NAME

sshd_config − OpenSSH SSH daemon configuration file

SYNOPSIS

/etc/ssh/sshd_config

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-argument pairs, one per line. 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 keywords 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** in ssh_config(5) for how to configure the client. Note that environment passing is only supported for protocol 2. 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", "inet" (use IPv4 only), or "inet6" (use IPv6 only). The default is "any".

AllowAgentForwarding
Specifies whether ssh-agent(1) forwarding is permitted. The default is "yes". Note that disabling agent forwarding does not improve 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 directives are processed in the following order: **DenyUsers**, **AllowUsers**, **DenyGroups**, and finally **AllowGroups**.

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

AllowTcpForwarding
Specifies whether TCP forwarding is permitted. The default is "yes". Note that disabling TCP forwarding does not improve 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. The allow/deny directives are processed in the following order: **DenyUsers**, **AllowUsers**, **DenyGroups**, and finally **AllowGroups**.

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

AuthorizedKeysFile
Specifies the file that contains the public keys that can be used for user authentication. **AuthorizedKeysFile** may contain tokens of the form %T which are substituted during connection setup. The following tokens are defined: %% is replaced by a literal '%', %h is replaced by the home directory of the user being authenticated, and %u is replaced by the username of that user. After expansion, **AuthorizedKeysFile** is taken to be an absolute path or one relative to the user's home directory. The default is ".ssh/authorized_keys".

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; empty lines and comments starting with '#' are ignored.

AuthorizedPrincipalsFile may contain tokens of the form %T which are substituted during connection setup. The following tokens are defined: %% is replaced by a literal '%', %h is replaced by the home directory of the user being authenticated, and %u is replaced by the username of that user. After expansion, **AuthorizedPrincipalsFile** is taken to be an absolute path or one relative to the user's home directory.

The default is not to use a principals 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 authentication 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. This option is only available for protocol version 2. By default, no banner is displayed.

ChallengeResponseAuthentication
Specifies whether challenge-response authentication is allowed (e.g. via PAM or though authentication styles supported in login.conf(5)) The default is "yes".

ChrootDirectory
Specifies a path to chroot(2) to after authentication. This path, and all its components, must be root-owned directories that are not writable by any other user or group. After the chroot, sshd(8) changes the working directory to the user's home directory.

The path may contain the following tokens that are expanded at runtime once the connecting user has been authenticated: %% is replaced by a literal '%', %h is replaced by the home directory of the user being authenticated, and %u is replaced by the username of that user.

The **ChrootDirectory** must contain the necessary files and directories 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), arandom(4) and tty(4) devices. For file transfer sessions using "sftp", no additional configuration of the environment is necessary if the in-process sftp server is used, though sessions which use logging do require /dev/log inside the chroot directory (see sftp-server(8) for details).

The default is not to chroot(2).

Ciphers

Specifies the ciphers allowed for protocol version 2. Multiple ciphers must be comma-separated. The supported ciphers are “3des-cbc”, “aes128-cbc”, “aes192-cbc”, “aes256-cbc”, “aes128-ctr”, “aes192-ctr”, “aes256-ctr”, “arcfour128”, “arcfour256”, “arcfour”, “blowfish-cbc”, “rijndael-cbc@lysator.liu.se”, and “cast128-cbc”. The default is:

```
aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,
aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,aes192-cbc,
aes256-cbc,arcfour,rijndael-cbc@lysator.liu.se
```

ClientAliveCountMax

Sets the number of client alive messages (see below) which may be sent without 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** (below). The client alive messages are sent through the encrypted channel and therefore will not be spoofable. The TCP keepalive option enabled by **TCPKeepAlive** is spoofable. The client alive mechanism is valuable when the client or server depend on knowing when a connection has become inactive.

The default value is 3. If **ClientAliveInterval** (see below) is set to 15, and **ClientAliveCountMax** is left at the default, unresponsive SSH clients will be disconnected after approximately 45 seconds. This option applies to protocol version 2 only.

