Network Authorization [7]

This section describes security in the UNICOS TCP/IP environment. It focuses on actions users should take to prevent malicious intrusion into the system. The following topics are discussed:

- The autologin feature
- Authorization files
- Solving authorization problems

Using autologin through Kerberos is discussed in the *Kerberos User's Guide*, publication SG–2409.

The autologin feature

7.1

UNICOS supports *autologin*, a feature that lets a user log in automatically across the network to an account that belongs to that user or to another user. This feature might be restricted when the UNICOS multilevel security (MLS) feature is enabled. See subsection 8.2.4, page 104, for more information. Check with your security administrator. The following files support autologin:

- The `.rhosts` and `/etc/hosts.equiv` files for incoming `rlogin`(1B), `rsh` (see `remsh`(1B)), and `rcp`(1) commands (this applies only to TCP/IP networks)
- The `.netrc` file for the outgoing `ftp`(1B) command and the `rexec`(3) library routine (for TCP/IP)

Although autologin is very convenient, it does present a major security threat to the system, and the following precautions should be taken when creating the `.rhosts` and `.netrc` files.

Precautions for the `.rhosts` file:

- Ensure that only the owner can read or write to the file.
- Ensure that the file contains only those hosts that are needed.
- Do not use wildcard characters.

Precautions for the `.netrc` files:

- Ensure that only the owner can read or write to the file.
- Do not put passwords in the `.netrc` file. The password parameter is for facilitating anonymous `ftp` and the `rexec`(3) library routine. If you need more information on anonymous `ftp`, contact your system administrator.

For information on setting up the `.rhosts` and `.netrc` files, see the following subsection.

Authorization files 7.2

Authorization files contain host and user information that is verified by the system before user privileges are granted on a remote system. These files can be used to ensure system security by limiting access to directories or to the UNICOS system. You can create the `.rhosts` and `.netrc` files; the system administrator can create the `/etc/hosts.equiv`, `/etc/ftpusers`, and `/etc/shells` files. In a UNICOS MLS system, the system administrator has additional control with the network access list (NAL) and workstation access list (WAL) in the `/etc/config/spnet.conf` file. The TCP/IP `telnet` utility does not use authorization files. The following list describes the authorization files:

File name	Description
`/etc/ftpusers`	System file that lists users who are prohibited from accessing a system by using the `ftp` program. If the system administrator has not set up an `ftpusers` file, or the file is empty, all valid UNICOS users can use `ftp`.
`/etc/hosts.equiv`	System file that lists equivalent host and alternative user names used during autologin. Direction is inbound. Associated commands are `rlogin`, `rcp`, and `rsh`.
`/etc/shells`	System file that lists valid login shells for inbound `ftp` users. If a UNICOS user's login shell is not in this file, the user is denied `ftp` access.

File name	Description
`$HOME/.netrc`	User file that lists autologin information for `ftp` and `rexec`(3) requests. Direction is outbound.
`$HOME/.rhosts`	User file that lists remote host names and login names of users who are allowed access to your home directory. Used during autologin. Direction is inbound. Associated commands are `rlogin`, `rcp`, and `rsh`.
`/etc/config/spnet.conf` (network access list)	On a UNICOS MLS system, the NAL defines the label range allowed for each connection to and from a remote host. This might prevent a user from connecting to or from specific remote hosts, or might prevent access at specific labels. The NAL also defines the level of trust for each node in a Trusted UNICOS system. (The NAL class must be C1 or higher to allow automatic authorization from that node. The class must be B1 or higher to allow connections at more than one label. See subsection 8.1.2 page 92, for more information.
`/etc/config/spnet.conf` (workstation access list)	On a UNICOS MLS system, the WAL defines the remote services allowed for specified users on specified remote nodes. This might prevent a remote copy, or remote login, for example. See subsection 8.1.5, page 95, for more information.

The `.rhosts` *and* `/etc/hosts.equiv` *files* 7.2.1

The `.rhosts` file in your home directory and the `/etc/hosts.equiv` file that the system administrator maintains provide authorization for the `rlogin`, `rcp`, and `rsh` utilities. For both files, authorization occurs in much the same

manner: both files on the destination host must contain an entry (or entries) authorizing connections from the desired source host; a similar mechanism permits connections from a specific source host to be explicitly denied authorization.

Note: When the UNICOS MLS feature is enabled, the autologin capability of inbound `rlogin`, `rcp`, and `rsh` might be restricted. See subsections 8.2.4, page 104; 8.2.5, page 106; and 8.2.7, page 112, for more information, and check with your security administrator.

Authorizing connections from remote hosts by using `.rhosts` 7.2.1.1

To authorize `rlogin`, `rcp`, and `rsh` connections to your Cray Research account from remote hosts, create a `.rhosts` file in the home directory of your account on the Cray Research system and place in it entries that authorize connections from specific account names on specific remote hosts. The `.rhosts` file is a simple text file; you can use any standard UNICOS text editor (for example, `vi`) to create or modify it.

