group information, etc. These checks are not done if the given username is known to not exist for the server, in order to not tie up system resources needlessly. However, this does mean that more work is done when handling "good" users than when handling "bad" users. This difference can be detected in the time it takes for proftpd to send a response to the USER and PASS commands. This means it is possible for an attacker to look for these statistical timing differences, and determine which users are "good" and which are "bad". From there, a determined attacker can focus their attention on the known good usernames. Note that the timings will vary depending on server load, number of users in the user base, type of storage of user data (e.g. LDAP directories, SQL tables, RADIUS servers, flat files, etc).

The mod_delay module attempts to prevent such timing differences by keeping track of the time taken to process the USER and PASS commands. It does this for the most recent USER and PASS commands. The timing data are stored in the module's DelayTable. If the module detects that proftpd has not taken enough time to handle one of these commands, compared to its past response times, a small delay will be added to the response cycle. The amount of delay is determined by the difference between the current time spent handling the command and the median time spent handling the same command in the past.

Installation

The mod_delay module is distributed with ProFTPD and compiled in by default.

See also

DelayEngine DelayTable

mod_ldap

Name

mod_ldap -- LDAP authentication support

Synopsis

mod_ldap

Description

mod_ldap provides LDAP authentication support for ProFTPD. It supports many features useful in "toaster" environments such as default UID/GID and autocreation/autogeneration of home directories.

See also

LDAPAliasDereference LDAPAttr LDAPAuthBinds LDAPDefaultAuthScheme LDAPDefaultGID LDAPDefaultUID LDAPDNInfo LDAPDoAuth LDAPDoGIDLookups LDAPDoQuotaLookups LDAPDoUIDLookups LDAPForceDefaultGID LDAPForceDefaultUID LDAPForceGeneratedHomedir LDAPForceHomedirOnDemand LDAPGenerateHomedir LDAPGenerateHomedirPrefix LDAPGenerateHomedirPrefixNoUsername LDAPHomedirOnDemand LDAPHomedirOnDemandPrefix LDAPHomedirOnDemandPrefixNoUsername LDAPHomedirOnDemandSuffix LDAPNegativeCache LDAPProtocolVersion LDAPQueryTimeout LDAPSearchScope LDAPServer LDAPUseTLS

mod_log

Name

mod_log -- Logging support

Synopsis

mod_log

Description

Logging support, including enhanced formatting options.

See also

AllowLogSymlinks ExtendedLog LogFormat ServerLog SystemLog

mod_ls

Name

mod_ls -- file listing functionality

Synopsis

mod_ls

Description

FIXME FIXME FIXME

See also

DirFakeGroup DirFakeMode DirFakeUser ListOptions ShowSymlinks UseGlobbing

mod_radius

Name

mod_radius -- RADIUS based authentication support

Synopsis

mod_radius

Description

This module provides RADIUS authentication and accounting support.

Strong authentication is in demand for Internet services. For many, this means using the RADIUS (Remote Authentication Dial-In User Service) protocol.

However, there are caveats to using RADIUS for authentication. RADIUS packets are sent in the clear, which means that they can easily be sniffed. First, do not have your authenticating RADIUS servers exposed to the Internet; keep them protected within your LAN. Second, it is highly recommended to use separate RADIUS servers for each of your services.

RADIUS Authentication

The RADIUS protocol can be used for answering the question "Should this user be allowed to login?" However, the "yes/no" answer is not everything that proftpd needs to log a user in; the server also requires the UID and GID to use for the authenticated user, home directory, and shell. This information is usually not available from the RADIUS servers, which means that using RADIUS to provide all the necessary login information can be problematic. The RadiusUserInfo directive is meant to be used to address this issue, to provide the missing information.

In those cases where the RADIUS servers can provide that additional login information, via custom attributes, the RadiusUserInfo directive can also be used obtain that information as well.

