Red Hat Enterprise Linux Release 9.2 Manual Pages on 'sshd_config.5' command

$ man sshd_config.5

SSHD_CONFIG(5)   BSD File Formats Manual   SSHD_CONFIG(5)

NAME

   sshd_config ? OpenSSH daemon configuration file

DESCRIPTION

   sshd(8) reads configuration data from /etc/ssh/sshd_config (or the file

   specified with -f on the command line). The file contains keyword-argu?

   ment pairs, one per line. For each keyword, the first obtained value

   will be used. Lines starting with ?#? and empty lines are interpreted as

   comments. Arguments may optionally be enclosed in double quotes (") in

   order to represent arguments containing spaces.

   The possible keywords and their meanings are as follows (note that key?

   words are case-insensitive and arguments are case-sensitive):

   AcceptEnv

        Specifies what environment variables sent by the client will be

        copied into the session's environ(7). See SendEnv and SetEnv in

        ssh_config(5) for how to configure the client. The TERM environ?

        ment variable is always accepted whenever the client requests a

        pseudo-terminal as it is required by the protocol. Variables are

        specified by name, which may contain the wildcard characters ?*?

        and ???. Multiple environment variables may be separated by

        whitespace or spread across multiple AcceptEnv directives. Be

        warned that some environment variables could be used to bypass

        restricted user environments. For this reason, care should be

taken in the use of this directive. The default is not to accept

any environment variables.

AddressFamily

Specifies which address family should be used by sshd(8). Valid

arguments are any (the default), inet (use IPv4 only), or inet6

(use IPv6 only).

AllowAgentForwarding

Specifies whether ssh-agent(1) forwarding is permitted. The de?

fault is yes. Note that disabling agent forwarding does not im?

prove security unless users are also denied shell access, as they

can always install their own forwarders.

AllowGroups

This keyword can be followed by a list of group name patterns,

separated by spaces. If specified, login is allowed only for

users whose primary group or supplementary group list matches one

of the patterns. Only group names are valid; a numerical group

ID is not recognized. By default, login is allowed for all

groups. The allow/deny groups directives are processed in the

following order: DenyGroups, AllowGroups.

See PATTERNS in ssh_config(5) for more information on patterns.

AllowStreamLocalForwarding

Specifies whether StreamLocal (Unix-domain socket) forwarding is

permitted. The available options are yes (the default) or all to

allow StreamLocal forwarding, no to prevent all StreamLocal for?

warding, local to allow local (from the perspective of ssh(1))

forwarding only or remote to allow remote forwarding only. Note

that disabling StreamLocal forwarding does not improve security

unless users are also denied shell access, as they can always in?

stall their own forwarders.

AllowTcpForwarding

Specifies whether TCP forwarding is permitted. The available op?

tions are yes (the default) or all to allow TCP forwarding, no to

prevent all TCP forwarding, local to allow local (from the per?

spective of ssh(1)) forwarding only or remote to allow remote
forwarding only. Note that disabling TCP forwarding does not im?
prove security unless users are also denied shell access, as they
can always install their own forwarders.

AllowUsers

This keyword can be followed by a list of user name patterns,
separated by spaces. If specified, login is allowed only for
user names that match one of the patterns. Only user names are
valid; a numerical user ID is not recognized. By default, login
is allowed for all users. If the pattern takes the form
USER@HOST then USER and HOST are separately checked, restricting
logins to particular users from particular hosts. HOST criteria
may additionally contain addresses to match in CIDR ad?
dress/masklen format. The allow/deny users directives are pro?
cessed in the following order: DenyUsers, AllowUsers.

See PATTERNS in ssh_config(5) for more information on patterns.

AuthenticationMethods

Specifies the authentication methods that must be successfully
completed for a user to be granted access. This option must be
followed by one or more lists of comma-separated authentication
method names, or by the single string any to indicate the default
behaviour of accepting any single authentication method. If the
default is overridden, then successful authentication requires
completion of every method in at least one of these lists.

For example, "publickey,password publickey,keyboard-interactive"
would require the user to complete public key authentication,
followed by either password or keyboard interactive authentica?
tion. Only methods that are next in one or more lists are of?
fered at each stage, so for this example it would not be possible
to attempt password or keyboard-interactive authentication before
public key.

For keyboard interactive authentication it is also possible to
restrict authentication to a specific device by appending a colon

followed by the device identifier bsdauth or pam. depending on

the server configuration. For example,

"keyboard-interactive:bsdauth" would restrict keyboard interac?

tive authentication to the bsdauth device.

If the publickey method is listed more than once, sshd(8) veri?

fies that keys that have been used successfully are not reused

for subsequent authentications. For example,

"publickey,publickey" requires successful authentication using

two different public keys.

Note that each authentication method listed should also be ex?

plicitly enabled in the configuration.

The available authentication methods are: "gssapi-with-mic",

"hostbased", "keyboard-interactive", "none" (used for access to

password-less accounts when PermitEmptyPasswords is enabled),

"password" and "publickey".

AuthorizedKeysCommand

Specifies a program to be used to look up the user's public keys.

The program must be owned by root, not writable by group or oth?

ers and specified by an absolute path. Arguments to

AuthorizedKeysCommand accept the tokens described in the TOKENS

section. If no arguments are specified then the username of the

target user is used.

The program should produce on standard output zero or more lines

of authorized_keys output (see AUTHORIZED_KEYS in sshd(8)).

AuthorizedKeysCommand is tried after the usual AuthorizedKeysFile

files and will not be executed if a matching key is found there.

By default, no AuthorizedKeysCommand is run.

AuthorizedKeysCommandUser

Specifies the user under whose account the AuthorizedKeysCommand

is run. It is recommended to use a dedicated user that has no

other role on the host than running authorized keys commands. If

AuthorizedKeysCommand is specified but AuthorizedKeysCommandUser

is not, then sshd(8) will refuse to start.

