NAME

ssh − secure shell client (remote login program)

SYNOPSIS

ssh [**−l** *login_name*] **hostname** [*command*]

ssh [**−a**] [**−c** *idea*|*blowfish*|*des*|*3des*|*arcfour*|*none*] [**−e** *escape_char*] [**−i** *identity_file*] [**−l** *login_name*] [**−n**] [**−k**] [**−V**] [**−o** *option*] [**−p** *port*] [**−q**] [**−P**] [**−t**] [**−v**] [**−x**] [**−C**] [**−g**] [**−L** *port*:*host*:*hostport*] [**−R** *port*:*host*:*hostport*] *hostname* [*command*]

DESCRIPTION

Ssh (Secure Shell) a program for logging into a remote machine and for executing commands in a remote machine. It is intended to replace rlogin and rsh, and provide secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel.

Ssh connects and logs into the specified *hostname*. The user must prove his/her identity to the remote machine using one of several methods.

First, if the machine the user logs in from is listed in */etc/hosts.equiv* or */etc/shosts.equiv* on the remote machine, and the user names are the same on both sides, the user is immediately permitted to log in. Second, if *.rhosts* or *.shosts* exists in the user's home directory on the remote machine and contains a line containing the name of the client machine and the name of the user on that machine, the user is permitted to log in. This form of authentication alone is normally not allowed by the server because it is not secure.

The second (and primary) authentication method is the **rhosts** or **hosts.equiv** method combined with RSA-based host authentication. It means that if the login would be permitted by *.rhosts*, *.shosts*, */etc/hosts.equiv*, *or /etc/shosts.equiv*, and additionally it can verify the client's host key (see *$HOME/.ssh/known_hosts* and */etc/ssh_known_hosts* in the **FILES** section), only then login is permitted. This authentication method closes security holes due to IP spoofing, DNS spoofing and routing spoofing. [Note to the administrator: */etc/hosts.equiv*, *.rhosts*, and the rlogin/rsh protocol in general, are inherently insecure and should be disabled if security is desired.]

As a third authentication method, **ssh** supports RSA based authentication. The scheme is based on public-key cryptography: there are cryptosystems where encryption and decryption are done using separate keys, and it is not possible to derive the decryption key from the encryption key. RSA is one such system. The idea is that each user creates a public/private key pair for authentication purposes. The server knows the public key, and only the user knows the private key. The file *$HOME/.ssh/authorized_keys* lists the public keys that are permitted for logging in. When the user logs in, the **ssh** program tells the server which key pair it would like to use for authentication. The server checks if this key is permitted, and if so, sends the user (actually the **ssh** program running on behalf of the user) a challenge, a random number, encrypted by the user's public key. The challenge can only be decrypted using the proper private key. The user's client then decrypts the challenge using the private key, proving that he/she knows the private key but without disclosing it to the server.

Ssh implements the RSA authentication protocol automatically. The user creates his/her RSA key pair by running **ssh-keygen**(1). This stores the private key in *.ssh/identity* and the public key in *.ssh/identity.pub* in the user's home directory. The user should then copy the *identity.pub* to *.ssh/authorized_keys* in his/her home directory on the remote machine (the *authorized_keys* file corresponds to the conventional *.rhosts* file, and has one key per line, though the lines can be very long). After this, the user can log in without giving the password. RSA authentication is much more secure than rhosts authentication.

The most convenient way to use RSA authentication may be with an authentication agent. See **ssh-agent**(1) for more information.

As a fourth authentication method, **ssh** supports authentication through TIS authentication server. The idea is that **ssh** asks TIS **authsrv**(8) to authenticate the user. Sometime, usernames in the TIS database cannot be the same as the local users. This can be the case if the user authenticates itself with a smartcard or a

Digipass. In that case, the username in the database is usually known as the serial number of the authentification device. The file */etc/sshd_tis.map* contains the mapping between local users and their corresponding name in the TIS database. If the file does not exist or the user is not found, the corresponding name in the TIS database is supposed to be the same.

If other authentication methods fail, **ssh** prompts the user for a password. The password is sent to the remote host for checking; however, since all communications are encrypted, the password cannot be seen by someone listening on the network.

When the user's identity has been accepted by the server, the server either executes the given command, or logs into the machine and gives the user a normal shell on the remote machine. All communication with the remote command or shell will be automatically encrypted.

