NAME

auth_open, auth_close, auth_recv, auth_send – auth server interface

SYNOPSIS

#include "firewall.h"

int auth_open(confp)
Cfg *confp;

int auth_close()

int auth_send(s)
char *s;

int auth_recv(s, siz)
char *s;
int siz;

DESCRIPTION

These functions facilitate communication with a third party authentication server **authsrv.** All communication with the server is in terms of newline-terminated text, with whitespace indicating token delimiters. Whitespace characters may be embedded in tokens by quoting them with matched single or double quotes.

Authenticating a user via the server is a 5 step process, consisting of opening a connection to the server, telling the server the user-id of the user who is being authenticated, getting an appropriate challenge from the server and presenting it to the user, getting the user's response and returning it to the server, and closing the connection. An annotated example in which a user is authenticated is presented below. (Error checking is removed)

```
#include <stdio.h>

#include "firewall.h"

main()
{
   char       cbuf[128];
   char       ubuf[128], buf[128];
   int        bad = 5;

    auth_open((Cfg *)0);

    /* get welcome message from auth server */
    if(auth_recv(buf,sizeof(buf))) {
       fprintf(stderr,"lost connection to server0);
       exit(1);
    }
    if(strncmp(buf,"Authsrv ready",13)) {
       fprintf(stderr,"%s\n",buf);
       auth_close();
       exit(1);
    }

    while(bad--) {
       fprintf(stderr,"User-Id: ");
       fgets(ubuf,sizeof(ubuf),stdin);

       sprintf(cbuf,"authorize %s myapplication",ubuf);
       auth_send(cbuf);
       auth_recv(cbuf,sizeof(cbuf));

```

```
33          if(!strncmp(cbuf,"challenge ",10)) {
34             fprintf(stderr,"Challenge \"%s\": ",&cbuf[10]);
35             gets(ubuf);
36          } else
37          if(!strncmp(cbuf,"password",8)) {
38             strcpy(ubuf,getpass("Password: "));
39          } else {
40             fprintf(stderr,"%s\n",cbuf);
41             continue;
42          }
43
44          sprintf(cbuf,"response '%s'",ubuf);
45          auth_send(cbuf);
46          auth_recv(cbuf,sizeof(cbuf));
47
48          if(strncmp(cbuf,"ok",2)) {
49             printf("%s\n",cbuf);
50             continue;
51          }
52          auth_close();
53          if(cbuf[2] != '\0')
54             printf("OK: %s\n",&cbuf[3]);
55          else
56             printf("OK\n");
57             exit(0);
58       }
56       auth_close();
59       exit(1);
60}
```

At line #11 in the example, a connection to the authentication server is opened. This connection will persist until explicitly closed with a call to **auth_close.** If multiple calls to **auth_open** are made without matching calls to **auth_close** the connection is preserved and is not reset. Normally **auth_open** is invoked with a pointer to the configuration record of the application, read with **cfg_read.** If using a client library with a compiled-in address for the authentication server, a null pointer may be passed instead.

At line #14, a welcome string is read from the server. If the calling application has permission to use the server and the server is operating normally, the first 13 characters of the first line it sends will read "Authsrv ready" In the event that they are not, the message will be an error message that should be presented to the user.

At line #28, the user-id that is to be authenticated is embedded in an authentication request that is sent to the server via **auth_send.** The server's response is read using **auth_recv.** Authentication requests take the form of: *authorize username.* Whenever a new authentication request is send, the server will reset its current notion of the authenticated user. The additional parameter "myapplication" serves as a comment for the authentication manager, which is attached to the authentication request solely for logging purposes.

At line #33, the server's response is examined. The server will return one of three forms of responses, either:
challenge challenge-string
password
other text
If the client application sees a response of "password" it should prompt the user for a non-echoing password. If the client application sees a response of "challenge" it should issue the remainder of the line to the user as a challenge. In the event of receiving any text other than "password" or "challenge" the text is treated as an error message that may be presented to the user.

At line #44, the user's response to the challenge/password request is sent to the user as a response message. Response messages take the form of: *response response-string* where the response string should usually be quoted in case white space is contained within it.

At line #48, the server's response is examined. If the response begins with "ok" then the server has authenticated the user. Any other response indicates an error. The response may contain additional information from the server and should be displayed to the user. If the response begins with "ok" but the third character is not a null string terminator, then the server has returned additional response information that should be displayed to the user.

RETURN VALUES

All routines return zero to indicate success. Nonzero return values indicate a fatal error, usually a lost network connection.

SEE ALSO

authsrv(8)

NAME

authmgr − network user authentication client

SYNOPSIS

authmgr

DESCRIPTION

The Firewall Toolkit user authentication client (authsrv) is a client-side interface to the FWTK user authentication daemon (**authmgr**). It provides a convenient, command line interface for creating, modifying, disabling, and deleting users. See **authsrv**(8) for more information on the authentication management system.

Configuration Options

authmgr reads configuration rules from the *netperm-table*. It reads all rules using the **authmgr** and ***** **authmgr** reads the *netperm-table* from top to bottom. If there are multiple rules in the table that could apply for a particular attribute, **authmgr** uses the first one that it finds. See **netperm-table**(5) for a more complete explanation of *netperm-table* syntax and precedence.

authmgr recognizes the following attributes:

authserver *host port*

Specifies the host running the authentication server (and the port on which it runs) that the proxies use for authenticating users.

host Specifies an IP address or host name.

port Specifies a service name or port number.

COMMANDS

authmgr recognizes the following commands from within its command line interface:

adduser *user*[*fullname*]

Adds a user to the authentication database. You must be a global administrator or a group administrator to use this command. If you are a group administrator, users you create are the automatically made members of your group. Note that you still must enable a user after you have added the user.

user Name of the user to be added to the authentication database. The user names must match the user names for the strong authentication system you are using. The user names in the user authentication management system do not generally need to match user names on the firewall itself. If you are creating a record for a use with the *login-sh* authentication wrapper program, the user name must match the standard UNIX user information (in */etc/passwd*).

fullname

The user's full name or other descriptive text about the user. Use quotes for information that contains spaces.

deluser *user*

Deletes the specified user from the authentication database. You must be a global administrator a group administrator to use this command. If you are a group administrator, the user must be a member of your group.

disable *user*

Disable the specified user from the authentication database. The user's information remains in the database, but does not allow the user to use the system. The user authentication system disables users after a set number (configurable by the administrator) of failed login attempts. You must be the database administrator or a group administrator to use this command. If you are a group administrator, the user must be a member of your group.

display *user*

Displays the status, authentication protocol, and last login of the specified user. You must be the database administrator or a group administrator to use this command. If you are a group administrator, the user must be a member of your group.

enable *user* **[onetime]**
Allows users who have been disabled to use the system again. You must be the database administrator or a group administrator to use this command. If you are a group administrator, the user must be a member of your group.

user Name of a user.

onetime
Enables the user for one successful authentication. After the user successfully authenticates once, **authmgr** automatically disables the user.

exit Exits **authmgr**.

group *user group*
Makes the user a member of the specified group. You must be a global administrator to use this command.

help Displays a list of valid commands for **authmgr**.

list [*group***]**

password [*user***]** *passwordtext*
Sets the password for the current user. You must be the database administrator or a group administrator to use this command. If you are a group administrator, the user must be a member of your group.

user Name of a user. If you specify a user name, changes the password for the specified user.

passwordtext
Password for the current or specified user. Valid values depend on the protocol being used. For the default protocols, valid values are:

SecurID
Nothing

SNK Eight 3-digit groups used to key the SNK calculator for this user

S/Key The seed for the user's password

password
A reusable password. Use quotes if the password contains spaces.

proto *user protocol*
Sets the authentication protocol for the specified user to the named protocol. You must be the database administrator or a group administrator to use this command. If you are a group administrator, the user must be a member of your group.

user Name of a user.

protocol
The name of the authentication protocol. Available protocols depends on the protocols compiled into **authmgr**. By default, valid values are:

- SecurID
- snk
- skey
- password

quit Exits **authmgr**.

rename *user newname* **[** *fullname***]**
Changes the existing record for the existing user with the specified information.

user Name of an existing user in the authentication database.

newname
The new user name that you wish to use.

fullname
The new user's full name or other descriptive text about the user. Use quotes for information that contains spaces.

unwiz *user*
Removes the group administrator flag from the specified user. You must be a global administrator to use this command. This command has no effect on global administrators. To remove global administrator privileges from a user, delete and recreate the user.

wiz *user*
Makes the specified user a group administrator of the group of which they are currently a member.