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. This option applies to protocol version 2 only.

Compression

Specifies whether compression is allowed, or delayed until the user has authenticated successfully. The argument must be “yes”, “delayed”, or “no”. The default is “delayed”.

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 recognized. By default, login is allowed for all groups. The allow/deny directives are processed in the following order: **DenyUsers**, **AllowUsers**, **DenyGroups**, and finally **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 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. The allow/deny directives are processed in the following order: **DenyUsers**, **AllowUsers**, **DenyGroups**, and finally **AllowGroups**.

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

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**.

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 connect. 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". Note that this option applies to protocol version 2 only.

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". Note that this option applies to protocol version 2 only.

GSSAPICleanupCredentials
Specifies whether to automatically destroy the user's credentials cache on logout. The default is "yes". Note that this option applies to protocol version 2 only.

GSSAPIStrictAcceptorCheck
Determines whether to be strict about the identity of the GSSAPI acceptor a client authenticates against. If "yes" then the client must authenticate against the host service on the current hostname. If "no" then the client may authenticate against any service key stored in the machine's default store. This facility is provided to assist with operation on multi homed machines. The default is "yes". Note that this option applies only to protocol version 2 GSSAPI connections, and setting it to "no" may only work with recent Kerberos GSSAPI libraries.

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".

HostbasedAuthentication
Specifies whether rhosts or /etc/hosts.equiv authentication together with successful public key client host authentication is allowed (host-based authentication). This option is similar to **RhostsRSAAuthentication** and applies to protocol version 2 only. The default is "no".

HostbasedUsesNameFromPacketOnly
Specifies whether or not the server will attempt to perform a reverse 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 certificate's public key must match a private host key already specified 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 default is `/etc/ssh/ssh_host_key` for protocol version 1, and `/etc/ssh/ssh_host_dsa_key`, `/etc/ssh/ssh_host_ecdsa_key` and `/etc/ssh/ssh_host_rsa_key` for protocol version 2. Note that `sshd`(8) will refuse to use a file if it is group/world-accessible. It is possible to have multiple host key files. "rsa1" keys are used for version 1 and "dsa", "ecdsa" or "rsa" are used for version 2 of the SSH protocol.

IgnoreRhosts
Specifies that `.rhosts` and `.shosts` files will not be used in **RhostsRSAAuthentication** or **HostbasedAuthentication**.

`/etc/hosts.equiv` and `/etc/ssh/shosts.equiv` are still used. The default is "yes".

IgnoreUserKnownHosts
Specifies whether `sshd`(8) should ignore the user's `~/.ssh/known_hosts` during **RhostsRSAAuthentication** or **HostbasedAuthentication**. The default is "no".

KbdInteractiveAuthentication
Specifies whether to allow keyboard-interactive authentication. The argument to this keyword must be "yes" or "no". The default is to use whatever value **ChallengeResponseAuthentication** is set to (by default "yes").

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 password 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".

KerberosUseKuserok
Specifies whether to look at .k5login file for user's aliases. The default is "yes".

KexAlgorithms
Specifies the available KEX (Key Exchange) algorithms. Multiple algorithms must be comma-separated. The default is "diffie-hellman-group-exchange-sha256", "diffie-hellman-group-exchange-sha1", "diffie-hellman-group14-sha1", "diffie-hellman-group1-sha1".

KeyRegenerationInterval
In protocol version 1, the ephemeral server key is automatically regenerated after this many seconds (if it has been used). The purpose of regeneration is to prevent decrypting captured sessions by later breaking into the machine and stealing the keys. The key is never stored anywhere. If the value is 0, the key is never regenerated. The default is 3600 (seconds).

ListenAddress
Specifies the local addresses sshd(8) should listen on. The following forms may be used:

ListenAddress *host*|*IPv4_addr*|*IPv6_addr*
ListenAddress *host*|*IPv4_addr*:*port*
ListenAddress [*host*|*IPv6_addr*]:*port*

If *port* is not specified, sshd will listen on the address and all prior **Port** options specified. The default is to listen on all local addresses. Multiple **ListenAddress** options are permitted. Additionally, any **Port** options must precede this option for non-port qualified addresses.