If a pseudo-terminal has been allocated (normal login session), the user can disconnect with "˜.", and suspend **ssh** with "˜^Z". All forwarded connections can be listed with "˜#", and if the session blocks waiting for forwarded X11 or TCP/IP connections to terminate, it can be backgrounded with "˜&" (this should not be used while the user shell is active, as it can cause the shell to hang). All available escapes can be listed with "˜?".

A single tilde character can be sent as "˜˜" (or by following the tilde by a character other than those described above). The escape character must always follow a newline to be interpreted as special. The escape character can be changed in configuration files or on the command line.

If no pseudo tty has been allocated, the session is transparent and can be used to reliably transfer binary data. On most systems, setting the escape character to ‘‘none’’ will also make the session transparent even if a tty is used.

The session terminates when the command or shell in on the remote machine exists and all X11 and TCP/IP connections have been closed. The exit status of the remote program is returned as the exit status of **ssh.**

If the user is using X11 (the **DISPLAY** environment variable is set), the connection to the X11 display is automatically forwarded to the remote side in such a way that any X11 programs started from the shell (or command) will go through the encrypted channel, and the connection to the real X server will be made from the local machine. The user should not manually set **DISPLAY**. Forwarding of X11 connections can be configured on the command line or in configuration files.

The DISPLAY value set by **ssh** will point to the server machine, but with a display number greater than zero. This is normal, and happens because **ssh** creates a "proxy" X server on the server machine for forwarding the connections over the encrypted channel.

Ssh will also automatically set up Xauthority data on the server machine. For this purpose, it will generate a random authorization cookie, store it in Xauthority on the server, and verify that any forwarded connections carry this cookie and replace it by the real cookie when the connection is opened. The real authentication cookie is never sent to the server machine (and no cookies are sent in the plain).

If the user is using an authentication agent, the connection to the agent is automatically forwarded to the remote side unless disabled on command line or in a configuration file.

Forwarding of arbitrary TCP/IP connections over the secure channel can be specified either on command line or in a configuration file. One possible application of TCP/IP forwarding is a secure connection to an electronic purse; another is going trough firewalls.

Ssh automatically maintains and checks a database containing RSA-based identifications for all hosts it has ever been used with. The database is stored in *.ssh/known_hosts* in the user's home directory. Additionally, the file */etc/ssh_known_hosts* is automatically checked for known hosts. Any new hosts are automatically added to the user's file. If a host's identification ever changes, **ssh** warns about this and disables password authentication to prevent a trojan horse from getting the user's password. Another purpose of this mechanism is to prevent man-in-the-middle attacks which could otherwise be used to circumvent the encryption. The **StrictHostKeyChecking** option (see below) can be used to prevent logins to machines whose host key is not known or has changed.

OPTIONS

−a Disables forwarding of the authentication agent connection. This may also be specified on a per-host basis in the configuration file.

−c *idea | des | 3des | blowfish | arcfour | none*
Selects the cipher to use for encrypting the session. **idea** is used by default. It is believed to be secure. **des** is the data encryption standard, but is breakable by governments, large corporations, and major criminal organizations. **3des** (triple-des) is encrypt-decrypt-encrypt triple with three different keys. It is presumably more secure than DES. It is used as default if both sites do not support IDEA. **blowfish** is an encryption algorithm invented by Bruce Schneier. It uses 128 bit keys. **arcfour** is an algorithm published in the Usenet News in 1995. This algorithm is believed to be equivalent with the RC4 cipher from RSA Data Security (RC4 is a trademark of RSA Data Security). This is the fastest algorithm currently supported. **none** disables encryption entirely; it is only intended for debugging, and it renders the connection insecure.

−e *ch | ˆch | none*
Sets the escape character for sessions with a pty (default: ˜). The escape character is only recognized at the beginning of a line. The escape character followed by a dot (.) closes the connection, followed by control-Z suspends the connection, and followed by itself sends the escape character once. Setting the character to ´none´ disables any escapes and makes the session fully transparent.

−f Requests ssh to go to background after authentication is done and forwardings have been established. This is useful if ssh is going to ask for passwords or passphrases, but the user wants it in the background. This may also be useful in scripts. This implies **−n.** The recommended way to start X11 programs at a remote site is with something like "ssh -f host xterm".

−i *identity_file*
Selects the file from which the identity (private key) for **RSA** authentication is read. Default is *.ssh/identity* in the user's home directory. Identity files may also be specified on a per-host basis in the configuration file. It is possible to have multiple −i options (and multiple identities specified in configuration files).

−k Disables forwarding of the kerberos tickets. This may also be specified on a per-host basis in the configuration file.