AuthorizedKeysFile

> Specifies the file that contains the public keys used for user
>
> authentication. The format is described in the AUTHORIZED_KEYS
>
> FILE FORMAT section of sshd(8). Arguments to AuthorizedKeysFile
>
> accept the tokens described in the TOKENS section. After expan?
>
> sion, AuthorizedKeysFile is taken to be an absolute path or one
>
> relative to the user's home directory. Multiple files may be
>
> listed, separated by whitespace. Alternately this option may be
>
> set to none to skip checking for user keys in files. The default
>
> is ".ssh/authorized_keys .ssh/authorized_keys2".

AuthorizedPrincipalsCommand

> Specifies a program to be used to generate the list of allowed
>
> certificate principals as per AuthorizedPrincipalsFile. The pro?
>
> gram must be owned by root, not writable by group or others and
>
> specified by an absolute path. Arguments to
>
> AuthorizedPrincipalsCommand accept the tokens described in the
>
> TOKENS section. If no arguments are specified then the username
>
> of the target user is used.
>
> The program should produce on standard output zero or more lines
>
> of AuthorizedPrincipalsFile output. If either
>
> AuthorizedPrincipalsCommand or AuthorizedPrincipalsFile is speci?
>
> fied, then certificates offered by the client for authentication
>
> must contain a principal that is listed. By default, no
>
> AuthorizedPrincipalsCommand is run.

AuthorizedPrincipalsCommandUser

> Specifies the user under whose account the
>
> AuthorizedPrincipalsCommand is run. It is recommended to use a
>
> dedicated user that has no other role on the host than running
>
> authorized principals commands. If AuthorizedPrincipalsCommand
>
> is specified but AuthorizedPrincipalsCommandUser is not, then
>
> sshd(8) will refuse to start.

AuthorizedPrincipalsFile

> Specifies a file that lists principal names that are accepted for

certificate authentication. When using certificates signed by a

key listed in TrustedUserCAKeys, this file lists names, one of

which must appear in the certificate for it to be accepted for

authentication. Names are listed one per line preceded by key

options (as described in AUTHORIZED_KEYS FILE FORMAT in sshd(8)).

Empty lines and comments starting with ?#? are ignored.

Arguments to AuthorizedPrincipalsFile accept the tokens described

in the TOKENS section. After expansion, AuthorizedPrincipalsFile

is taken to be an absolute path or one relative to the user's

home directory. The default is none, i.e. not to use a princi?

pals file ? in this case, the username of the user must appear in

a certificate's principals list for it to be accepted.

Note that AuthorizedPrincipalsFile is only used when authentica?

tion proceeds using a CA listed in TrustedUserCAKeys and is not

consulted for certification authorities trusted via

~/.ssh/authorized_keys, though the principals= key option offers

a similar facility (see sshd(8) for details).

Banner The contents of the specified file are sent to the remote user

before authentication is allowed. If the argument is none then

no banner is displayed. By default, no banner is displayed.

CASignatureAlgorithms

The default is handled system-wide by crypto-policies(7). Infor?

mation about defaults, how to modify the defaults and how to cus?

tomize existing policies with sub-policies are present in manual

page update-crypto-policies(8).

Specifies which algorithms are allowed for signing of certifi?

cates by certificate authorities (CAs). If the specified list

begins with a ?+? character, then the specified algorithms will

be appended to the default set instead of replacing them. If the

specified list begins with a ?-? character, then the specified

algorithms (including wildcards) will be removed from the default

set instead of replacing them.

Certificates signed using other algorithms will not be accepted

for public key or host-based authentication.

ChrootDirectory

Specifies the pathname of a directory to chroot(2) to after au‐

thentication. At session startup sshd(8) checks that all compo‐

nents of the pathname are root-owned directories which are not

writable by any other user or group. After the chroot, sshd(8)

changes the working directory to the user's home directory. Ar‐

guments to ChrootDirectory accept the tokens described in the

TOKENS section.

The ChrootDirectory must contain the necessary files and directo‐

ries to support the user's session. For an interactive session

this requires at least a shell, typically sh(1), and basic /dev

nodes such as null(4), zero(4), stdin(4), stdout(4), stderr(4),

and tty(4) devices. For file transfer sessions using SFTP no ad‐

ditional configuration of the environment is necessary if the in-

process sftp-server is used, though sessions which use logging

may require /dev/log inside the chroot directory on some operat‐

ing systems (see sftp-server(8) for details).

For safety, it is very important that the directory hierarchy be

prevented from modification by other processes on the system (es‐

pecially those outside the jail). Misconfiguration can lead to

unsafe environments which sshd(8) cannot detect.

The default is none, indicating not to chroot(2).

Ciphers

The default is handled system-wide by crypto-policies(7). Infor‐

mation about defaults, how to modify the defaults and how to cus‐

tomize existing policies with sub-policies are present in manual

page update-crypto-policies(8).

Specifies the ciphers allowed. Multiple ciphers must be comma-

separated. If the specified list begins with a ?+? character,

then the specified ciphers will be appended to the built-in

openssh default set instead of replacing them. If the specified

list begins with a ?-? character, then the specified ciphers (in‐

cluding wildcards) will be removed from the built-in openssh de?

fault set instead of replacing them. If the specified list be?

gins with a ?^? character, then the specified ciphers will be

placed at the head of the built-in openssh default set.

The supported ciphers are:

    3des-cbc

    aes128-cbc

    aes192-cbc

    aes256-cbc

    aes128-ctr

    aes192-ctr

    aes256-ctr

    aes128-gcm@openssh.com

    aes256-gcm@openssh.com

    chacha20-poly1305@openssh.com

The list of available ciphers may also be obtained using "ssh -Q

cipher".