Each entry in the `.rhosts` file is a separate line of text with the following format:

```
host account optional_comment
```

Such an entry in your `.rhosts` file in the home directory of your account on the local Cray Research host authorizes the user with the account name of *account* on the remote host with name *host* to access your account on the local Cray Research system. Some implementations require *host* to be a fully qualified domain name. To operate the host and account name parts of each entry, the account name part and the optional comment, use either a space or a tab character.

If you omit the account name part of an entry in your `.rhosts` file, the authorization mechanism assumes your local account name in its place (for example, if your account name on the local Cray Research host is `andrea`, and you place an entry in your `.rhosts` file with just the host name `other`, it will authorize the identically named account name `andrea` on host `other`). You can explicitly deny authorization to all accounts on a specific system by placing a hyphen (–) in the account name part of an entry for that system, as in the following example:

host –

This specification denies authorization to any connection from the remote host called *host*. Similarly, you can explicitly deny authorization to a specific account name on all systems by placing a hyphen in the host name part of an entry for that user, as in the following example:

– user

This specification denies authorization to any account called *user*, no matter from which remote host the connection originates.

Multiple names are not permitted in the host or in the account parts of a single entry. To authorize an account name from more than one remote host, or several accounts from a single remote host, simply create multiple entries in the `.rhosts` file, as in the following `.rhosts` file example:

```
# .rhosts file
#
host1.cray.com jen  # User jen can access the local host from host1.cray.com
host2 jen   # User jen can access the local host from host2
host3 –     # No accounts from host3 can access the local host
 – ted      # User ted cannot access the local host
```

Note: Although the convenience of the `.rhosts` feature makes it tempting to include many host names in the `.rhosts` file (for example, when a connection from an otherwise seldom-used remote host is desired), serious security considerations stem from indiscriminate listing of host names in `.rhosts` files. For guidelines on how to include host names in your `.rhosts` file in a secure manner, see subsection 7.1, page 77.

Authorizing connections from remote hosts by using `/etc/host.equiv` 7.2.1.2

The system administrator of your local Cray Research host can place the names of various remote hosts in the `/etc/hosts.equiv` file. Hosts listed in this file are considered to be equivalent to the Cray Research host; any account name that is identical on the two hosts is authorized automatically for connection from the remote host to the Cray Research host through the `rlogin`, `rcp`, and `rsh` utilities. Conversely, the system administrator can specifically deny authorization to connections from certain remote hosts by placing a hyphen beside entries in the `/etc/hosts.equiv` file.

These security measures are in effect regardless of any entries in `.rhosts` files; that is, entries in the `/etc/hosts.equiv` file override individual users' `.rhosts` files. Any host listed in the `/etc/hosts.equiv` file will have its `rlogin`, `rcp`, and `rsh` connections to the Cray Research host automatically authorized, even if you try to restrict connections from such a remote host by placing a – entry for it in your `.rhosts` file. Similarly, any host with a – entry in the `/etc/hosts.equiv` file will have authorization of its `rlogin`, `rcp`, and `rsh` connections denied automatically, even if you try to authorize connections from such a remote host by placing its name in your `.rhosts` file.

Because the `/etc/hosts.equiv` file is a text file (like the `.rhosts` file), you can use any standard UNICOS utility (such as `vi`, `ed`, or `cat`) to determine whether an entry in `/etc/hosts.equiv` is impeding your attempts to permit or deny authorization by way of your `.rhosts` file. Following is a sample `/etc/hosts.equiv` file (which overrides the sample `.rhosts` file in the previous example):

```
# hosts.equiv
#
host1 ted    # Allows user ted to access the local host from host1
host2 – jen  # Allows all host2 users except jen to access local host
host3        # Allows all users on host3 to access local host
```

Note: When the UNICOS MLS feature is enabled and the configuration parameter `NETW_RCMD_COMPAT` is not set, special restrictions apply. You may have to place entries in the `/$HOME/.rhosts` file even when the `/etc/hosts.equiv` file has entries for corresponding hosts. See subsection 8.2.4, page 104, for more information.

Authorizing connections from remote hosts by using `.rhosts` 7.2.1.3

When trying to connect to a remote host by using the `rlogin`, `rcp`, or `rsh` utility, authorization is handled in a manner specific to the remote host. Therefore, you should consult with the system administrator of the remote host, or see the remote host's vendor documentation, for the correct information about authorization on that host. Nevertheless, in practice, if the remote host is running an operating system derived from the 4.3BSD operating system, you can probably authorize a connection to that host from your local Cray Research host by connecting and logging in to the remote host (using, for example, `telnet`) and then using any available text editor to enter the

name of your local Cray Research host in the `.rhosts` file in your home directory on the remote host. If this does not appear to work, consult with the system administrator or see the vendor-supplied documentation for the remote host.