? Displays a list of valid commands for **authmgr**.

EXAMPLES

This example shows the configuration lines in the *netperm-table* that indicate that the authentication client accesses the authentication server running on the same machine on port 7777:

```
authmgr:    authserver 127.0.0.1 7777
```

This example shows an administrator running as **root** starting **authmgr** on the firewall, logging in as an administrative user, and creating and enabling a new user in the authentication database.

```
root@firewall # authmgr
Connected to server
authmgr-> login fwadmin
Password:                password does not display
Logged in
authmgr-> list
Report for users in database
user     group    longname     status proto    last
----     -----    --------     ----- -----     ----
fwadmin                        y W  Skey       Thu May  2 11:41:42 1996
scooter           Scooter Lindley y    Snk     Wed May  1 09:02:12 1996
authmgr-> adduser john 'John Whorfin'
ok - user added initially disabled
authmgr-> enable john
enabled
authmgr-> proto john Snk
changed
authmgr-> pass john '160 270 203 065 022 034 232 162'
Secret key changed
authmgr-> list
Report for users in database
user     group    longname     status proto    last
----     -----    --------     ----- -----     ----
fwadmin                        y W  Skey       Thu May  2 11:41:42 1996
scooter           Scooter Lindley y    Snk     Wed May  1 09:02:12 1996
john             John Whorfin    ena  Snk      never
authmgr-> quit
#
```

FILES

/usr/local/etc/netperm-table
: The network permissions file contains configuration information for the Firewall Toolkit, including **authmgr**.

BUGS

Report bugs to the fwtk-users mailing list. Include a complete example, explaining what you expected to happen and what actually happened. Be sure to indicate the type of system (operating system, hardware, etc.) you are using, as well as the version of authmgr**.**

AUTHOR

Trusted Information Systems, Inc.

SEE ALSO

netperm-table(5), **rc**(8), **authsrv**(8)

NAME

authsrv – network user authentication daemon for third-party systems

SYNOPSIS

authsrv [-daemon *port***] [-version]**

DESCRIPTION

The Firewall Toolkit user authentication daemon (**authsrv**) acts as a piece of "middleware" to provide a unified interface for several strong authentication systems and the firewall.

The Firewall Toolkit user authentication management system allows you to easily integrate several different strong authentication systems into your general firewall administration. From within **authsrv** you can create, modify, disable, delete, and examine users. **authsrv** maintains a database for this information.

The Firewall Toolkit supports a variety of strong authentication options. The authentication management system understands the types of passwords that these systems use, and provides a consistent user interface to these systems. Currently supported systems are:

SecurID
This system, available from Security Dynamics, uses a time-based password.

Secure Net Key (SNK)
This system, from Digital Pathways, uses a random challenge password.

S/Key This system, from Bellcore, uses a one-time password.

Reusable Passwords
This system, a part of the user authentication system included with the Firewall Toolkit, is a reusable password option.

The various proxies use the information in the user authentication management system any time you have configured the proxies to require authentication. Using the default Firewall Toolkit configuration, this occurs any time a user from an untrusted network tries to access a service inside the perimeter. Recall that untrusted networks are those from which the firewall will accept requests only after authentication by the user.

Users

User names you create in the user authentication management system are used only for strong authentication. The user names must match the user names for the strong authentication system you are using.

The user names in the user authentication management system do not generally need to match user names on the firewall itself. By default, you do not create any user accounts on the firewall. The exception to this rule is the *login-sh* authentication wrapper program. The *login-sh* program authenticates users before logging them into the firewall. Then, the information in the user authentication management system must match the standard UNIX user information (in */etc/passwd*) for these users.

The user names in the user authentication management system do not need to match any user names on your internal network. For example, John Whorfin might use **john** as his user name on internal networks. He could use **whorfin** for strong authentication at the firewall. You may wish to use the same names for the convenience of your users.

Groups

The Firewall Toolkit user authentication management system also makes use of groups. Groups allow you to permit or deny services based on groups of user names, rather than individual user names. For example, you can configure the telnet proxy to permit service to everyone in group **sales**.

Just as is the case with user names, the groups that you create in the Firewall Toolkit user authentication management system are not the same as the groups you create on the firewall or on the internal network. You can of course use the same names, for easier administration.

Administrators

In addition, the user authentication management system also supports the notion of administrators within the system. Global administrators can create both groups and users. Initially, **root** is the only global administrator. Creating another global administrator is part of the configuration process for the user authentication system. Within each group, you can also designate group administrators who can create and modify users only within their own group.

As with general UNIX system administration, you have the ability to create a hierarchical system in which you have multiple groups and group administrators. You can also create a monolithic system in which there are multiple administrators and few group administrators.

OPTIONS

Command Line Options

authsrv recognizes the following command line options (whether started from the command line or from within */etc/rc.local*):

-daemon *port*

Indicates that **authsrv** runs as a daemon, and the port (name or number) on which **authsrv** listens.

Configuration Options

authsrv reads configuration rules from the */usr/local/etc/netperm-table*. It reads all rules using the **authsrv** and * (wildcard) keywords. **authsrv** reads the *netperm-table* from top to bottom. If there are multiple rules in the table that could apply for a particular attribute, **authsrv** uses the first one that it finds. See **netperm-table**(5) for a more complete explanation of *netperm-table* syntax and precedence.

authsrv recognizes the following attributes:

badsleep *seconds*

Specifies the amount of time the authentication server disallows logins from a user who has attempted (and failed) to login five times in a row.

seconds

Specifies the number of seconds the authentication server sleeps before allowing login attempts from a user who has attempted (and failed) to login five times in a row. If this option is set to 0, the authentication server allows an unlimited number of unsuccessful login attempts. If this option is not set, the authentication server disables the account after the user attempts (and fails) to login five times in a row.

database *path*

Specifies the pathname of the database that the authentication server uses. This option is mandatory, unless you compile the authentication server with a specific database path.

directory *directory*

Specifies the directory that **authsrv** makes its root directory before providing service.

groupid *group*

Specifies the name of the group **authsrv** uses when running.

group Specifies either a name or numeric id from the */etc/group* file.

permit-hosts *hosts*

Specifies single hosts, entire networks, or subnets. Specify by IP address or host name. The wildcard * is valid.

deny-hosts *hosts*

Specifies the hosts that cannot use **authsrv**.

hosts Specifies single hosts, entire networks, or subnets. Specify by IP address or host name. The wildcard * is valid.

nobogus true

Specifies that the authentication server indicates when a user name does not exist when users attempt to login and fail.

If this option is not specified and a user enters a non-existent user name, the authentication server always responds with a bogus SNK challenge.

Note that you must remove or comment out this setting if you wish to disable it. The settings

nobogus false and **nobogus off** are not valid.