ClientAliveCountMax

Sets the number of client alive messages which may be sent with?

out sshd(8) receiving any messages back from the client. If this

threshold is reached while client alive messages are being sent,

sshd will disconnect the client, terminating the session. It is

important to note that the use of client alive messages is very

different from TCPKeepAlive. The client alive messages are sent

through the encrypted channel and therefore will not be spoofa?

ble. The TCP keepalive option enabled by TCPKeepAlive is spoofa?

ble. The client alive mechanism is valuable when the client or

server depend on knowing when a connection has become unrespon?

sive.

The default value is 3. If ClientAliveInterval is set to 15, and

ClientAliveCountMax is left at the default, unresponsive SSH

clients will be disconnected after approximately 45 seconds.

Setting a zero ClientAliveCountMax disables connection termina?

tion.

ClientAliveInterval

Sets a timeout interval in seconds after which if no data has

been received from the client, sshd(8) will send a message

through the encrypted channel to request a response from the

client. The default is 0, indicating that these messages will

not be sent to the client.

Compression

Specifies whether compression is enabled after the user has au?

thenticated successfully. The argument must be yes, delayed (a

legacy synonym for yes) or no. The default is yes.

DenyGroups

This keyword can be followed by a list of group name patterns,

separated by spaces. Login is disallowed for users whose primary

group or supplementary group list matches one of the patterns.

Only group names are valid; a numerical group ID is not recog?

nized. By default, login is allowed for all groups. The al?

low/deny groups directives are processed in the following order:

DenyGroups, AllowGroups.

See PATTERNS in ssh_config(5) for more information on patterns.

DenyUsers

This keyword can be followed by a list of user name patterns,

separated by spaces. Login is disallowed for user names that

match one of the patterns. Only user names are valid; a numeri?

cal user ID is not recognized. By default, login is allowed for

all users. If the pattern takes the form USER@HOST then USER and

HOST are separately checked, restricting logins to particular

users from particular hosts. HOST criteria may additionally con?

tain addresses to match in CIDR address/masklen format. The al?

low/deny users directives are processed in the following order:

DenyUsers, AllowUsers.

See PATTERNS in ssh_config(5) for more information on patterns.

DisableForwarding

Disables all forwarding features, including X11, ssh-agent(1),
TCP and StreamLocal. This option overrides all other forwarding-
related options and may simplify restricted configurations.

ExposeAuthInfo

Writes a temporary file containing a list of authentication meth?
ods and public credentials (e.g. keys) used to authenticate the
user. The location of the file is exposed to the user session
through the SSH_USER_AUTH environment variable. The default is
no.

FingerprintHash

Specifies the hash algorithm used when logging key fingerprints.
Valid options are: md5 and sha256. The default is sha256.

ForceCommand

Forces the execution of the command specified by ForceCommand,
ignoring any command supplied by the client and ~/.ssh/rc if
present. The command is invoked by using the user's login shell
with the -c option. This applies to shell, command, or subsystem
execution. It is most useful inside a Match block. The command
originally supplied by the client is available in the
SSH_ORIGINAL_COMMAND environment variable. Specifying a command
of internal-sftp will force the use of an in-process SFTP server
that requires no support files when used with ChrootDirectory.
The default is none.

GatewayPorts

Specifies whether remote hosts are allowed to connect to ports
forwarded for the client. By default, sshd(8) binds remote port
forwardings to the loopback address. This prevents other remote
hosts from connecting to forwarded ports. GatewayPorts can be
used to specify that sshd should allow remote port forwardings to
bind to non-loopback addresses, thus allowing other hosts to con?
nect. The argument may be no to force remote port forwardings to
be available to the local host only, yes to force remote port
forwardings to bind to the wildcard address, or clientspecified

to allow the client to select the address to which the forwarding

is bound. The default is no.

GSSAPIAuthentication

Specifies whether user authentication based on GSSAPI is allowed.

The default is no.

GSSAPICleanupCredentials

Specifies whether to automatically destroy the user's credentials

cache on logout. The default is yes.

GSSAPIEnablek5users

Specifies whether to look at .k5users file for GSSAPI authentica?

tion access control. Further details are described in ksu(1).

The default is no.

GSSAPIKeyExchange

Specifies whether key exchange based on GSSAPI is allowed. GSSAPI

key exchange doesn't rely on ssh keys to verify host identity.

The default is no.

GSSAPIStrictAcceptorCheck

Determines whether to be strict about the identity of the GSSAPI

acceptor a client authenticates against. If set to yes then the

client must authenticate against the host service on the current

hostname. If set to no then the client may authenticate against

any service key stored in the machine's default store. This fa?

cility is provided to assist with operation on multi homed ma?

chines. The default is yes.

GSSAPIStoreCredentialsOnRekey

Controls whether the user's GSSAPI credentials should be updated

following a successful connection rekeying. This option can be

used to accepted renewed or updated credentials from a compatible

client. The default is ?no?.

For this to work GSSAPIKeyExchange needs to be enabled in the

server and also used by the client.

GSSAPIKexAlgorithms

The default is handled system-wide by crypto-policies(7). Infor?

mation about defaults, how to modify the defaults and how to cus?

tomize existing policies with sub-policies are present in manual

page update-crypto-policies(8).

The list of key exchange algorithms that are accepted by GSSAPI

key exchange. Possible values are

gss-gex-sha1-

gss-group1-sha1-

gss-group14-sha1-

gss-group14-sha256-

gss-group16-sha512-

gss-nistp256-sha256-

gss-curve25519-sha256-

This option only applies to connections using GSSAPI.

HostbasedAcceptedAlgorithms

Specifies the signature algorithms that will be accepted for

hostbased authentication as a list of comma-separated patterns.