RADIUS Accounting

While RADIUS is primarily used for authentication, the protocol also allows for accounting of user activities. The mod_radius module makes use of this ability, using RADIUS accounting packets to transmit the following data:

* Acct-Authentic: How the user was authenticated (e.g. locally, or via RADIUS) * Acct-Session-Id: The process ID of the FTP session * Acct-Session-Time: The duration of the FTP session, in seconds * Acct-Input-Octets: The number of bytes uploaded (includes appending to files) * Acct-Output-Octets: The

number of bytes downloaded Merely configuring a RadiusAcctServer enables the module's accounting capabilities. Common Attributes The following RADIUS attributes are sent with every RADIUS packet generated by mod_radius: * User-Name: The name of the logging-in user * NAS-Identifier: Always "ftp" * NAS-IP-Address: IP address of FTP server * NAS-Port: Port of FTP server * NAS-Port-Type: Always Virtual. * Calling-Station-Id: IP address of connecting FTP client

See also

RadiusAcctServer RadiusAuthServer RadiusEngine RadiusLog RadiusRealm RadiusUserInfo

mod_ratio

Name

mod_ratio -- FIX ME FIX ME

Synopsis

mod_ratio

Description

FIXME FIXME FIXME

See also

AnonRatio ByteRatioErrMsg CwdRatioMsg FileRatioErrMsg GroupRatio HostRatio LeechRatioMsg RatioFile Ratios RatioTempFile SaveRatios UserRatio

mod_readme

Name

mod_readme -- "README" file support

Synopsis

mod_readme

Description

FIXME FIXME FIXME

See also

DisplayReadme

RewriteConditionRewriteEngineRewriteLockRewriteLogRewriteMapRewriteRule

mod_sql

Name

mod_sql -- SQL support module

Synopsis

mod_sql

Description

This module provides the necessary support for SQL based authentication, logging and other features as required. It replaces the SQL modules which were shipped with 1.2.0rc2 and earlier.

See also

SQLAuthenticate SQLAuthTypes SQLBackend SQLConnectInfo SQLDefaultGID SQLDefaultHomedir SQLDefaultUID SQLEngine SQLGroupInfo SQLGroupWhereClause SQLHomedirOnDemand SQLLog SQLLogFile SQLMinID SQLMinUserGID SQLMinUserUID SQLNamedQuery SQLNegativeCache SQLRatios SQLRatioStats SQLShowInfo SQLUserInfo SQLUserWhereClause

mod_tls

Name

mod_tls -- TLS/SSL support module

Synopsis

mod_tls

Description

This module provides the necessary support for encrypting you ftp sessions.

Installation

The mod_tls module is distributed with ProFTPD. Simply follow the normal steps for using third-party modules in proftpd: ./configure --with-modules=mod_tls make make install You may need to specify the location of the OpenSSL header and library files in your configure command, e.g.: ./configure --with-modules=mod_tls \ --with-includes=/usr/local/openssl/include \ --with-libraries=/usr/local/openssl

See also

TLSCACertificateFile TLSCACertificatePath TLSCARevocationFile TLSCARevocationPath TLSCertificateChainFile TLSCipherSuite TLSDHParamFile TLSDSACertificateFile TLSDSACertificateKeyFile TLSEngine TLSLog TLSOptions TLSPassPhraseProvider TLSProtocol TLSRandomSeed TLSRenegotiate TLSRequired TLSRSACertificateFile TLSRSACertificateKeyFile TLSVerifyClient TLSVerifyDepth

mod_wrap

Name

mod_wrap -- Interface to libwrap

Synopsis

mod_wrap

Description

It enables the daemon to use the common tcpwrappers access control library while in standalone mode, and in a very configurable manner. It is not compiled by default.

If not installed on your system, the TCP wrappers library, required by this module, can be found here, on Wietse Venema's site. Once installed, it highly recommended that the hosts_access(3) and hosts_access(5) man pages be read and understood.

Many programs will automatically add entries in the common allow/deny files, and use of this module will allow a ProFTPD daemon running in standalone mode to adapt as these entries are added. The portsentry program does this, for example: when illegal access is attempted, it will add hosts to the /etc/hosts.deny file.