-l *login_name*
Specifies the user to log in as on the remote machine. This may also be specified on a per-host basis in the configuration file.

−n Redirects stdin from /dev/null (actually, prevents reading from stdin). This must be used when **ssh** is run in the background. A common trick is to use this to run X11 programs in a remote machine. For example, "ssh -n shadows.cs.hut.fi emacs &" will start an emacs on shadows.cs.hut.fi, and the X11 connection will be automatically forwarded over an encrypted channel. The **ssh** program will be put in the background. (This does not work if **ssh** needs to ask for a password or passphrase; see also the -f option.)

−o *'option'*
Can be used to give options in the format used in the config file. This is useful for specifying options for which there is no separate command-line flag. The option has the same format as a line in the configuration file.

−p *port* Port to connect to on the remote host. This can be specified on a per-host basis in the configuration file.

−q Quiet mode. Causes all warning and diagnostic messages to be suppressed. Only fatal errors are displayed.

−P Use non privileged port. With this you cannot use rhosts or rsarhosts authentications, but it can be used to bypass some firewalls that dont allow privileged source ports to pass.

−t Force pseudo-tty allocation. This can be used to execute arbitary screen-based programs on a remote machine, which can be very useful e.g. when implementing menu services.

−v Verbose mode. Causes **ssh** to print debugging messages about its progress. This is helpful in debugging connection, authentication, and configuration problems.

−V Print only version number and exit.

−g Allows remote hosts to connect local port forwarding ports. The default is that only localhost may connect to locally binded ports.

−x Disables X11 forwarding. This can also be specified on a per-host basis in a configuration file.

−C Requests compression of all data (including stdin, stdout, stderr, and data for forwarded X11 and TCP/IP connections). The compression algorithm is the same used by gzip, and the "level" can be controlled by the **CompressionLevel** option (see below). Compression is desirable on modem lines and other slow connections, but will only slow down things on fast networks. The default value can be set on a host-by-host basis in the configuration files; see the **Compress** option below.

−L *port:host:hostport*
Specifies that the given port on the local (client) host is to be forwarded to the given host and port on the remote side. This works by allocating a socket to listen to **port** on the local side, and whenever a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to **host:hostport** from the remote machine. Port forwardings can also be specified in the configuration file. Only root can forward privileged ports.

−R *port:host:hostport*
Specifies that the given port on the remote (server) host is to be forwarded to the given host and port on the local side. This works by allocating a socket to listen to **port** on the remote side, and whenever a connection is made to this port, the connection is forwarded over the secure channel, and a connection is made to **host:hostport** from the local machine. Port forwardings can also be specified in the configuration file. Privileged ports can be forwarded only when logging in as root on the remote machine.

CONFIGURATION FILES

Ssh obtains configuration data from the following sources (in this order): command line options, user's configuration file (*$HOME/.ssh/config*), and system-wide configuration file (*/etc/ssh_config*). For each parameter, the first obtained value will be used. The configuration files contain sections bracketed by "Host" specifications, and that section is only applied for hosts that match one of the patterns given in the specification. The matched host name is the one given on the command line.

Since the first obtained value for each parameter is used, more host-specific declarations should be given near the beginning of the file, and general defaults at the end.

The configuration file has the following format:

Empty lines and lines starting with ´#´ are comments.

Otherwise a line is of the format "keyword arguments" or "keyword = arguments". The possible keywords and their meanings are as follows (note that the configuration files are case-sensitive, but keywords are case-insensitive):

Host Restricts the following declarations (up to the next **Host** keyword) to be only for those hosts that match one of the patterns given after the keyword. ´*´ and ´?´ can be as wildcards in the patterns. A single ´*´ as a pattern can be used to provide global defaults for all hosts. The host is the *hostname* argument given on the command line (i.e., the name is not converted to a canonicalized host name before matching).

BatchMode
If set to "yes", passphrase/password querying will be disabled. This option is useful in scripts and other batch jobs where you have no user to supply the password. The argument must be "**yes**" or "**no**".

Cipher Specifies the cipher to use for encrypting the session. Currently, *idea*, *des*, *3des*, *blowfish*, *arcfour*, and *none* are supported. The default is "idea" (or "3des" if "idea" is not supported by both hosts). Using "none" (no encryption) is intended only for debugging, and will render the connection insecure.

ClearAllForwardings
Clears all forwardings after reading all config files and parsing command line. This is usefull to disable forwardings in config file when you want to make second connection to host having forwardings in config file. Scp sets this on by default so it will not fail even if you have some forwardings set in config file.