Alternately if the specified list begins with a ?+? character,

then the specified signature algorithms will be appended to the

default set instead of replacing them. If the specified list be?

gins with a ?-? character, then the specified signature algo?

rithms (including wildcards) will be removed from the default set

instead of replacing them. If the specified list begins with a

?^? character, then the specified signature algorithms will be

placed at the head of the default set. The default for this op?

tion is:

ssh-ed25519-cert-v01@openssh.com,

ecdsa-sha2-nistp256-cert-v01@openssh.com,

ecdsa-sha2-nistp384-cert-v01@openssh.com,

ecdsa-sha2-nistp521-cert-v01@openssh.com,

sk-ssh-ed25519-cert-v01@openssh.com,

sk-ecdsa-sha2-nistp256-cert-v01@openssh.com,

rsa-sha2-512-cert-v01@openssh.com,

rsa-sha2-256-cert-v01@openssh.com,

ssh-rsa-cert-v01@openssh.com,

ssh-ed25519,

ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,

sk-ssh-ed25519@openssh.com,

sk-ecdsa-sha2-nistp256@openssh.com,

rsa-sha2-512,rsa-sha2-256,ssh-rsa

The list of available signature algorithms may also be obtained

using "ssh -Q HostbasedAcceptedAlgorithms". This was formerly

named HostbasedAcceptedKeyTypes.

HostbasedAuthentication

Specifies whether rhosts or /etc/hosts.equiv authentication to?

gether with successful public key client host authentication is

allowed (host-based authentication). The default is no.

HostbasedUsesNameFromPacketOnly

Specifies whether or not the server will attempt to perform a re?

verse name lookup when matching the name in the ~/.shosts,

~/.rhosts, and /etc/hosts.equiv files during

HostbasedAuthentication. A setting of yes means that sshd(8)

uses the name supplied by the client rather than attempting to

resolve the name from the TCP connection itself. The default is

no.

HostCertificate

Specifies a file containing a public host certificate. The cer?

tificate's public key must match a private host key already spec?

ified by HostKey. The default behaviour of sshd(8) is not to

load any certificates.

HostKey

Specifies a file containing a private host key used by SSH. The

defaults are /etc/ssh/ssh_host_ecdsa_key,

/etc/ssh/ssh_host_ed25519_key and /etc/ssh/ssh_host_rsa_key.

Note that sshd(8) will refuse to use a file if it is group/world-

accessible and that the HostKeyAlgorithms option restricts which

of the keys are actually used by sshd(8).

It is possible to have multiple host key files. It is also pos?

sible to specify public host key files instead. In this case op?

erations on the private key will be delegated to an ssh-agent(1).

HostKeyAgent

Identifies the UNIX-domain socket used to communicate with an

agent that has access to the private host keys. If the string

"SSH_AUTH_SOCK" is specified, the location of the socket will be

read from the SSH_AUTH_SOCK environment variable.

HostKeyAlgorithms

The default is handled system-wide by crypto-policies(7). Infor?

mation about defaults, how to modify the defaults and how to cus?

tomize existing policies with sub-policies are present in manual

page update-crypto-policies(8).

Specifies the host key signature algorithms that the server of?

fers. The list of available signature algorithms may also be ob?

tained using "ssh -Q HostKeyAlgorithms".

IgnoreRhosts

Specifies whether to ignore per-user .rhosts and .shosts files

during HostbasedAuthentication. The system-wide /etc/hosts.equiv

and /etc/ssh/shosts.equiv are still used regardless of this set?

ting.

Accepted values are yes (the default) to ignore all per-user

files, shosts-only to allow the use of .shosts but to ignore

.rhosts or no to allow both .shosts and rhosts.

IgnoreUserKnownHosts

Specifies whether sshd(8) should ignore the user's

~/.ssh/known_hosts during HostbasedAuthentication and use only

the system-wide known hosts file /etc/ssh/known_hosts. The de?

fault is ?no?.

Include

Include the specified configuration file(s). Multiple pathnames

may be specified and each pathname may contain glob(7) wildcards

that will be expanded and processed in lexical order. Files

without absolute paths are assumed to be in /etc/ssh. An Include

directive may appear inside a Match block to perform conditional

inclusion.

IPQoS   Specifies the IPv4 type-of-service or DSCP class for the connec?

tion. Accepted values are af11, af12, af13, af21, af22, af23,

af31, af32, af33, af41, af42, af43, cs0, cs1, cs2, cs3, cs4, cs5,

cs6, cs7, ef, le, lowdelay, throughput, reliability, a numeric

value, or none to use the operating system default. This option

may take one or two arguments, separated by whitespace. If one

argument is specified, it is used as the packet class uncondi?

tionally. If two values are specified, the first is automati?

cally selected for interactive sessions and the second for non-

interactive sessions. The default is af21 (Low-Latency Data) for

interactive sessions and cs1 (Lower Effort) for non-interactive

sessions.

KbdInteractiveAuthentication

Specifies whether to allow keyboard-interactive authentication.

All authentication styles from login.conf(5) are supported. The

default is yes. The argument to this keyword must be yes or no.

ChallengeResponseAuthentication is a deprecated alias for this.

KerberosAuthentication

Specifies whether the password provided by the user for

PasswordAuthentication will be validated through the Kerberos

KDC. To use this option, the server needs a Kerberos servtab

which allows the verification of the KDC's identity. The default

is no.

KerberosGetAFSToken

If AFS is active and the user has a Kerberos 5 TGT, attempt to

acquire an AFS token before accessing the user's home directory.

The default is no.

KerberosOrLocalPasswd

If password authentication through Kerberos fails then the pass?

word will be validated via any additional local mechanism such as

/etc/passwd. The default is yes.

KerberosTicketCleanup

Specifies whether to automatically destroy the user's ticket

cache file on logout. The default is yes.

KerberosUniqueCCache

Specifies whether to store the acquired tickets in the per-ses?

sion credential cache under /tmp/ or whether to use per-user cre?

dential cache as configured in /etc/krb5.conf. The default value

no can lead to overwriting previous tickets by subseqent connec?

tions to the same user account.

KerberosUseKuserok

Specifies whether to look at .k5login file for user's aliases.

The default is yes.

KexAlgorithms

The default is handled system-wide by crypto-policies(7). Infor?

mation about defaults, how to modify the defaults and how to cus?

tomize existing policies with sub-policies are present in manual

page update-crypto-policies(8).