The following example illustrates a sample `.rhosts` file on a Cray Research host for a user with a login name of `scott`. `scott` wants to authorize connections from his accounts with the same login name on another Cray Research host (`othercray`) and from the front-end workstation (`biology`). `scott` also is working on a project with a user with a login name of `betsy` on another workstation (`math`), and he wants to authorize connections from that account on that workstation. Finally, he wants to deny explicitly authorization to any account from the host called `chemistry` (possibly because the account called `scott` on that system belongs to a different user), and to the account called `trouble` on any host (possibly because the user of that account name is a known security risk). A `.rhosts` file to set up these authorizations might contain the following entries:

```
# sample .rhosts file on a Cray Research system
#
othercray scott # my login (scott) from our other Cray Research system
biology    # my login (scott) from my workstation
math  betsy     # while we're working on XYZ project
chemistry -     # no one from host chemistry
- trouble  # no one named "trouble"
```

The authorization lines for `othercray` and `biology` show that `scott` can authorize the identical login name on another system either explicitly by listing the login name or simply by omitting the login name part of the entry.

The `.netrc` *file* 7.2.2

You can create the `.netrc` file in your home directory on the Cray Research system to provide authorization for the `ftp` facility and `rexec`(3) library routine. When you invoke either one, the program looks for a `.netrc` file in your home directory. If the program finds this file, it uses the information contained in the file to log you in automatically to the remote host. If you do not have a `.netrc` file, you will be prompted for your login name and password.

The `.netrc` file is a simple text file; you can use any standard UNICOS text editor, such as `vi`, to create or modify it. If `.netrc` contains password or account information, `ftp` will use the file only if the file permissions are set so that only the owner of the file has read and write permissions. That means that the owner should use `chmod`(1) to set the file permissions to `600`. If the file permissions allow any other user to read and write the file, `ftp` will fail. For information on security concerns for the `.netrc` file, see subsection 7.1, page 77.

The `.netrc` file can contain one or more entries. Each entry describes default values and macros to use when connecting to a specified remote host. Each entry is made up of token pairs, which includes a keyword and a value. Each token is a string of characters, separated by a space, tab, comma, newline, or a string of characters between two double quotation marks. The backslash (\) is a special character. You can embed any of the special characters (space, tab, comma, newline, double quotation mark, or backslash) into a token by preceding it with a backslash. Usually, each entry is on a separate line.

Permissible token pairs 7.2.2.1

The recognized keywords are `machine`, `login`, `password`, `account`, and `macdef`.

A list of the known token pairs follows. The `machine` *remote_hostname* token pair defines the start of an entry. All other token pairs are optional and can be specified in any order, though they are usually given in the order that follows. You will be prompted for any information that is missing from the `.netrc` file (for example, the `password` *password* token pair) and is needed to establish a connection. The `macdef` token pair differs from the other token pairs; after the `macdef` *macro_name* token pair, all characters up to a blank line are assumed to be the definition of the macro.

Token	Description
`machine` *remote_hostname*	Identifies the name of the remote host to which a connection is to be established. The `.netrc` file is searched for a `machine` token that matches the remote host name specified on the `ftp` or `rexec` command line or as an `open` command argument. After a match is found, the subsequent `.netrc` tokens are processed until the end of the file is reached or until another `machine` token is found.
`login` *login_name*	Specifies the name of a user at the remote host. If this token is present, the autologin process logs in to the remote host by using the specified name.
`password` *password*	Specifies a password. If this token is present, the autologin process supplies the specified string when the file transfer server requires a password as part of the login process. If the `.netrc` file can be read by anyone other than the user, and this token is present in the file, the autologin process is aborted. For security purposes, clear-text passwords should not be used.
`account` *account_name*	Supplies an additional account password. If this token is present, the autologin process supplies the specified string if the file transfer server requires an additional account password.

Token	Description
`macdef` *macro_name macro*	Defines a macro for use in the `ftp` session. This token is similar to the `macdef` command of `ftp`. A macro is defined with the specified name; its contents begin with the next `.netrc` line and continue until a blank line is encountered. If a macro called `init` is defined, it is executed automatically as the last step of the autologin process.