Compression
Specifies whether to use compression. The argument must be "**yes**" or "**no**".

CompressionLevel
Specifies the compression level to use if compression is enable. The argument must be an integer from 1 (fast) to 9 (slow, best). The default level is 6, which is good for most applications. The meaning of the values is the same as in GNU GZIP.

ConnectionAttempts
Specifies the number of tries (one per second) to make before falling back to rsh or exiting. The argument must be an integer. This may be useful in scripts if the connection sometimes fails.

EscapeChar
Sets the escape character (default: ˜). The escape character can also be set on the command line. The argument should be a single character, ´ˆ´ followed by a letter, or ‘‘none’’ to disable the escape character entirely (making the connection transparent for binary data).

FallBackToRsh
Specifies that if connecting via **ssh** fails due to a connection refused error (there is no **sshd** listening on the remote host), **rsh** should automatically be used instead (after a suitable warning about the session being unencrypted). The argument must be "**yes**" or "**no**".

ForwardAgent
Specifies whether the connection to the authentication agent (if any) will be forwarded to the remote machine. The argument must be "**yes**" or "**no**".

ForwardX11
Specifies whether X11 connections will be automatically redirected over the secure channel and **DISPLAY** set. The argument must be "**yes**" or "**no**".

GatewayPorts

Specifies that also remote hosts may connect to locally forwarded ports. The argument must be "**yes**" or "**no**".

GlobalKnownHostsFile

Specifies a file to use instead of */etc/ssh_known_hosts*.

HostName

Specifies the real host name to log into. This can be used to specify nicnames or abbreviations for hosts. Default is the name given on the command line. Numeric IP addresses are also permitted (both on the command line and in **HostName** specifications).

IdentityFile

Specifies the file from which the user's RSA authentication identity is read (default *.ssh/identity* in the user's home directory). Additionally, any identities represented by the authentication agent will be used for authentication. The file name may use the tilde syntax to refer to a user's home directory. It is possible to have multiple identity files specified in configuration files; all these identities will be tried in sequence.

KeepAlive

Specifies whether the system should send 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.

The default is "yes" (to send keepalives), and the client will notice if the network goes down or the remote host dies. This is important in scripts, and many users want it too.

To disable keepalives, the value should be set to "no" in both the server and the client configuration files.

KerberosAuthentication

Specifies whether Kerberos V5 authentication will be used.

KerberosTgtPassing

Specifies whether a Kerberos V5 TGT will be forwarded to the server.

LocalForward

Specifies that a TCP/IP port on the local machine be forwarded over the secure channel to given host:port from the remote machine. The first argument must be a port number, and the second must be host:port. Multiple forwardings may be specified, and additional forwardings can be given on the command line. Only the root can forward privileged ports.

NumberOfPasswordPrompts

Specifies number of password prompts before giving up. The argument to must be integer. Note that server also limits number of attempts (currently 5), so setting this larger doesn't have any effect. Default value is one.

PasswordAuthentication

Specifies whether to use password authentication. The argument to this keyword must be "**yes**" or "**no**".

PasswordPromptHost
Specifies whether to include the remote host name in the password prompt. The argument to this keyword must be "**yes**" or "**no**".

PasswordPromptLogin
Specifies whether to include the remote login name in the password prompt. The argument to this keyword must be "**yes**" or "**no**".

Port Specifies the port number to connect on the remote host. Default is 22.

ProxyCommand
Specifies the command to use to connect to the server. The command string extends to the end of the line, and is executed with /bin/sh. In the command string, %h will be substituted by the host name to connect and %p by the port. The command can be basically anything, and should read from its stdin and write to its stdout. It should eventually connect an **sshd** server running on some machine, or execute "sshd -i" somewhere. Host key management will be done using the Host-Name of the host being connected (defaulting to the name typed by the user).

Note that **ssh** can also be configured to support the SOCKS system using the --with-socks4 or --with-socks5 compile-time configuration option.

RemoteForward
Specifies that a TCP/IP port on the remote machine be forwarded over the secure channel to given host:port from the local machine. The first argument must be a port number, and the second must be host:port. Multiple forwardings may be specified, and additional forwardings can be given on the command line. Only the root can forward privileged ports.