Specifies the available KEX (Key Exchange) algorithms. Multiple

algorithms must be comma-separated. Alternately if the specified

list begins with a ?+? character, then the specified methods will

be appended to the built-in openssh default set instead of re?

placing them. If the specified list begins with a ?-? character,

then the specified methods (including wildcards) will be removed

from the built-in openssh default set instead of replacing them.

If the specified list begins with a ?^? character, then the spec?

ified methods will be placed at the head of the built-in openssh

default set. The supported algorithms are:

curve25519-sha256

curve25519-sha256@libssh.org

diffie-hellman-group1-sha1

diffie-hellman-group14-sha1

diffie-hellman-group14-sha256

diffie-hellman-group16-sha512

diffie-hellman-group18-sha512

diffie-hellman-group-exchange-sha1

diffie-hellman-group-exchange-sha256

ecdh-sha2-nistp256

ecdh-sha2-nistp384

ecdh-sha2-nistp521

sntrup761x25519-sha512@openssh.com

The list of available key exchange algorithms may also be ob?

tained using "ssh -Q KexAlgorithms".

ListenAddress

Specifies the local addresses sshd(8) should listen on. The fol?

lowing forms may be used:

ListenAddress hostname|address [rdomain domain]

ListenAddress hostname:port [rdomain domain]

ListenAddress IPv4_address:port [rdomain domain]

ListenAddress [hostname|address]:port [rdomain domain]

The optional rdomain qualifier requests sshd(8) listen in an ex?

plicit routing domain. If port is not specified, sshd will lis?

ten on the address and all Port options specified. The default

is to listen on all local addresses on the current default rout?

ing domain. Multiple ListenAddress options are permitted. For

more information on routing domains, see rdomain(4).

LoginGraceTime

The server disconnects after this time if the user has not suc?

cessfully logged in. If the value is 0, there is no time limit.

The default is 120 seconds.

LogLevel

Gives the verbosity level that is used when logging messages from

sshd(8). The possible values are: QUIET, FATAL, ERROR, INFO,

VERBOSE, DEBUG, DEBUG1, DEBUG2, and DEBUG3. The default is INFO.

DEBUG and DEBUG1 are equivalent. DEBUG2 and DEBUG3 each specify

higher levels of debugging output. Logging with a DEBUG level

violates the privacy of users and is not recommended.

LogVerbose

Specify one or more overrides to LogLevel. An override consists

of a pattern lists that matches the source file, function and

line number to force detailed logging for. For example, an over?

ride pattern of:

kex.c:*:1000,*:kex_exchange_identification():*,packet.c:*

would enable detailed logging for line 1000 of kex.c, everything

in the kex_exchange_identification() function, and all code in

the packet.c file. This option is intended for debugging and no

overrides are enabled by default.

MACs   The default is handled system-wide by crypto-policies(7). Infor?

mation about defaults, how to modify the defaults and how to cus?

tomize existing policies with sub-policies are present in manual

page update-crypto-policies(8).

Specifies the available MAC (message authentication code) algo?

rithms. The MAC algorithm is used for data integrity protection.

Multiple algorithms must be comma-separated. If the specified

list begins with a ?+? character, then the specified algorithms

will be appended to the built-in openssh default set instead of

replacing them. If the specified list begins with a ?-? charac?

ter, then the specified algorithms (including wildcards) will be

removed from the built-in openssh default set instead of replac?

ing them. If the specified list begins with a ?^? character,

then the specified algorithms will be placed at the head of the

built-in openssh default set.

The algorithms that contain "-etm" calculate the MAC after en?

cryption (encrypt-then-mac). These are considered safer and

their use recommended. The supported MACs are:

hmac-md5

hmac-md5-96

hmac-sha1

hmac-sha1-96

hmac-sha2-256

hmac-sha2-512

umac-64@openssh.com

umac-128@openssh.com

hmac-md5-etm@openssh.com

hmac-md5-96-etm@openssh.com

hmac-sha1-etm@openssh.com

hmac-sha1-96-etm@openssh.com

hmac-sha2-256-etm@openssh.com

hmac-sha2-512-etm@openssh.com

umac-64-etm@openssh.com

umac-128-etm@openssh.com

The list of available MAC algorithms may also be obtained using

"ssh -Q mac".

Match   Introduces a conditional block. If all of the criteria on the

Match line are satisfied, the keywords on the following lines

override those set in the global section of the config file, un?

til either another Match line or the end of the file. If a key?

word appears in multiple Match blocks that are satisfied, only

the first instance of the keyword is applied.

The arguments to Match are one or more criteria-pattern pairs or

the single token All which matches all criteria. The available

criteria are User, Group, Host, LocalAddress, LocalPort, RDomain,

and Address (with RDomain representing the rdomain(4) on which

the connection was received).

The match patterns may consist of single entries or comma-sepa?

rated lists and may use the wildcard and negation operators de?

scribed in the PATTERNS section of ssh_config(5).

The patterns in an Address criteria may additionally contain ad?

dresses to match in CIDR address/masklen format, such as

192.0.2.0/24 or 2001:db8::/32. Note that the mask length pro?

vided must be consistent with the address - it is an error to

specify a mask length that is too long for the address or one

with bits set in this host portion of the address. For example,

192.0.2.0/33 and 192.0.2.0/8, respectively.