{permit | deny}-operation {user *users* **| group** *groups***}** *service destination* **[***options***] [time** *start end* **]**
Specifies explicitly permitted or denied operations for particular users or groups at particular times of day. Note that the authentication server only uses these rules when the proxy uses the **–extnd** attribute.

user *users*
Specifies the names of users for which the proxies use this rule. The wildcard * is valid.

group *groups*
Specifies the names of groups for which the proxies use this rule. The wildcard * is valid.

service Specifies the name of a service for which this rule applies. Valid values are:

- ftp-gw FTP proxy
- rlogin-gw Rlogin proxy
- telnet-gw TELNET proxy
- * all of these proxies

destination
Specifies the hosts to which the proxies can or cannot send requests. Specify individual machines, entire networks, or subnets. Use IP addresses or host names. The wildcard * is valid.

options Specifies particular operations for each protocol that can be controlled. Valid values are:

- ftp-gw Consult the **ftpd**(1) man page
- rlogin-gw none
- tn-gw none

time Indicates that this rule goes into effect and stops having an effect at a particular time.

start Specifies the time at which the proxy begins using this rule. Specify time in hours and minutes (between 00:00 and 23:59).

end Specifies the time at which the proxy stops using this rule. Specify time in hours and minutes (between 00:00 and 23:59).

securidhost *firewall*
Specifies the name of the firewall that is registered as the client host name on the ACE server. Because the firewall may have different host names, this allows you to specify which host name to use.

firewall Specifies the name of the firewall that is registered as the client host name on the ACE Server. Specify an individual machine. Use an IP addresses or host name.

timeout *seconds*
Specifies the number of seconds **authsrv** is idle (with no network activity) before disconnecting

permit-unknown *names*
Specifies a list of additional names that **authsrv** checks (in addition to the authentication database) when checking for extended permissions on a per user basis. If the user name is not in the authentication database, or in the list of names, **authsrv** logs the attempt and indicates that the user is not valid. If the user name is found in the list of names, **authsrv** assigns the user name to the group **unknown**.

names Specifies a list of names, separated by spaces. The wildcard * is valid.

userid *user*

Specifies the user ID **authsrv** uses when running.

user Specifies either a name or numeric id from the */etc/passwd* file.

COMMANDS

authsrv recognizes the following commands from within its command line interface:

adduser *user*[*fullname*]

Adds a user to the authentication database. You must be a global administrator or a group administrator to use this command. If you are a group administrator, users you create are the automatically made members of your group. Note that you still must enable a user after you have added the user.

user Name of the user to be added to the authentication database. The user names must match the user names for the strong authentication system you are using. The user names in the user authentication management system do not generally need to match user names on the firewall itself. If you are creating a record for a use with the *login-sh* authentication wrapper program, the user name must match the standard UNIX user information (in */etc/passwd*).

fullname

The user's full name or other descriptive text about the user. Use quotes for information that contains spaces.

authenticate *user*
authorize *user*

Performs the first part of authenticating as the specified user with the authentication database. You must use the **response** command to finish authenticating. After successfully authenticating as the specified user, you operate as that user, with all their privileges.

deluser *user*

Deletes the specified user from the authentication database. You must be a global administrator a group administrator to use this command. If you are a group administrator, the user must be a member of your group.

disable *user*

Disable the specified user from the authentication database. The user's information remains in the database, but does not allow the user to use the system. The user authentication system disables users after a set number (configurable by the administrator) of failed login attempts. You must be the database administrator or a group administrator to use this command. If you are a group administrator, the user must be a member of your group.

display *user*

Displays the status, authentication protocol, and last login of the specified user. You must be the database administrator or a group administrator to use this command. If you are a group administrator, the user must be a member of your group.

enable *user* **[onetime]**

Allows users who have been disabled to use the system again. You must be the database administrator or a group administrator to use this command. If you are a group administrator, the user must be a member of your group.

user Name of a user.

onetime

Enables the user for one successful authentication. After the user successfully authenticates once, **authsrv** automatically disables the user.

exit Exits **authsrv**.

group *user group*
Makes the user a member of the specified group. You must be a global administrator to use this command.

help Displays a list of valid commands for **authsrv**.

list [*group***]**
ls [*group***]**
Lists all users in the system, or the members of the specified group. Group administrators may list their own groups, but not the entire database.

operation {user *users* **| group** *groups***}** *service* **[***destination***] [***options***] [time** *start end* **]**
Determines whether there are any matching per user restrictions specified for this user in the *netperm-table*. All options match those specified in the *netperm-table*.

passwd [*user***]** *passwordtext*
password [*user***]** *passwordtext*
Sets the password for the current user. You must be the database administrator or a group administrator to use this command. If you are a group administrator, the user must be a member of your group.

user Name of a user. If you specify a user name, changes the password for the specified user.

passwordtext
Password for the current or specified user. Valid values depend on the protocol being used. For the default protocols, valid values are:

SecurID
Nothing

SNK Eight 3-digit groups used to key the SNK calculator for this user

S/Key The seed for the user's password

password
A reusable password. Use quotes if the password contains spaces.

proto *user protocol*
Sets the authentication protocol for the specified user to the named protocol. You must be the database administrator or a group administrator to use this command. If you are a group administrator, the user must be a member of your group.

user Name of a user.

protocol
The name of the authentication protocol. Available protocols depends on the protocols compiled into **authsrv**. By default, valid values are:

- SecurID
- snk
- skey
- password

quit Exits **authsrv**.

rename *user newname* **[** *fullname***]**
Changes the existing record for the existing user with the specified information.

user Name of an existing user in the authentication database.

newname
The new user name that you wish to use.

fullname
The new user's full name or other descriptive text about the user. Use quotes for information that contains spaces.

response *text*
Allows you to enter the appropriate password or token to finish authenticating as the specified user with the authentication database. Use quotes for information that contains spaces.

superwiz *user*
Makes the specified user a global administrator.

unwiz *user*
Removes the group administrator flag from the specified user. You must be a global administrator to use this command. This command has no effect on global administrators. To remove global administrator privileges from a user, delete and recreate the user.

wiz *user*
Makes the specified user a group administrator of the group of which they are currently a member.

? Displays a list of valid commands for **authsrv**.

EXAMPLES

This example shows the configuration lines in the *netperm-table* that the authentication server uses the authentication database in */usr/local/etc/fw-authdb*:

```
authsrv:        database /usr/local/etc/fw-authdb
```

This example shows an administrator running as **root** starting **authsrv** on the firewall, creating and enabling a new user in the authentication database.

```
root@firewall # authsrv

-administrator mode-
authsrv# list
Report for users in database
user      group     longname        status proto     last
----      -----     --------        ----- -----      ----
fwadmin                             y    Skey      Thu May  2 11:41:42 1996
scooter             Scooter Lindley y    Snk       Wed May  1 09:02:12 1996
authsrv# adduser john 'John Whorfin'
ok - user added initially disabled
authsrv# enable john
enabled
authsrv# proto john Snk
changed
authsrv# pass '160 270 203 065 022 034 232 162' john
Secret key changed
authsrv# list
Report for users in database
user      group     longname        status proto     last
----      -----     --------        ----- -----      ----
fwadmin                             y    Skey      Thu May  2 11:41:42 1996
scooter             Scooter Lindley y    Snk       Wed May  1 09:02:12 1996
john                John Whorfin    ena  Snk       never
authsrv# quit
#
```

FILES

/etc/rc.local
: Command script that controls automatic reboot, and includes startup information for **authsrv**.

/usr/local/etc/fw-authdb
: The database containing user authentication information.

/usr/local/etc/netperm-table
: The network permissions file contains configuration information for the Firewall Toolkit Internet Firewall, including **authsrv**.

BUGS

Report bugs to the fwall-users mailing list. Include a complete example, explaining what you expected to happen and what actually happened. Be sure to indicate the type of system (operating system, hardware, etc.) you are using, as well as the version of the authsrv proxy.

AUTHOR

Trusted Information Systems, Inc.

SEE ALSO

netperm-table(5), **rc**(8), **authmgr**(8)

NAME

ftp-gw – FTP proxy

SYNOPSIS

ftp-gw [-daemon *port*] **[-version]**

DESCRIPTION

The Firewall Toolkit FTP proxy is an application level proxy that provides configurable access control, authentication and logging mechanisms. The FTP proxy, which runs on the firewall, passes FTP requests through the firewall (at the application level), using rules you supply. You can configure the proxies to allow connections based on:

- source IP address
- source host name
- destination IP address
- destination host name
- FTP command (for example, STOR and RETR)

All packets, and therefore all application requests go to the firewall. On the firewall, the FTP proxy software relays information from one side of the firewall to the other. The proxy prevents the applications on outside networks from talking directly with the applications on your inside network, and vice versa. No IP packets pass from one side of the firewall to the other. All data is passed at the application level.