RhostsAuthentication
Specifies whether to try rhosts based authentication. Note that this declaration only affects the client side and has no effect whatsoever on security. Disabling rhosts authentication may reduce authentication time on slow connections when rhosts authentication is not used. Most servers do not permit RhostsAuthentication because it is not secure (see RhostsRSAAuthentication). The argument to this keyword must be "**yes**" or "**no**".

RhostsRSAAuthentication
Specifies whether to try rhosts based authentication with RSA host authentication. This is the primary authentication method for most sites. The argument must be "**yes**" or "**no**".

RSAAuthentication
Specifies whether to try RSA authentication. The argument to this keyword must be "**yes**" or "**no**". RSA authentication will only be attempted if the identity file exists, or an authentication agent is running.

StrictHostKeyChecking
If this flag is set to "yes", **ssh** ssh will never automatically add host keys to the *$HOME/.ssh/known_hosts* file, and refuses to connect hosts whose host key has changed. This provides maximum protection against trojan horse attacks. However, it can be somewhat annoying if you don't have good */etc/ssh_known_hosts* files installed and frequently connect new hosts. Basically this option forces the user to manually add any new hosts. Normally this option is set to

"ask", and new hosts will automatically be added to the known host files after you have confirmed you really want to do that. If this is set to "no" then new host will automatically be added to the known host files. The host keys of known hosts will be verified automatically in either case.

The argument must be "**yes**", "**no**" or "**ask**".

TISAuthentication
Specifies whether to try TIS authentication. The argument to this keyword must be "**yes**" or "**no**".

UsePrivilegedPort
Specifies whether to use privileged port when connecting to other end. The default is yes if rhosts or rsarhosts authentications are enabled.

User Specifies the user to log in as. This can be useful if you have a different user name in different machines. This saves the trouble of having to remember to give the user name on the command line.

UserKnownHostsFile
Specifies a file to use instead of *$HOME/.ssh/known_hosts*.

UseRsh
Specifies that rlogin/rsh should be used for this host. It is possible that the host does not at all support the **ssh** protocol. This causes **ssh** to immediately exec **rsh.** All other options (except **HostName**) are ignored if this has been specified. The argument must be "**yes**" or "**no**".

XAuthLocation
Specifies the path to xauth program.

ENVIRONMENT

Ssh will normally set the following environment variables:

DISPLAY
The DISPLAY variable indicates the location of the X11 server. It is automatically set by **ssh** to point to a value of the form "hostname:n" where hostname indicates the host where the shell runs, and n is an integer >= 1. Ssh uses this special value to forward X11 connections over the secure channel. The user should normally not set DISPLAY explicitly, as that will render the X11 connection insecure (and will require the user to manually copy any required authorization cookies).

HOME
Set to the path of the user's home directory.

LOGNAME
Synonym for USER; set for compatibility with systems that use this variable.

MAIL Set to point the user's mailbox.

PATH Set to the default PATH, as specified when compiling **ssh** or, on some systems, */etc/environment* or */etc/default/login*.

SSH_AUTH_SOCK
if exists, is used to indicate the path of a unix-domain socket used to communicate with the authentication agent (or its local representative).

SSH_CLIENT
Identifies the client end of the connection. The variable contains three space-separated values:

client ip-address, client port number, and server port number.

SSH_ORIGINAL_COMMAND
This will be the original command line of given by protocol if forced command is run. It can be used to fetch arguments etc from the other end.

SSH_TTY
This is set to the name of the tty (path to the device) associated with the current shell or command. If the current session has no tty, this variable is not set.

TZ The timezone variable is set to indicate the present timezone if it was set when the daemon was started (e.i., the daemon passes the value on to new connections).

USER Set to the name of the user logging in.

Additionally, **ssh** reads */etc/environment* and *$HOME/.ssh/environment*, and adds lines of the format *VAR-NAME=value* to the environment. Some systems may have still additional mechanisms for setting up the environment, such as */etc/default/login* on Solaris.

FILES

$HOME/.ssh/known_hosts
Records host keys for all hosts the user has logged into (that are not in */etc/ssh_known_hosts*). See **sshd** manual page.

$HOME/.ssh/random_seed
Used for seeding the random number generator. This file contains sensitive data and should read/write for the user and not accessible for others. This file is created the first time the program is run and updated automatically. The user should never need to read or modify this file.

$HOME/.ssh/identity
Contains the RSA authentication identity of the user. This file contains sensitive data and should be readable by the user but not accessible by others. It is possible to specify a passphrase when generating the key; the passphrase will be used to encrypt the sensitive part of this file using **IDEA**.