Only a subset of keywords may be used on the lines following a

Match keyword. Available keywords are AcceptEnv,

AllowAgentForwarding, AllowGroups, AllowStreamLocalForwarding,

AllowTcpForwarding, AllowUsers, AuthenticationMethods,

AuthorizedKeysCommand, AuthorizedKeysCommandUser,

AuthorizedKeysFile, AuthorizedPrincipalsCommand,

AuthorizedPrincipalsCommandUser, AuthorizedPrincipalsFile,

Banner, ChrootDirectory, ClientAliveCountMax,

ClientAliveInterval, DenyGroups, DenyUsers, DisableForwarding,

ForceCommand, GatewayPorts, GSSAPIAuthentication,

HostbasedAcceptedAlgorithms, HostbasedAuthentication,

HostbasedUsesNameFromPacketOnly, IgnoreRhosts, Include, IPQoS,

KbdInteractiveAuthentication, KerberosAuthentication,

KerberosUseKuserok, LogLevel, MaxAuthTries, MaxSessions,

PasswordAuthentication, PermitEmptyPasswords, PermitListen,

PermitOpen, PermitRootLogin, PermitTTY, PermitTunnel,

PermitUserRC, PubkeyAcceptedAlgorithms, PubkeyAuthentication,

RekeyLimit, RevokedKeys, RDomain, SetEnv, StreamLocalBindMask,

StreamLocalBindUnlink, TrustedUserCAKeys, X11DisplayOffset,

X11MaxDisplays, X11Forwarding and X11UseLocalhost.

MaxAuthTries

Specifies the maximum number of authentication attempts permitted

per connection. Once the number of failures reaches half this

value, additional failures are logged. The default is 6.

MaxSessions

Specifies the maximum number of open shell, login or subsystem

(e.g. sftp) sessions permitted per network connection. Multiple

sessions may be established by clients that support connection

multiplexing. Setting MaxSessions to 1 will effectively disable

session multiplexing, whereas setting it to 0 will prevent all

shell, login and subsystem sessions while still permitting for?

warding. The default is 10.

MaxStartups

Specifies the maximum number of concurrent unauthenticated con‐

nections to the SSH daemon. Additional connections will be

dropped until authentication succeeds or the LoginGraceTime ex‐

pires for a connection. The default is 10:30:100.

Alternatively, random early drop can be enabled by specifying the

three colon separated values start:rate:full (e.g. "10:30:60").

sshd(8) will refuse connection attempts with a probability of

rate/100 (30%) if there are currently start (10) unauthenticated

connections. The probability increases linearly and all connec‐

tion attempts are refused if the number of unauthenticated con‐

nections reaches full (60).

ModuliFile

Specifies the moduli(5) file that contains the Diffie-Hellman

groups used for the ‐diffie-hellman-group-exchange-sha1‐ and

‐diffie-hellman-group-exchange-sha256‐ key exchange methods. The

default is /etc/ssh/moduli.

PasswordAuthentication

Specifies whether password authentication is allowed. The de‐

fault is yes.

PermitEmptyPasswords

When password authentication is allowed, it specifies whether the

server allows login to accounts with empty password strings. The

default is no.

PermitListen

Specifies the addresses/ports on which a remote TCP port forward‐

ing may listen. The listen specification must be one of the fol‐

lowing forms:

PermitListen port

PermitListen host:port

Multiple permissions may be specified by separating them with

whitespace. An argument of any can be used to remove all re‐

strictions and permit any listen requests. An argument of none
can be used to prohibit all listen requests. The host name may
contain wildcards as described in the PATTERNS section in
ssh_config(5). The wildcard ?*? can also be used in place of a
port number to allow all ports. By default all port forwarding
listen requests are permitted. Note that the GatewayPorts option
may further restrict which addresses may be listened on. Note
also that ssh(1) will request a listen host of ?localhost? if no
listen host was specifically requested, and this name is treated
differently to explicit localhost addresses of ?127.0.0.1? and
?::1?.

PermitOpen

Specifies the destinations to which TCP port forwarding is per?
mitted. The forwarding specification must be one of the follow?
ing forms:

PermitOpen host:port

PermitOpen IPv4_addr:port

PermitOpen [IPv6_addr]:port

Multiple forwards may be specified by separating them with white?
space. An argument of any can be used to remove all restrictions
and permit any forwarding requests. An argument of none can be
used to prohibit all forwarding requests. The wildcard ?*? can
be used for host or port to allow all hosts or ports respec?
tively. Otherwise, no pattern matching or address lookups are
performed on supplied names. By default all port forwarding re?
quests are permitted.

PermitRootLogin

Specifies whether root can log in using ssh(1). The argument
must be yes, prohibit-password, forced-commands-only, or no. The
default is prohibit-password.

If this option is set to prohibit-password (or its deprecated
alias, without-password), password and keyboard-interactive au?
thentication are disabled for root.

If this option is set to forced-commands-only, root login with

public key authentication will be allowed, but only if the

command option has been specified (which may be useful for taking

remote backups even if root login is normally not allowed). All

other authentication methods are disabled for root.

If this option is set to no, root is not allowed to log in.

PermitTTY

Specifies whether pty(4) allocation is permitted. The default is

yes.

PermitTunnel

Specifies whether tun(4) device forwarding is allowed. The argu?

ment must be yes, point-to-point (layer 3), ethernet (layer 2),

or no. Specifying yes permits both point-to-point and ethernet.

The default is no.

Independent of this setting, the permissions of the selected

tun(4) device must allow access to the user.

PermitUserEnvironment

Specifies whether ~/.ssh/environment and environment= options in

~/.ssh/authorized_keys are processed by sshd(8). Valid options

are yes, no or a pattern-list specifying which environment vari?

able names to accept (for example "LANG,LC_*"). The default is

no. Enabling environment processing may enable users to bypass

access restrictions in some configurations using mechanisms such

as LD_PRELOAD.

PermitUserRC

Specifies whether any ~/.ssh/rc file is executed. The default is

yes.

PerSourceMaxStartups

Specifies the number of unauthenticated connections allowed from

a given source address, or ?none? if there is no limit. This

limit is applied in addition to MaxStartups, whichever is lower.

The default is none.

PerSourceNetBlockSize

Specifies the number of bits of source address that are grouped together for the purposes of applying PerSourceMaxStartups lim?its. Values for IPv4 and optionally IPv6 may be specified, sepa?rated by a colon. The default is 32:128, which means each ad?dress is considered individually.