The default configuration for the FTP proxy does not allow inbound connections from outside the defense perimeter to inside. This means that users must connect to the firewall, authenticate, and then can FTP to the host inside the perimeter.

The FTP proxy (**ftp-gw**) generally runs as a daemon (invoked from */etc/rc.local*) and listens for requests on the standard FTP port (21, as indicated in */etc/services*). Whenever the system receives an FTP request on this port, the FTP proxy checks its configuration information (in the *netperm-table*) and determines whether the initiating host has permission to use FTP. If the host does not have permission, the FTP daemon logs the connection attempt and displays an error message.

If the host has permission, the proxy authenticates the user (if required), logs the transaction and passes the request to the destination host.

The default configuration for the FTP proxy actually uses the network access control daemon (**netacl**) to start the proxy. This configuration allows the firewall to start the FTP proxy or the UNIX FTP daemon (**ftpd**) to handle FTP requests. This allows people to FTP files directly to and from the firewall itself, if needed. FTP proxy remains active until either side closes the connection.

OPTIONS

Command Line Options

The FTP proxy recognizes the following command line options (whether started from the command line or from within */etc/rc.local*):

-daemon *port*

Indicates that the FTP proxy runs as a daemon, and the port (name or number) on which the FTP proxy listens.

Configuration Options

The FTP proxy reads configuration rules from the */usr/local/etc/netperm-table*. It reads all rules using the **ftp-gw** and ***** (wildcard) keywords. The FTP proxy reads the *netperm-table* from top to bottom. If there are multiple rules in the table that could apply for a particular attribute, the FTP proxy uses the first one that it finds. See **netperm-table**(5) for a more complete explanation of *netperm-table* syntax and precedence.

The FTP proxy recognizes the following attributes:

authserver *host port*

Specifies the host running the authentication server (and the port on which it runs) that the proxies use for authenticating users.

host Specifies an IP address or host name.

port Specifies a service name or port number.

denial-msg *file*
Specifies the name of the file the FTP proxy displays when it denies access to a user because they do not have permission to use the FTP proxy. If no file is specified, the FTP proxy generates a default message.

denydest-msg *file*
Specifies the name of the file the FTP proxy displays when it denies access to a user because they are trying to access a destination that they are not permitted to access. If no file is specified, the FTP proxy generates a default message.

hosts host-pattern [host-pattern2...] [options]
rules specify host and access permissions. Typically, a hosts rule will be in the form of:

ftp-gw: deny-hosts unknown ftp-gw: hosts 192.33.112.* 192.94.214.* -log { retr stor }

There may be several host patterns following the "hosts" keyword, ending with the first optional parameter beginning with '-'. Optional parameters permit the selective enabling or disabling of logging information, etc. Sub-options are:

–noinput specifies that no matter what, the proxy should not accept input over a PORT. Attempts to do so result in the port being closed.

–nooutput specifies that no matter what, the proxy should not transmit output over a PORT. Attempts to do so result in the port being closed.

–log operation
–log { operation1 operation2 ... }
specifies that a log entry to the system log should be made whenever the listed operations are performed through the proxy. (See **ftpd(8)** for a list of known FTP operations)

–authall specifies that the proxy should permit no operation (other than the *quit* command) until the user has authenticated to the server.

–auth operation
–auth { operation1 operation2 ... }
specifies that the operations listed should not be permitted until the user has authenticated to the server.

–dest pattern
–dest { pattern1 pattern2 ... }
specifies a list of valid destinations. If no list is specified, all destinations are considered valid. The -dest list is processed in order as it appears on the options line. -dest entries preceeded with a '!' character are treated as negation entries. Therefore the rule:

-dest !*.mit.edu -dest *

will permit hosts that are not in the domain "mit.edu" to be connected to.

–deny operation
–deny { operation1 operation2 ... } specifies a list of FTP operations to deny. By default, all operations are permitted.

directory *directory*
Specifies the directory that the FTP proxy makes its root directory before providing service. This option is equivalent to the **chroot** option in previous versions.

groupid *group*
Specifies the name of the group the FTP proxy uses when running.

group Specifies either a name or numeric id from the */etc/group* file.

help-msg *file*
Specifies the name of the file the FTP proxy displays when the user accesses the help command. If no file is specified, the FTP proxy displays a list of internal commands.

timeout *seconds*
Specifies the number of seconds the FTP proxy is idle (with no network activity) before disconnecting

userid *user*
Specifies the user ID the proxy uses when running.

user Specifies either a name or numeric id from the */etc/passwd* file. This option is equivalent to the **-user** command in previous versions.

welcome-msg *file*
Specifies the name of the file the proxy displays as a welcome banner upon successful connection to the proxy. If no file is specified, the proxy generates a default message.

Proxy Commands

The FTP proxy recognizes all FTP server commands. The following ones are of special interest:

help Displays a list of valid commands for the FTP server.

quit Exits the FTP proxy.

quote *options*
Sends the options verbatim to the remote FTP server. This option is useful when the FTP client does not have support for authentication. In order to authenticate to the FTP proxy, you must use the **quote auth** *username* command to send the user name and the **quote resp** *password* command to send the password. When using the **quote** command, all information (including passwords) is echoed to the screen.

user *username@host*
Connects to the remote host using the given user name and password.

username
Specifies the name of the user on the remote host.

host Specifies the name of the remote host.

You must use the **user** command to tell the proxy the name of the remote host to which you wish to connect.

? Displays a list of valid commands for the FTP server.

EXAMPLES

This example shows the configuration lines in the *netperm-table* that denies **ftp.bigu.edu** as a destination and sets the name of the file that the FTP proxy displays when users try to access this site:

```
#deny ftp.bigu.edu as a destination (by IP address)
ftp-gw: deny-destination 192.168.1.100

#file to display when users try to access a denied site
ftp-gw: denial-msg /usr/local/etc/ftp-deny.txt
```

A common policy for the FTP proxy is to authenticate all requests from untrusted networks to or through the firewall. This example shows a sample FTP session from an untrusted network to a trusted network, using S/Key authentication at the firewall:

blaze.clientsite.com-27: **ftp firewall.yoyodyne.com**
Connected to firewall.yoyodyne.com
220-Proxy first requires authentication
220 firewall.yoyodyne.com FTP proxy (Version 3.1) ready.
Name (firewall.yoyodyne.com:clancy): **clancy**
331 Skey Challenge: s/key 653 fi19289
Password: *password does not display*
230 User authenticated to proxy
ftp> **user clancy@dimension**
331- (-----GATEWAY CONNECTED TO dimension----)
331- (220 dimension FTP server ready.)
331 Password required for clancy.
Password: **#########**
230 User clancy logged in.
ftp>

FILES

/etc/rc.local
: Command script that controls automatic reboot, and includes startup information for the FTP proxy.

/usr/local/etc/netperm-table
: The network permissions file contains configuration information for the Firewall Toolkit, including the FTP proxy.

BUGS

Report bugs to fwtk-support@tis.com or fwtk-users@tis.com. example, explaining what you expected to happen and what actually happened. Be sure to indicate the type of system (operating system, hardware, etc.) you are using, as well as the version of the FTP proxy.

AUTHOR

Trusted Information Systems, Inc.

SEE ALSO

netperm-table(5), **rc**(8), **authsrv**(8), **netacl**(8)

NAME

HTTP-GW - Gopher / HTTP proxy

SYNOPSIS

http-gw [options] - invoked from inetd or rc.local.

DESCRIPTION

HTTP-GW provides Gopher and HTTP proxy services with logging and access control. It allows Gopher and Gopher+ client to access Gopher, Gopher+, and FTP servers. It also allows WWW clients such as Mosaic to access HTTP, Gopher and FTP servers. Both standard and proxy aware WWW clients are supported. The proxy supports **common use** of the Gopher, Gopher+ and HTTP protocols. Except where noted, **client** means Gopher, Gopher+, WWW, or proxy aware WWW clients, **server** means Gopher, Gopher+, HTTP, or FTP servers.