See also

TCPAccessFiles TCPAccessSyslogLevels TCPGroupAccessFiles TCPServiceName TCPUserAccessFiles

mod_xfer

Name

mod_xfer -- FIX ME FIX ME

Synopsis

mod_xfer

Description

FIXME FIXME FIXME

See also

AllowOverwrite DeleteAbortedStores DisplayFileTransfer HiddenStor HiddenStores MaxRetrieveFileSize MaxStoreFileSize StoreUniquePrefix TimeoutNoTransfer TimeoutStalled TransferRate UseSendfile

Chapter 3. List of configuration contexts

server config

Name

server config -- server config

Synopsis

server config

Description

FIXME FIXME FIXME

See also

Global

Name

Global -- Global

Synopsis

Global

Description

FIXME FIXME FIXME

See also

VirtualHost

Name

VirtualHost -- VirtualHost

Synopsis

VirtualHost

Description

FIXME FIXME FIXME

See also

Anonymous

Name

Anonymous -- Anonymous

Synopsis

Anonymous

Description

FIXME FIXME FIXME

See also

AccessDenyMsg AccessGrantMsg AllowAll AllowFilter AllowForeignAddress AllowOverride AllowOverwrite AllowRetrieveRestart AllowStoreRestart AnonRatio AnonRejectPasswords AnonRequirePassword AuthAliasOnly AuthUsingAlias ByteRatioErrMsg CDPath CwdRatioMsg DefaultChdir DeleteAbortedStores DenyAll DenyFilter Directory DirFakeGroup DirFakeMode DirFakeUser DisplayChdir DisplayFileTransfer DisplayFirstChdir DisplayGoAway DisplayLogin DisplayQuit DisplayReadme ExtendedLog FileRatioErrMsg Group GroupOwner GroupPassword GroupRatio HiddenStor HiddenStores HideGroup HideNoAccess HideUser HostRatio Include LDAPGenerateHomedirPrefixNoUsername LDAPHomedirOnDemandPrefixNoUsername LeechRatioMsg Limit ListOptions LoginPasswordPrompt MaxClients MaxClientsPerHost MaxClientsPerUser MaxConnectionsPerHost MaxHostsPerUser MaxRetrieveFileSize MaxStoreFileSize PathAllowFilter PathDenyFilter RatioFile Ratios RatioTempFile RequireValidShell RewriteCondition RewriteRule RootLogin RootRevoke SaveRatios ShowSymlinks SQLEngine SQLNamedQuery StoreUniquePrefix TCPAccessFiles TCPAccessSyslogLevels TimeoutSession TimesGMT TLSPassPhraseProvider TransferLog TransferRate Umask UseFtpUsers UseGlobbing User UserAlias UserDirRoot UserOwner UserPassword UserRatio UseUTF8 WtmpLog

Limit

Name

Limit -- Limit

Synopsis

Limit

Description

FIXME FIXME FIXME

See also

Allow AllowAll AllowClass AllowGroup AllowUser AnonRatio ByteRatioErrMsg CwdRatioMsg Deny DenyAll DenyClass DenyGroup DenyUser DisplayFileTransfer FileRatioErrMsg GroupRatio HostRatio IgnoreHidden LDAPGenerateHomedirPrefixNoUsername LDAPHomedirOnDemandPrefixNoUsername LeechRatioMsg Order RatioFile Ratios RatioTempFile SaveRatios SQLNamedQuery TLSPassPhraseProvider UserRatio UseUTF8

.ftpaccess

Name

.ftpaccess -- .ftpaccess

Synopsis

.ftpaccess

Description

FIXME FIXME FIXME

See also

AllowAll AllowFilter AllowOverwrite AllowRetrieveRestart AllowStoreRestart AnonRatio ByteRatioErrMsg CwdRatioMsg DeleteAbortedStores DenyAll DenyFilter DirFakeGroup DirFakeUser DisplayFileTransfer FileRatioErrMsg GroupOwner GroupRatio HideFiles HostRatio LDAPGenerateHomedirPrefixNoUsername LDAPHomedirOnDemandPrefixNoUsername LeechRatioMsg Limit ListOptions MaxRetrieveFileSize MaxStoreFileSize PathAllowFilter PathDenyFilter RatioFile Ratios RatioTempFile SaveRatios SQLNamedQuery StoreUniquePrefix TLSPassPhraseProvider TransferRate Umask UserRatio UseUTF8

Notes

[1]