LoginGraceTime
The server disconnects after this time if the user has not successfully 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.

MACs Specifies the available MAC (message authentication code) algorithms. The MAC algorithm is used in protocol version 2 for data integrity protection. Multiple algorithms must be comma-separated. The default is:

```
hmac-md5,hmac-sha1,umac-64@openssh.com,
hmac-ripemd160,hmac-sha1-96,hmac-md5-96,
hmac-sha2-256,hmac-sha2-512,hmac-ripemd160@openssh.com
```

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, until either another **Match** line or the end of the file.

The arguments to **Match** are one or more criteria-pattern pairs. The available criteria are **User**, **Group**, **Host**, and **Address**. The match patterns may consist of single entries or comma-separated lists and may use the wildcard and negation operators described in the **PATTERNS** section of ssh_config(5).

The patterns in an **Address** criteria may additionally contain addresses to match in CIDR address/masklen format, e.g. “192.0.2.0/24” or “3ffe:ffff::/32”. Note that the mask length provided 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 **AllowAgentForwarding**, **AllowTcpForwarding**, **Banner**, **ChrootDirectory**, **ForceCommand**, **GatewayPorts**, **GSSAPIAuthentication**, **HostbasedAuthentication**, **KbdInteractiveAuthentication**, **KerberosAuthentication**, **KerberosUseKuserok**, **MaxAuthTries**, **MaxSessions**, **PubkeyAuthentication**, **AuthorizedKeysCommand**, **AuthorizedKeysCommandRunAs**, **PasswordAuthentication**, **PermitEmptyPasswords**, **PermitOpen**, **PermitRootLogin**, **RequiredAuthentications1**, **RequiredAuthentications2**, **RhostsRSAAuthentication**, **RSAAuthentication**, **X11DisplayOffset**, **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 sessions permitted per network connection. The default is 10.

MaxStartups
Specifies the maximum number of concurrent unauthenticated connections to the SSH daemon. Additional connections will be dropped until authentication succeeds or the **LoginGraceTime** expires 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 connection attempts are refused if the number of unauthenticated connections reaches "full" (60).

PasswordAuthentication
Specifies whether password authentication is allowed. The default 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".

PermitOpen
Specifies the destinations to which TCP port forwarding is permitted. The forwarding specification must be one of the following forms:

PermitOpen *host*:*port*
PermitOpen *IPv4_addr*:*port*
PermitOpen *[IPv6_addr]*:*port*

Multiple forwards may be specified by separating them with whitespace. An argument of "any" can be used to remove all restrictions and permit any forwarding requests. By default all port forwarding requests are permitted.

PermitRootLogin
Specifies whether root can log in using ssh(1). The argument must be "yes", "without-password", "forced-commands-only", or "no". The default is "yes".

If this option is set to "without-password", password authentication is 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.

PermitTunnel
Specifies whether tun(4) device forwarding is allowed. The argument 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".

PermitUserEnvironment
Specifies whether ~/.ssh/environment and **environment=** options in ~/.ssh/authorized_keys are processed by sshd(8). The default is "no". Enabling environment processing may enable users to bypass access restrictions in some configurations using mechanisms such as LD_PRELOAD.

PidFile
Specifies the file that contains the process ID of the SSH daemon. 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".

Protocol
Specifies the protocol versions sshd(8) supports. The possible values are '1' and '2'. Multiple versions must be comma-separated. The default is "2,1". Note that the order of the protocol list does not indicate preference, because the client selects among multiple protocol versions offered by the server. Specifying "2,1" is identical to "1,2".

PubkeyAuthentication
Specifies whether public key authentication is allowed. The default is "yes". Note that this option applies to protocol version 2 only.