Proxy aware clients should be configured to use the proxy. Non proxy aware clients should be setup so that their **HOME PAGE** is the proxy. If you are installing a firewall where you already have users using Gopher or WWW, then they need to edit their **Hotlists** to route the requests through the proxy.

WWW (URLs)

Insert the string **http://firewall/** in front of the existing URL.

Gopher

Change the Gopher menu information from

Host=somehost
Port=someport
Path=somepath
to
Host=firewall
Port=70
Path=gopher://somehost:someport/somepath

Assuming that the proxy has been configured to be on the default HTTP and Gopher ports (80 and 70 repectively)

OPTIONS

–d file This option can only be used if the proxy was compiled with BINDDEBUG. It allows debugging information to be written to the specified file.

–D This option turns on the debugging log if specified. The proxy must be compiled with BINDDEBUG for the option to be recognised. **-daemon port** Indicates that the HTTP proxy runs as a daemon, and the port (name or number) on which the HTTP proxy listens.

-version

Displays version information for the HTTP proxy on stdout.

OPERATION

HTTP-GW is invoked from **inetd(8),** it reads its configuration and checks to see if the system that has just connected is permitted to use the proxy. If not, it returns a message/menu and logs the connection. If the peer is permitted to use the proxy, **HTTP-GW** reads in a single line request which it then decodes. If needed, more lines are read from the client. Most requests carry the information that the proxy needs in the first line.

When a user initiates a request, the client determines three pieces of information. These are **host, port** and a **selector.** The client then connects to the host on the port and sends the selector. When using a proxy, the host and port refer to the proxy itself. The proxy has to determine the host and port from information contained in the selector. The proxy does this by re-writing the information it passes back to the client. Both Gopher and WWW clients do none or only minimal processing on the selector. If the proxy cannot find it's special information in the selector, it looks in it's configuration file to see if a server has been defined that the request can be handed off to.

There are three main types of information that the proxy has to process.

Gopher menus
these contain a **description** (to be displayed to the user), a **selector,** a **host,** and a **port.** The first character of the description is used to tell the client the sort of information that the entry refers to.

HTML files
contain Hypertext which can contain embedded **links** to other documents. The proxy has to parse the HTML file and re-write the links so that the client routes the request through the proxy.

other data files
these can be roughly classified as **text** or **binary** data. The proxy just passes the data through without changing it.

The proxy encodes the extra information into the selector by converting it into a **URL** (Universal Resource Locator) This is also the form of selector that is used in **HTML** documents.

When building a Gopher Menu from an FTP directory list, the proxy has to guess what Gopher type to specify by looking at the file extension.

Description	Gopher type	Extensions
GIF Image	g	.gif
DOS archives	5	.zip .zoo .arj .arc .lzh
DOS binaries	9	.exe .com .dll .lib .sys
Misc Images	I	.jpg .jpeg .pict .pct .tiff .tif .pcx
Unix binaries	9	.tar .z .gz
MAC archives	4	.hqx
Misc sounds	s	.au .snd .wav
HTML Documents	h	.html .htm
Misc Documents	9	.doc .wri
Directories	1	Filenames that end in /
Plain text	0	All other extensions.

CONFIGURATION

HTTP-GW reads its configuration rules and permissions information from the firewall configuration table **netperm-table,** retrieving all rules specified for "http-gw". The "ftp-gw" rules are also retrieved. The "ftp-gw" rules are consulted when looking for **host** rules once the "http-gw" ones have been searched. The following configuration rules are recognized:

userid user
specify a numeric user-id or the name of a password file entry. If this value is specified, **HTTP-GW** will set its user-id before providing service. Note that this option is included mostly for completeness, as **HTTP-GW** performs no local operations that are likely to introduce a security hole.

directory pathname
specifies a directory to which **HTTP-GW** will chroot(2) prior to providing service.

timeout secondsvalue
This value is used as a dead-watch timer when the proxy is reading data from the net. Defaults to 60 minutes.

default-policy option [options]
This rule sets the default global policy on rewriting Java, ActiveX and Javascript tags in html. The allowed options below can be overridden on the host selection entries: **−java, −nojava, −safejava, −javascript, −nojavascript, −safejavascript, −noactivex, −activex, −safeactivex** which can be overridden on the host selection entries.

browser idstring [options]
This rule defines the browser which sends idstring as the beginning of the User-agent: header and marks it safe for the selected applet types. The options can be: **–safejava, –safejavascript, –safeactivex** which can be combined when the browser is considered safe for multiple types of applets. When a policy:, default-policy: or host permission rule specifies -safexxxx, then the applet type xxxx is allowed only when the browser is defined as safe for that type of executable.

policy option [options]
This rule sets the final global policy on rewriting JAVA and Javascript tags in html. The allowed options are: **–java, –nojava, –safejava, –javascript, –nojavascript –safejavascript, –noactivex, –activex, –safeactivex** which override any individual policy on the host selection entries.

default-gopher server
define a gopher server to which requests can be handed off.

default-httpd server
define an HTTP server to which requests can be handed off if they came from a WWW client using the HTTP protocol.

ftp-proxy server
define an **ftp-gw** that should be used to access **FTP** servers. **If not specified, the proxy will do the FTP transaction with the FTP server.** Since the ftp-gw rules will be used if there are no relevant HTTP-GW rules then this is not a major problem.

hosts host-pattern [host-pattern ...] [options]
permit-hosts host-pattern [host-pattern ...] [options]
deny-hosts host-pattern [host-pattern ...]
rules specify host and access permissions. Typically, a hosts rule will be in the form of:

```
http-gw:        deny-hosts unknown
http-gw:        hosts 192.33.112.* 192.94.214.*
```

There may be several host patterns following the "hosts" keyword, ending with the first optional parameter beginning with '-'. Optional parameters permit the selective enabling or disabling of logging information, etc.

permit-hosts options

permit-hosts rules may take options. Some of the options take parameters. The functions are defined later (see **gopher functions)**

–java
allows Java applet execution for this host entry.

–nojava
denies Java applet execution for this host entry.

–safejava
permits Java execution only if browser listed as safe.

–javascript
allows Javascript execution for this host entry.

–nojavascript
denies Javascript execution for this host entry.

–safejavascript

permits JavaScript execution only if browser listed as safe.

–activex
allows ActiveX execution for this host entry.

–noactivex
denies ActiveX execution for this host entry.

–safeactivex
permits ActiveX execution only if browser listed as safe.

-permit function
-permit { function [function ...] }
Only the specified functions are permitted. Other functions will be denyed. If this option is not specified then all functions are initially permitted.

–deny function
–deny { function [function ...] }
specifies a list of Gopher/HTTP functions to deny.

–gopher server
Make **server** the default server for this transaction.

–httpd server
Make **server** the default HTTP server for this transaction. Will be used if the request came in via the HTTP protocol.

–filter function
–filter { function [function ...] }
will remove the specified functions when re-writing selectors and URL's. Does not stop the user from entering selectors that the client will execute locally but can be used to remove them from retrieved documents.

The following options are also recognised and processed since they may be specified on an ftp-gw config line.

–noinput
Disables data read functions.

–nooutput
Disables data write functions.

–log function
–log { function [function ...] }
specifies that a log entry to the system log should be made whenever the listed functions are performed through the proxy.

–authall
specifies that all functions require the user to be authenticated.

–auth function
–auth { function [function ...] }
specifies that the functions listed require the user to be authenticated.

–dest pattern
–dest { pattern [pattern ...] }
specifies a list of valid destinations. If no list is specified, all destinations are considered valid. The -dest list is processed in order as it appears on the options line. -dest entries preceeded with a '!' character are treated as negation entries. Therefore the rule:

-dest !*.mit.edu -dest *

will permit hosts that are not in the domain "mit.edu" to be connected to.

gopher functions

The proxy characterizes each transaction as one of a number of functions. For the **deny** options it is the request that is used. For **filter** options it is the returned selectors that are used.