PidFile

Specifies the file that contains the process ID of the SSH dae?mon, or none to not write one. The default is /var/run/sshd.pid.

Port   Specifies the port number that sshd(8) listens on. The default is 22. Multiple options of this type are permitted. See also ListenAddress.

PrintLastLog

Specifies whether sshd(8) should print the date and time of the last user login when a user logs in interactively. The default is yes.

PrintMotd

Specifies whether sshd(8) should print /etc/motd when a user logs in interactively. (On some systems it is also printed by the shell, /etc/profile, or equivalent.) The default is yes.

PubkeyAcceptedAlgorithms

The default is handled system-wide by crypto-policies(7). Infor?mation about defaults, how to modify the defaults and how to cus?tomize existing policies with sub-policies are present in manual page update-crypto-policies(8).

Specifies the signature algorithms that will be accepted for pub?lic key authentication as a list of comma-separated patterns. Alternately if the specified list begins with a ?+? character, then the specified algorithms will be appended to the built-in openssh default set instead of replacing them. If the specified list begins with a ?-? character, then the specified algorithms (including wildcards) will be removed from the built-in openssh default set instead of replacing them. If the specified list be?gins with a ?^? character, then the specified algorithms will be

placed at the head of the built-in openssh default set.

The list of available signature algorithms may also be obtained

using "ssh -Q PubkeyAcceptedAlgorithms".

PubkeyAuthOptions

Sets one or more public key authentication options. The sup‐

ported keywords are: none (the default; indicating no additional

options are enabled), touch-required and verify-required.

The touch-required option causes public key authentication using

a FIDO authenticator algorithm (i.e. ecdsa-sk or ed25519-sk) to

always require the signature to attest that a physically present

user explicitly confirmed the authentication (usually by touching

the authenticator). By default, sshd(8) requires user presence

unless overridden with an authorized_keys option. The

touch-required flag disables this override.

The verify-required option requires a FIDO key signature attest

that the user was verified, e.g. via a PIN.

Neither the touch-required or verify-required options have any

effect for other, non-FIDO, public key types.

PubkeyAuthentication

Specifies whether public key authentication is allowed. The de‐

fault is yes.

RekeyLimit

Specifies the maximum amount of data that may be transmitted be‐

fore the session key is renegotiated, optionally followed by a

maximum amount of time that may pass before the session key is

renegotiated. The first argument is specified in bytes and may

have a suffix of ?K?, ?M?, or ?G? to indicate Kilobytes,

Megabytes, or Gigabytes, respectively. The default is between

?1G? and ?4G?, depending on the cipher. The optional second

value is specified in seconds and may use any of the units docu‐

mented in the TIME FORMATS section. The default value for

RekeyLimit is default none, which means that rekeying is per‐

formed after the cipher's default amount of data has been sent or

received and no time based rekeying is done.

RequiredRSASize

Specifies the minimum RSA key size (in bits) that sshd(8) will accept. User and host-based authentication keys smaller than this limit will be refused. The default is 1024 bits. Note that this limit may only be raised from the default.

RevokedKeys

Specifies revoked public keys file, or none to not use one. Keys listed in this file will be refused for public key authentica? tion. Note that if this file is not readable, then public key authentication will be refused for all users. Keys may be speci? fied as a text file, listing one public key per line, or as an OpenSSH Key Revocation List (KRL) as generated by ssh-keygen(1). For more information on KRLs, see the KEY REVOCATION LISTS sec? tion in ssh-keygen(1).

RDomain

Specifies an explicit routing domain that is applied after au? thentication has completed. The user session, as well as any forwarded or listening IP sockets, will be bound to this rdomain(4). If the routing domain is set to %D, then the domain in which the incoming connection was received will be applied.

SecurityKeyProvider

Specifies a path to a library that will be used when loading FIDO authenticator-hosted keys, overriding the default of using the built-in USB HID support.

SetEnv Specifies one or more environment variables to set in child ses? sions started by sshd(8) as ?NAME=VALUE?. The environment value may be quoted (e.g. if it contains whitespace characters). Envi? ronment variables set by SetEnv override the default environment and any variables specified by the user via AcceptEnv or PermitUserEnvironment.

StreamLocalBindMask

Sets the octal file creation mode mask (umask) used when creating

a Unix-domain socket file for local or remote port forwarding.

This option is only used for port forwarding to a Unix-domain

socket file.

The default value is 0177, which creates a Unix-domain socket

file that is readable and writable only by the owner. Note that

not all operating systems honor the file mode on Unix-domain

socket files.

StreamLocalBindUnlink

Specifies whether to remove an existing Unix-domain socket file

for local or remote port forwarding before creating a new one.

If the socket file already exists and StreamLocalBindUnlink is

not enabled, sshd will be unable to forward the port to the Unix-

domain socket file. This option is only used for port forwarding

to a Unix-domain socket file.

The argument must be yes or no. The default is no.

StrictModes

Specifies whether sshd(8) should check file modes and ownership

of the user's files and home directory before accepting login.

This is normally desirable because novices sometimes accidentally

leave their directory or files world-writable. The default is

yes. Note that this does not apply to ChrootDirectory, whose

permissions and ownership are checked unconditionally.

Subsystem

Configures an external subsystem (e.g. file transfer daemon).

Arguments should be a subsystem name and a command (with optional

arguments) to execute upon subsystem request.

The command sftp-server implements the SFTP file transfer subsys?

tem.

Alternately the name internal-sftp implements an in-process SFTP

server. This may simplify configurations using ChrootDirectory

to force a different filesystem root on clients.

By default no subsystems are defined.

SyslogFacility

Gives the facility code that is used when logging messages from
sshd(8). The possible values are: DAEMON, USER, AUTH, AUTHPRIV,
LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7.
The default is AUTH.