AuthorizedKeysCommand
Specifies a program to be used for lookup of the user's public keys. The program will be invoked with its first argument the name of the user being authorized, and should produce on standard output AuthorizedKeys lines (see AUTHORIZED_KEYS in sshd(8)). By default (or when set to the empty string) there is no AuthorizedKeysCommand run. If the AuthorizedKeysCommand does not successfully authorize the user, authorization falls through to the AuthorizedKeysFile. Note that this option has an effect only with PubkeyAuthentication turned on.

AuthorizedKeysCommandRunAs
Specifies the user under whose account the AuthorizedKeysCommand is run. Empty string (the default value) means the user being authorized is used.

RequiredAuthentications[12]
Specifies required methods of authentications that has to succeed before authorizing the connection. (RequiredAuthentication1 for Protocol version 1, and RequiredAuthentication2 for v2)

```
RequiredAuthentications1 method[,method...]
RequiredAuthentications2 method[,method...]
```

Example 1:

```
RequiredAuthentications2 password,hostbased
```

Example 2:

```
RequiredAuthentications2 publickey,password
```

Available methods:

```
password, keyboard-interactive, publickey, hostbased, gssapi-keyex, gssapi-wi
```

RevokedKeys
Specifies a list of revoked public keys. Keys listed in this file will be refused for public key authentication. Note that if this file is not readable, then public key authentication will be refused for all users.

RhostsRSAAuthentication
Specifies whether rhosts or /etc/hosts.equiv authentication together with successful RSA host authentication is allowed. The default is “no”. This option applies to protocol version 1 only.

RSAAuthentication
Specifies whether pure RSA authentication is allowed. The default is “yes”. This option applies to protocol version 1 only.

ServerKeyBits
Defines the number of bits in the ephemeral protocol version 1 server key. The minimum value is 512, and the default is 1024.

ShowPatchLevel
Specifies whether **sshd** will display the patch level of the binary in the identification string. The patch level is set at compile-time. The default is “no”. This option applies to protocol version 1 only.

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”.

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(8) implements the “sftp” file transfer subsystem.

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. Note that this option applies to protocol version 2 only.

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 temporarily, 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 authorities that are trusted sign user certificates for authentication. 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 authentication 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 in certificates, please see the **CERTIFICATES** section in ssh-keygen(1).

UseDNS
Specifies whether sshd(8) should look up the remote host name and check that the resolved host name for the remote IP address maps back to the very same IP address. The default is "yes".

UseLogin
Specifies whether login(1) is used for interactive login sessions. The default is "no". Note that login(1) is never used for remote command execution. Note also, that if this is enabled, **X11Forwarding** will be disabled because login(1) does not know how to handle xauth(1) cookies. If **UsePrivilegeSeparation** is specified, it will be disabled after authentication.

UsePAM
Enables the Pluggable Authentication Module interface. If set to "yes" this will enable PAM authentication using **ChallengeResponseAuthentication** and **PasswordAuthentication** in addition to PAM account and session module processing for all authentication types.

Because PAM challenge-response authentication usually serves an equivalent role to password authentication, you should disable either **PasswordAuthentication** or **ChallengeResponseAuthentication.**

If **UsePAM** is enabled, you will not be able to run sshd(8) as a non-root user. The default is "no".

UsePrivilegeSeparation
Specifies whether sshd(8) separates privileges by creating an unprivileged child process to deal with incoming network traffic. After successful authentication, another process will be created that has the privilege of the authenticated user. The goal of privilege separation is to prevent privilege escalation by containing any corruption within the unprivileged processes. The default is "yes".

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.

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** below), 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 administrator 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. X11 forwarding is automatically disabled if **UseLogin** is

enabled.

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 wildcard address. The argument must be "yes" or "no". The default is "yes".

XAuthLocation
Specifies the full pathname of the xauth(1) program. The default is /usr/bin/xauth.

TIME FORMATS

sshd(8) command-line arguments and configuration file options that specify 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 following:

⟨**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)

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 necessary) that it be world-readable.

SEE ALSO

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 created 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.