Function	Description
dir	Fetching Gopher Menus Getting a directory listing via FTP Fetching an HTML document (This is being studied)
read	Fetching a file of any type. HTML files are treated as **read** even though they are also **dir**
write	Putting a file of any type. Needs **plus** since only available to Gopher+ and HTTP/1.x
ftp	Accessing an FTP server
plus	Gopher+ operations HTTP methods other than GET
wais	WAIS index operations
exec	Operations that require a program to be run, e.g. telnet. (See **SECURITY**)

SECURITY

There are a number of situations that the user needs to know about. The most important of which is the way that certain functions are handled by the **client, server,** and **proxy** programs. When the client wants to perform certain actions such as **telnet** then the client program often runs the telnet command to perform the function. If the client passes arguments to the program then there is a chance that rogue commands could be executed as well as the intended one. Gopher requests to do FTP operations cause the server to run the FTP program. Again there is the chance that the server could be tricked into running rogue commands as well as the intended one.

Most client programs only know how to display a small number of data types and they rely on external **viewers** to handle the other data types. Again, there is a chance that client programs can be tricked into running rogue commands as well as the intended ones.

INSTALLATION

To install **HTTP-GW** first place the executable in a system area, then modify **/etc/inetd.conf.** The TCP service port on which to install the Gopher/HTTP proxy will depend on local site configuration. You would normally configure the proxy to be on ports 70 and 80. 70 is the normal Gopher port and 80 is the normal HTTP port. Once **inetd.conf has been modified, restart or reload inetd.** Verify installation by attempting a connection and monitoring the system logs.

Typical configuration of the proxy in a firewall situation involves rules to block all systems that are not in the DNS from using the proxy, but to permit all systems on the internal protected network to use the proxy. I.e.:

http-gw: deny-hosts unknown
http-gw: hosts 192.33.112.* 192.94.214.*

FILES

/etc/inetd.conf **/etc/services netperm-table**

SEE ALSO

netperm-table(5) **inetd**(8) **authd**(8) **ftp-gw**(8)

BUGS

NAME

login-sh – authenticating login shell

SYNOPSIS

login-sh

DESCRIPTION

The login shell program is a wrapper program that authenticates the user (using strong authentication) before passing control to the real login shell. It provides authentication and logging.

A user logs into the firewall via the console, TELNET or Rlogin. This calls the standard login program (*/bin/login*) to process the login. The login program asks for a user name. The login program reads the password file (*/etc/passwd*) and determines that this user does not require a password (because the password field is empty). It then passes the information to the program specified in the shell field, the login shell program (*/usr/local/etc/login-sh*).

The login shell program prompts the user for the appropriate response for their authentication method (SNK response, S/Key password, etc.) The login shell program checks its configuration information (in the *netperm-table*). It authenticates the user using the authentication server specified in the netperm-table. The login shell program then reads the shell file configuration file (by default, */usr/local/etc/login-shellfile*). It passes the login information on to the executable specified for the user in the shell configuration file, which is normally the user's shell.

OPTIONS

Command Line Options

The **login-sh** recognizes no command line options.

Configuration Options

The **login-sh** reads configuration rules from the */usr/local/etc/netperm-table*. It reads all rules using the **login-sh** and ***** (wildcard) keywords. **login-sh** reads the *netperm-table* from top to bottom. If there are multiple rules in the table that could apply for a particular attribute, the **login-sh** uses the first one that it finds. See **netperm-table**(5) for a more complete explanation of *netperm-table* syntax and precedence.

The **login-sh** recognizes the following attributes:

authserver *host port*

Specifies the host running the authentication server (and the port on which it runs) that the proxies use for authenticating users.

host Specifies an IP address or host name.

port Specifies a service name or port number.

shellfile *file*

Specifies the name of the file in which the login shell finds information about users and their actual shells. This attribute is required.

EXAMPLES

This example shows the entry in */etc/shells* that defines **login-sh** as a valid shell:

```
/usr/local/etc/login-sh
```

This example shows the entry in the */etc/passwd* file, which indicate that this user's shell is **login-sh**:

```
scooter::518:10:Scooter Lindley:/home/scooter:/usr/local/etc/login-sh
```

This example shows the contents of the shell configuration file (by default, */usr/local/etc/login-shellfile*):

```
scooter   /usr/bin/tcsh   -tcsh
```

This example shows the configuration options in the *netperm-table* that tell the **login-sh** program which file to use:

#define the file that contains actual shell info
login-sh: shellfile /usr/local/etc/login-shellfile

FILES

/etc/passwd
File containing information, including the name of their shell, for each user.

/etc/shells
File containing a list of legal shells.

/usr/local/etc/login-shellfile
The login shell configuration file that contains information about user's actual shells.

/usr/local/etc/netperm-table
The network permissions file contains configuration information for the Firewall Toolkit, including the **login-sh**

BUGS

Report bugs to fwtk-support@tis.com or fwtk-users@tis.com. Include a complete example, explaining what you expected to happen and what actually happened. Be sure to indicate the type of system (operating system, hardware, etc.) you are using, as well as the version of **login-sh**.

AUTHOR

Trusted Information Systems, Inc.

SEE ALSO

passwd(4), **shells**(4), **netperm-table**(5), **rc**(8), **authsrv**(8)

NAME

netacl − TCP network access control daemon

SYNOPSIS

netacl [-daemon *port***]** *service* **[-version]**

DESCRIPTION

The Firewall Toolkit network access control daemon is a TCP wrapper program that provides configurable access control and logging mechanisms. The network access control daemon, which runs on the firewall, starts different applications based on the source address of the request.

Using the network access control daemon, you can allow certain hosts to access a standard UNIX program, such as the TELNET daemon, while requests from all other hosts access the TELNET proxy.

The network access control daemon allows you to configure which hosts have access to which TCP-based services. Note that it does not allow you to start UDP-based services. The access control daemon logs all successful and unsuccessful connections.

The firewall runs different instances of the network access control daemon (**netacl**) on different ports for different applications, based on the information in the */etc/rc.local*) file. The */etc/rc.local* file indicates which services should run on which ports. For example, the firewall runs an instance of the network access control daemon on port 23 to handle TELNET requests.

When the network access control daemon receives a request on a port on which it is listening, the daemon checks its configuration information (in the *netperm-table*) and determines whether the initiating host has permission to initiate this type of request. The network access control daemon then verifies that it has permission to run. If the host does not have permission or the network access control daemon is not permitted to run, the firewall displays an error message.

If the host has permission, the proxy and the network access control daemon is permitted to run, the network access control daemon then starts the program specified in the netperm-table. For example, the network access control daemon might start the TELNET proxy (**tn-gw**) for some initiating hosts and the actual TELNET daemon (**telnetd**) for other initiating hosts.

OPTIONS

Command Line Options

The network access control daemon recognizes the following command line options (whether started from the command line or from within */etc/rc.local*):

-daemon *port*

Indicates that the network access control daemon runs as a daemon, and the port (name or number) on which the network access control daemon listens.

service Indicates the name of the service the network access control daemon runs

-version

Displays version information for the network access control daemon on stdout.

Configuration Options

The network access control daemon reads configuration rules from the */usr/local/etc/netperm-table*. It reads all rules using the **netacl** (or the name specified with the -as option) and * (wildcard) keywords. The network access control daemon reads the *netperm-table* from top to bottom. If there are multiple rules in the table that could apply for a particular attribute, the network access control daemon uses the first one that it finds. See **netperm-table**(5) for a more complete explanation of *netperm-table* syntax and precedence.