TCPKeepAlive

Specifies whether the system should send TCP keepalive messages
to the other side. If they are sent, death of the connection or
crash of one of the machines will be properly noticed. However,
this means that connections will die if the route is down tempo?
rarily, and some people find it annoying. On the other hand, if
TCP keepalives are not sent, sessions may hang indefinitely on
the server, leaving "ghost" users and consuming server resources.
The default is yes (to send TCP keepalive messages), and the
server will notice if the network goes down or the client host
crashes. This avoids infinitely hanging sessions.
To disable TCP keepalive messages, the value should be set to no.

TrustedUserCAKeys

Specifies a file containing public keys of certificate authori?
ties that are trusted to sign user certificates for authentica?
tion, or none to not use one. Keys are listed one per line;
empty lines and comments starting with ?#? are allowed. If a
certificate is presented for authentication and has its signing
CA key listed in this file, then it may be used for authentica?
tion for any user listed in the certificate's principals list.
Note that certificates that lack a list of principals will not be
permitted for authentication using TrustedUserCAKeys. For more
details on certificates, see the CERTIFICATES section in
ssh-keygen(1).

UseDNS Specifies whether sshd(8) should look up the remote host name,
and to check that the resolved host name for the remote IP ad?
dress maps back to the very same IP address.
If this option is set to no (the default) then only addresses and
not host names may be used in ~/.ssh/authorized_keys from and

sshd_config Match Host directives.

UsePAM Enables the Pluggable Authentication Module interface. If set to

yes this will enable PAM authentication using

KbdInteractiveAuthentication and PasswordAuthentication in addi?

tion to PAM account and session module processing for all authen?

tication types.

Because PAM keyboard-interactive authentication usually serves an

equivalent role to password authentication, you should disable

either PasswordAuthentication or KbdInteractiveAuthentication.

If UsePAM is enabled, you will not be able to run sshd(8) as a

non-root user. The default is no.

VersionAddendum

Optionally specifies additional text to append to the SSH proto?

col banner sent by the server upon connection. The default is

none.

X11DisplayOffset

Specifies the first display number available for sshd(8)'s X11

forwarding. This prevents sshd from interfering with real X11

servers. The default is 10.

X11MaxDisplays

Specifies the maximum number of displays available for sshd(8)'s

X11 forwarding. This prevents sshd from exhausting local ports.

The default is 1000.

X11Forwarding

Specifies whether X11 forwarding is permitted. The argument must

be yes or no. The default is no.

When X11 forwarding is enabled, there may be additional exposure

to the server and to client displays if the sshd(8) proxy display

is configured to listen on the wildcard address (see

X11UseLocalhost), though this is not the default. Additionally,

the authentication spoofing and authentication data verification

and substitution occur on the client side. The security risk of

using X11 forwarding is that the client's X11 display server may

be exposed to attack when the SSH client requests forwarding (see

the warnings for ForwardX11 in ssh_config(5)). A system adminis?

trator may have a stance in which they want to protect clients

that may expose themselves to attack by unwittingly requesting

X11 forwarding, which can warrant a no setting.

Note that disabling X11 forwarding does not prevent users from

forwarding X11 traffic, as users can always install their own

forwarders.

X11UseLocalhost

Specifies whether sshd(8) should bind the X11 forwarding server

to the loopback address or to the wildcard address. By default,

sshd binds the forwarding server to the loopback address and sets

the hostname part of the DISPLAY environment variable to

localhost. This prevents remote hosts from connecting to the

proxy display. However, some older X11 clients may not function

with this configuration. X11UseLocalhost may be set to no to

specify that the forwarding server should be bound to the wild?

card address. The argument must be yes or no. The default is

yes.

XAuthLocation

Specifies the full pathname of the xauth(1) program, or none to

not use one. The default is /usr/bin/xauth.

TIME FORMATS

sshd(8) command-line arguments and configuration file options that spec?

ify time may be expressed using a sequence of the form: time[qualifier],

where time is a positive integer value and qualifier is one of the fol?

lowing:

?none? seconds

s | S   seconds

m | M   minutes

h | H   hours

d | D   days

w | W   weeks

Each member of the sequence is added together to calculate the total time value.

Time format examples:

600   600 seconds (10 minutes)

10m   10 minutes

1h30m   1 hour 30 minutes (90 minutes)

TOKENS

Arguments to some keywords can make use of tokens, which are expanded at runtime:

%%   A literal ?%?.

%D   The routing domain in which the incoming connection was re?

   ceived.

%F   The fingerprint of the CA key.

%f   The fingerprint of the key or certificate.

%h   The home directory of the user.

%i   The key ID in the certificate.

%K   The base64-encoded CA key.

%k   The base64-encoded key or certificate for authentication.

%s   The serial number of the certificate.

%T   The type of the CA key.

%t   The key or certificate type.

%U   The numeric user ID of the target user.

%u   The username.

AuthorizedKeysCommand accepts the tokens %%, %f, %h, %k, %t, %U, and %u.

AuthorizedKeysFile accepts the tokens %%, %h, %U, and %u.

AuthorizedPrincipalsCommand accepts the tokens %%, %F, %f, %h, %i, %K,

%k, %s, %T, %t, %U, and %u.

AuthorizedPrincipalsFile accepts the tokens %%, %h, %U, and %u.

ChrootDirectory accepts the tokens %%, %h, %U, and %u.

RoutingDomain accepts the token %D.

FILES

/etc/ssh/sshd_config

   Contains configuration data for sshd(8). This file should be

writable by root only, but it is recommended (though not neces?

sary) that it be world-readable.

SEE ALSO

sftp-server(8), sshd(8)

AUTHORS

OpenSSH is a derivative of the original and free ssh 1.2.12 release by

Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo

de Raadt and Dug Song removed many bugs, re-added newer features and cre?

ated OpenSSH. Markus Friedl contributed the support for SSH protocol

versions 1.5 and 2.0. Niels Provos and Markus Friedl contributed support

for privilege separation.

BSD   August 12, 2021   BSD