$HOME/.ssh/identity.pub
Contains the public key for authentication (public part of the identity file in human-readable form). The contents of this file should be added to *$HOME/.ssh/authorized_keys* on all machines where you wish to log in using RSA authentication. This file is not sensitive and can (but need not) be readable by anyone. This file is never used automatically and is not necessary; it is only provided for the convenience of the user.

$HOME/.ssh/config
This is the per-user configuration file. The format of this file is described above. This file is used by the **ssh** client. This file does not usually contain any sensitive information, but the recommended permissions are read/write for the user, and not accessible by others.

$HOME/.ssh/authorized_keys
Lists the RSA keys that can be used for logging in as this user. The format of this file is described in the **sshd** manual page. In the simplest form the format is the same as the .pub identity files (that is, each line contains the number of bits in modulus, public exponent, modulus, and comment fields, separated by spaces). This file is not highly sensitive, but the recommended permissions are read/write for the user, and not accessible by others.

/etc/ssh_known_hosts
Systemwide list of known host keys. This file should be prepared by the system administrator to contain the public host keys of all machines in the organization. This file should be world-readable. This file contains public keys, one per line, in the following format (fields separated by spaces): system name, number of bits in modulus, public exponent, modulus, and optional comment field. When different names are used for the same machine, all such names should be listed, separated by commas. The format is described on the **sshd** manual page.

The canonical system name (as returned by name servers) is used by **sshd** to verify the client host when logging in; other names are needed because **ssh** does not convert the user-supplied name to a canonical name before checking the key, because someone with access to the name servers would then be able to fool host authentication.

/etc/ssh_config
Systemwide configuration file. This file provides defaults for those values that are not specified in the user's configuration file, and for those users who do not have a configuration file. This file must be world-readable.

$HOME/.rhosts
This file is used in **.**rhosts authentication to list the host/user pairs that are permitted to log in. (Note that this file is also used by rlogin and rsh, which makes using this file insecure.) Each line of the file contains a host name (in the canonical form returned by name servers), and then a user name on that host, separated by a space. This file must be owned by the user, and must not have write permissions for anyone else. The recommended permission is read/write for the user, and not accessible by others.

Note that by default **sshd** will be installed so that it requires successful RSA host authentication before permitting **.**rhosts authentication. If your server machine does not have the client's host key in */etc/ssh_known_hosts*, you can store it in *$HOME/.ssh/known_hosts*. The easiest way to do this is to connect back to the client from the server machine using ssh; this will automatically add the host key in *$HOME/.ssh/known_hosts*.

$HOME/.shosts
This file is used exactly the same way as **.**rhosts. The purpose for having this file is to be able to use rhosts authentication with **ssh** without permitting login with rlogin or rsh.

/etc/hosts.equiv
This file is used during **.**rhosts authentication. It contains canonical hosts names, one per line (the full format is described on the **sshd** manual page). If the client host is found in this file, login is automatically permitted provided client and server user names are the same. Additionally, successful RSA host authentication is normally required. This file should only be writable by root.

/etc/shosts.equiv
This file is processed exactly as */etc/hosts.equiv*. This file may be useful to permit logins using **ssh** but not using rsh/rlogin.

/etc/sshrc
Commands in this file are executed by **ssh** when the user logs in just before the user's shell (or command) is started. See the **sshd** manual page for more information.

$HOME/.ssh/rc
Commands in this file are executed by **ssh** when the user logs in just before the user's shell (or command) is started. See the **sshd** manual page for more information.

INSTALLATION

Ssh is normally installed as suid root. It needs root privileges only for rhosts authentication (rhosts authentication requires that the connection must come from a privileged port, and allocating such a port requires root privileges). It also needs to be able to read */etc/ssh_host_key* to perform **RSA** host authentication. It is possible to use **ssh** without root privileges, but rhosts authentication will then be disabled. **Ssh** drops any extra privileges immediately after the connection to the remote host has been made.

Considerable work has been put into making **ssh** secure. However, if you find a security problem, please report it immediately to <ssh-bugs@cs.hut.fi>.

AUTHOR

Tatu Ylonen <ylo@ssh.fi>

Information about new releases, mailing lists, and other related issues can be found from the ssh WWW home page at http://www.cs.hut.fi/ssh.

SEE ALSO

sshd(8), **ssh-keygen**(1), **ssh-agent**(1), **ssh-add**(1), **scp**(1), **make-ssh-known-hosts**(1), **rlogin**(1), **rsh**(1), **telnet**(1)