The network access control daemon recognizes the following attributes:

groupid *group*

Specifies the name of the group the network access control daemon uses when running.

group Specifies either a name or numeric id from the */etc/group* file.

hosts *[options]*

pecifies a host permission rule. Host permission rules are in the form of:

netacl-in.telnet permit-hosts host1 host2 -options netacl-in.telnet deny-hosts host1 host2 -options

Following the **permit-hosts** or **deny-hosts** clause is a list of host names or IP-addresses, which can contain wildcards. Host names are searched in order until the first option (starting with a '-') is encountered, at which point, if there is a match for that rule, it will be accepted. If the rule is a deny-hosts rule, the program will log the denial of service and exit. If it is a permit-hosts rule, the options will be processed and executed in order. If there is no rule explicitly permitting or deny a service, the service is denied. Options are:

–exec executable [args] specifies a program to invoke to handle the service. This option **must** be the final option in the rule. A -exec option **must** be present in every rule.

–user userid where userid is the numeric UID or the name from a login in **/etc/passwd which the program should be invoked as.**

–chroot rootdir specifies a directory to which **netacl** should chroot(2) prior to invoking the service program. This requires that the service program be present, and the pathname for the executable be relative to the new root.

timeout *seconds*

Specifies the number of seconds the network access control daemon is idle (with no network activity) before disconnecting

EXAMPLES

This example shows the configuration lines in the netperm-table that start the TELNET daemon for connections from localhost and the TELNET proxy for all other connections:

```
#starts telnet for connections from localhost
netacl-telnetd: permit-hosts 127.0.0.1 -exec /usr/libexec/telnetd

#starts the telnet proxy for all other connections
netacl-telnetd: exec /usr/local/etc/tn-gw
```

Another example, using netacl to verify permission for using a service:

```
netacl-telnetd: permit-hosts 1.2.3.* -exec /usr/libexec/telnetd
netacl-ftpd: permit-hosts unknown -exec /bin/cat /etc/noftp.txt
netacl-ftpd: permit-hosts 1.2.3.* -exec /usr/etc/in.ftpd
netacl-ftpd: permit-hosts * -chroot /home/ftp -exec /bin/ftpd -f
```

In the example above, **netacl** is configured to permit telnet only for hosts in a particular subnet. **ftpd** is configured to accept all connections from systems that do not have a valid DNS name ("unknown") and to invoke **cat** to display a file when a connection is made. This provides an easy and flexible means of politely informing someone that they are not permitted to use a service. Hosts in the specified subnet are connected to the real FTP server in **/usr/etc/in.ftpd** but all connections from other networks are connected to a version of the FTP server that is already chrooted to the FTP area, effectively making all FTP activity "captive".

FILES

/etc/rc.local

Command script that controls automatic reboot, and includes startup information for the network access control daemon.

/usr/local/etc/netperm-table

The network permissions file contains configuration information for the Firewall Toolkit, including the network access control daemon.

BUGS

Report bugs to fwtk-support@tis.com or fwtk-users@tis.com. Include a complete example, explaining what you expected to happen and what actually happened. Be sure to indicate the type of system (operating system, hardware, etc.) you are using, as well as the version of the netacl proxy.

AUTHOR

Trusted Information Systems, Inc.

SEE ALSO

netperm-table(5), **rc**(8)

NAME

netperm-table − configuration file for Firewall Toolkit proxies and utilities

DESCRIPTION

The network permissions table (*/usr/local/etc/netperm-table*) contains configuration information for the Firewall Toolkit. The proxies and other applications read their configuration information from this table. The rules in the table include two types of information: generic proxy rules and application-specific rules.

The permissions/configuration file is organized into rules, one rule per line of text. The first part of each rule is the name of the application that rule applies to, followed by a colon. Multiple applications can be targeted by a single rule, by separating the names with commas, or wildcarding them with an asterisk. When an application extracts its configuration information, it only extracts the rules that apply to it, preserving the order in which they appeared in the file. For example, if the application **smap** read its configuration rules, the following would all match:

```
# sample rules for smap
smap, smapd:  userid 4
smap, smapd:  directory /mail/inspool
smap:         timeout 3600
```

Comments can be inserted in the configuration file by starting the line with '#' as the first character.

Once an application has matched a rule, the rule is tokenized into whitespace-delimited strings for later application specific use. Typically, the application retrieves matching rules based on the first word in the rule; the remaining words serve as parameters to that clause. Special modifiers are recognized for each clause, if it begins with a **permit-** or **deny-** the rule is internally flagged as granting or revoking permission for that clause. I.e., if an application retrieves all of its configuration clauses for "hosts", the following will be returned:

```
netacl-in.ftpd: permit-hosts 192.33.112.117 -exec /usr/etc/in.ftpd
netacl-in.ftpd: permit-hosts 198.137.240.101 -exec /usr/etc/in.ftpd
netacl-in.ftpd: deny-hosts unknown
netacl-in.ftpd: deny-hosts *
```

NETPERM-TABLE SYNTAX

Precedence

Applications and proxies read the tables from the top of the table to the bottom. They use the first rule that applies for a particular attribute. If there are multiple rules in the table that could apply for an attribute, the application uses the first one it finds. For example, a *netperm-table* contains the following rule:

```
smapd:  userid  uucp
```

and later in the file contains the rule:

```
smapd:  userid    mail
```

When **smapd** parses the *netperm-table*, it will use the first rule it finds, and run as the user uucp.

Format

Each line in the *netperm-table* contains a separate configuration rule in the format:

keyword: attribute valuelist

where:

keyword

indicates the application to which the rule on that line applies. The wildcard (*) indicates the rule is valid for all applications and proxies. Comma separated lists of multiple keywords indicates the rules apply to the proxies or applications listed. The keyword usually matches the name of the service in the */etc/rc.local* file.

attribute
is a configuration parameter for that application or proxy.

valuelist
is the value for the specific configuration parameter. Some attributes allow multiple values.

A rule must fit on a single line. The length of a line varies by operating system, but is generally around 1,024 bytes. There is no provision for continuing lines.

Whitespace and tabs are both valid separators.

Comments

A hash mark (#) at the beginning of a line indicates a comment. Applications ignore any text between the # and the end of the line.

CONVENTIONS

Much of the usage of the **netperm-table** is application specific, but where possible, several conventions are enforced. When a host name or host IP address is given to match against, matching is performed based on whether the pattern to match against is all digits and decimal points, or whether it contains other characters. If the pattern to match against is entirely digits and decimals, matching is performed against the IP address, otherwise it is performed against the hostname. Generally, to prevent being vulnerable to DNS spoofing, one should use configuration rules based on IP addresses. When matching, asterisk wildcards are supported, with syntax similar to the shell's, matching as many characters as possible. When resolving IP addresses to DNS names, if the reverse lookup fails, the host name is set to "unknown." When resolution is performed, a check is made to ensure that the IP address for the DNS name returned by the reverse lookup is the same. If it is not, the host name is set to "unknown" and a warning is logged. This permits rules to specifically operate on hosts that didn't have valid DNS mappings. Thus:

192.33.* will match against IP address
*.tis.com will match against host name
unknown will match against hosts not in the DNS

Note that sites using NIS or other resolver systems where the *gethostbyaddr* library routine returns a non-fully-qualified domain name will have problems generating a matching rule using their domain name. In this case, the numeric IP address should be used. In general, using IP addresses will provide better security, by removing the threat of DNS spoofing attacks.

DIAGNOSTICS

Syslog error messages are generated by malformed lines, including lines that are missing the application name or colon, lines that are missing any clause following the colon, lines that have too many tokens, or a memory allocation error.

FILES

/usr/local/etc/netperm-table

CAVEATS

The location of **netperm-table** is compiled into the components of the firewall toolkit. Its location may change, depending on compile time options.

Applications in the toolkit that attempt to open **netperm-table** and fail will exit with a warning in syslog.

FILES

/usr/local/etc/netperm-table
The network permissions file contains configuration information for the Firewall Toolkit

BUGS

Report bugs to fwtk-support@tis.com or fwtk-users@tis.com. Include a complete example, explaining what you expected to happen and what actually happened. Be sure to indicate the type of system (operating system, hardware, etc.) you are using, as well as the version of the *netperm-table*.