Example of a `.netrc` *file* 7.2.2.2

This subsection shows an example of a `.netrc` file. This example contains a set of tokens for three different remote hosts. The first set indicates that, when connecting to host `biology`, you must use the login name `bonnie`. Because the password was omitted, you are prompted for the password during each login process. The second set indicates that, when connecting to the host `chemistry`, you must use the login name `bonnie2`, and it also defines two macros, `lsf` and `pwdlsf`. The third set is an entry for anonymous `ftp`. The anonymous facility is the ability to use `ftp` to access another host without having an account or password on that host. The login name for anonymous `ftp` is usually `anonymous`. The password should be a name that describes the user; however, for security purposes, you should not use clear-text passwords in `.netrc`. To identify the user, this entry contains the password `bonnie`. Usually, the anonymous facility is not enabled; when it is enabled, only a limited number of files can be accessed on that host.

```
# .netrc file example

machine biology login bonnie
machine chemistry login bonnie2
    macdef lsf
    ls -CF

    macdef pwdlsf
    pwd
    ls -CF
machine blackhole login anonymous password bonnie
```

The `/etc/shells` *and* `/etc/ftpusers` *files* 7.2.3

The system administrator maintains the `/etc/shells` file to determine what command shells can be used to access the UNICOS system. The system administrator maintains the `/etc/ftpusers` file to determine who can use `ftp` to access the UNICOS system. The `/etc/shells` file is an ASCII file that contains valid login shells; the `/etc/ftpusers` file is an ASCII file that contains login names that are not valid, one user name per line. When the `ftp` daemon is invoked, it makes the following checks for the login name of the user who is trying to gain access:

1. Determines whether the login shell of the user is listed in `/etc/shells`.
2. If the `/etc/shells` file does not exist, the daemon uses a default list of `/bin/sh`, `/bin/csh`, and `/bin/ksh`.
3. If the user's login shell is not listed, the user is denied access.
4. Checks the `/etc/ftpusers` file for the login name of the user who is trying to gain access.
5. If the name is there, `ftp` denies the user access.
6. If `/etc/ftpusers` is nonexistent or empty, all valid UNICOS users are considered valid users of `ftp`.

Following is an example of an `/etc/shells` file:

```
# /etc/shells file example
#
# Valid login shells for FTP users
#
/bin/csh
/bin/sh
/bin/ksh
/usr/lbin/oursh    # A local shell
```

An example of an `/etc/ftpusers` file follows:

```
# /etc/ftpusers file example
#
# Denied ftp and ftam users
deb
mk
adam
```

Solving authorization problems
7.3

Table 4 gives examples of possible autologin problems and solutions.

Table 4. Authorization problems and solutions

Problem	Solution
You cannot achieve autologin to a remote host.	See the remote host's vendor documentation for the proper authorization procedures on the remote host. (If the remote host is running an operating system based on 4.3BSD, the solution may be as simple as putting an entry that contains the name of the local host in the `.rhosts` file in your home directory on the remote host.)
You cannot achieve autologin to a remote host because your login name on the local host is different from the login name on the remote host.	See the remote host's vendor documentation for the proper remote host procedures to authorize an autologin from a different login name. (If the remote host is running an operating system based on 4.3BSD, the solution may be as simple as putting an entry that contains the local host name and your account name in the `.rhosts` file that is in your home directory on the remote host, and then using the `-l` *login_name* option on the command line whenever you use the `rlogin` or `rsh` utility.) On a UNICOS MLS system, you may be prohibited from using autologin for a different account name than your local account name. Check with your system administrator and see subsection 8.2.4, page 104, for information about restrictions.
You cannot achieve autologin to a Cray Research host.	Place an entry containing the name of the host from which you are trying to perform the autologin in the `.rhosts` file in your home directory on the Cray Research host. Ensure that only the owner has write permission. Check with your system administrator about whether a UNICOS MLS system is running, and whether autologin is restricted. You may need to place an entry for your local host in the `/etc/hosts.equiv` file, and you may have to use the same account name on both local and remote systems.

Table 4. Authorization problems and solutions
(continued)

Problem	Solution
You want to authorize specific users on certain remote hosts to access your account on a Cray Research host.	Place entries in your `.rhosts` file in your home directory on the Cray Research host, listing every remote host and user you want to authorize for autologin access to your account. Such authorized users must then use the `-l` *login_name* option (*login_ name* represents your account name on the Cray Research host) on the command line whenever using the `rlogin` or `rsh` utility to access your account. Check with your system administrator about whether a UNICOS MLS system is running, and whether autologin is restricted. You may be prohibited from using different account names between the local and remote systems.
You want to forbid users on certain remote hosts access to your account on a Cray Research host.	Place entries in your `.rhosts` file in your home directory on the Cray Research host, listing every remote host from which you want to forbid access to your Cray Research account. Follow each entry with a –. (This designation does not deny access to a user whose account name on a remote host matches your account name on the Cray Research host, and/or whose remote host name the system administrator has placed in the `/etc/hosts.equiv` file on the Cray Research host; the entry in `/etc/hosts.equiv` overrides the entry in your `.rhosts` file.) If you are on a UNICOS MLS system, you might be able to prevent access to your account even when entries to the `/etc/hosts.equiv` file exist. See your system administrator for more information.