AUTHOR

Trusted Information Systems, Inc.

SEE ALSO

authmgr(8), **authsrv** (8), **ftp-gw**(8), **http-gw**(8), **login-sh**(8), **plug-gw**(8), **rlogin-gw**(8), **smap**(8), **smapd**(8), **tn-gw**(8), **x-gw**(8)

NAME

plug-gw – plug proxy

SYNOPSIS

plug-gw [-daemon *port***]** *service* **[-version]**

DESCRIPTION

The Firewall Toolkit plug proxy is an application level proxy that provides configurable access control, authentication and logging mechanisms. The plug proxy, which runs on the firewall, passes NNTP or other TCP-based application requests through the firewall (at the application level), using rules you supply. You can configure instances of the plug proxy to service:

- NNTP news feeds
- webster
- whois

This is not an exhaustive list. The plug proxy is protocol neutral, so you can tunnel a variety of other TCP-based applications. Weigh the risks carefully for each application.

For each version of the plug proxy, you can configure the proxy to allow connections based on:

- source IP address
- source host name
- source port
- destination IP address
- destination host name
- destination port

All packets, and therefore all application requests go to the firewall. On the firewall, the plug proxy software relays information from one side of the firewall to the other. The proxy prevents the applications on outside networks from talking directly with the applications on your inside network, and vice versa. No IP packets pass from one side of the firewall to the other. All data is passed at the application level.

The firewall runs different instances of the plug proxy (**plug-gw**) as daemons (invoked from */etc/rc.local*) on different ports for different applications, based on the information in the */etc/services* and */etc/rc.local* files. These files indicate which services the firewall should run on which ports. For example, the firewall runs an instance of the plug proxy on port 119 to handle NNTP requests.

Whenever the system receives a request on one of these ports, the plug proxy checks its configuration information (in the *netperm-table*) and determines whether the initiating host has permission to initiate this type of request. If the host does not have permission, the plug daemon logs the connection attempt and displays an error message.

If the host has permission, the proxy logs the transaction and passes the request to the destination host. The plug proxy remains active until either side closes the connection.

WARNING:

Allowing proprietary protocols through your firewall is a really big unknown. Because the protocols are proprietary, the firewall and the proxy have no idea what sorts of data or requests the applications are sending. Nor do we have any idea how safe the actual application is. Do not use the plug proxy for proprietary protocols without first performing a risk assessment.

OPTIONS

Command Line Options

The plug proxy recognizes the following command line options (whether started from the command line or from within */etc/rc.local*):

-daemon *port*
Indicates that the plug proxy runs as a daemon, and the port (name or number) on which the plug proxy listens.

service Indicates the name of the service the plug proxy connects as.

-version
Displays version information for the plug proxy on stdout.

Configuration Options

The plug proxy reads configuration rules from the */usr/local/etc/netperm-table*. It reads all rules using the **plug-gw** (or the name specified with the -as option) and * (wildcard) keywords. The plug proxy reads the *netperm-table* from top to bottom. If there are multiple rules in the table that could apply for a particular attribute, the plug proxy uses the first one that it finds. See **netperm-table**(5) for a more complete explanation of *netperm-table* syntax and precedence.

The plug proxy recognizes the following attributes:

groupid *group*
Specifies the name of the group the plug proxy uses when running.

group Specifies either a name or numeric id from the */etc/group* file.

port *portid* **host-pattern** *[options]*
specifies a connection rule. When a connection is made, a match is searched for on the port-id and calling host. The port-id may be either a numeric value (e.g.: 119) or a value from **/etc/services** (e.g.: "nntp"). If the calling port matches, then the host-pattern is checked for a match, following the standard address matching rules employed by the firewall. If the rule matches, the connection will be made based on the remaining options in the rule, all of which begin with '-'. Sub-options are:

–plug-to host specifies the name or address of the host to connect to. This option is mandatory.

–privport indicates that a reserved port number should be used when connecting. Reserved port numbers must be specified for protocols like **rlogin** which rely on them for "security."

–port portid specifies a different port. The default port is the same as the port used by the incoming connection.

timeout *seconds*
Specifies the number of seconds the plug proxy is idle (with no network activity) before disconnecting.

userid *user*
Specifies the user ID the proxy uses when running.

user Specifies either a name or numeric id from the */etc/passwd* file.

EXAMPLES

This example shows the configuration lines in the *netperm-table* for a one-to-one connection from inside to outside:

```
# allows one host inside to connect to one host outside
qotd-gw: port qotd 10.0.1.12 -plug-to info.bigu.edu -port qotd
```

FILES

/etc/rc.local
Command script that controls automatic reboot, and includes startup information for the plug proxy.

/usr/local/etc/netperm-table
The network permissions file contains configuration information for the Firewall Toolkit, including the plug proxy.

NOTES

Since incoming connection hosts can be wildcarded, **plug-gw** works well in a many-to-one relationship but does not work at all in a one-to-many relationship. If, for example, a site has 3 news feeds - it is easy to configure plug-gw to plugboard any connections from those 3 hosts to an internal news server, but unless there are multiple instances of **plug-gw** on different ports, and the internal news server's software can support connecting on a non-standard port, modification to software will be required.

BUGS

Report bugs to fwtk-support@tis.com or fwtk-users@tis.com. Include a complete example, explaining what you expected to happen and what actually happened. Be sure to indicate the type of system (operating system, hardware, etc.) you are using, as well as the version of the plug proxy.

AUTHOR

Trusted Information Systems, Inc.

SEE ALSO

netperm-table(5), **rc**(8)

NAME

rlogin-gw – Rlogin proxy

SYNOPSIS

rlogin-gw [-daemon *port*] **[-version]**

DESCRIPTION

The Firewall Toolkit Rlogin proxy is an application level proxy that provides configurable access control, authentication and logging mechanisms. The Rlogin proxy, which runs on the firewall, passes Rlogin requests through the firewall (at the application level), using rules you supply. You can configure the proxies to allow connections based on:

- source IP address
- source host name
- destination IP address
- destination host name

All packets, and therefore all application requests go to the firewall. On the firewall, the Rlogin proxy software relays information from one side of the firewall to the other. The proxy prevents the applications on outside networks from talking directly with the applications on your inside network, and vice versa. No IP packets pass from one side of the firewall to the other. All data is passed at the application level.

The default configuration for the Rlogin proxy requires authentication for inbound connections from outside the defense perimeter to inside. This means that users must connect to the firewall, authenticate, and then can Rlogin to the host inside the perimeter.

The Rlogin proxy (**rlogin-gw**) generally runs as a daemon (invoked from */etc/rc.local*) and listens for requests on the standard Rlogin port (513, as indicated in */etc/services*). Whenever the system receives a Rlogin request on this port, the Rlogin proxy checks its configuration information (in the *netperm-table*) and determines whether the initiating host has permission to use Rlogin. If the host does not have permission, the Rlogin daemon logs the connection attempt and displays an error message.

If the host has permission, the proxy authenticates the user (if required), logs the transaction and passes the request to the destination host.

The default configuration for the Rlogin proxy actually uses the network access control daemon (**netacl**) to start the proxy. This configuration allows the firewall to start the Rlogin proxy or the UNIX Rlogin daemon (**rlogind**) to handle Rlogin requests. This allows people to Rlogin directly to and from the firewall itself, if needed. The Rlogin proxy remains active until either side closes the connection.

OPTIONS

Command Line Options

The Rlogin proxy recognizes the following command line options (whether started from the command line or from within */etc/rc.local*):

-daemon *port*

Indicates that the Rlogin proxy runs as a daemon, and the port (name or number) on which the Rlogin proxy listens.

-version

Displays version information for the Rlogin proxy on stdout.

Configuration Options

The Rlogin proxy reads configuration rules from the */usr/local/etc/netperm-table*. It reads all rules using the **rlogin-gw** and ***** (wildcard) keywords. The Rlogin proxy reads the *netperm-table* from top to bottom. If there are multiple rules in the table that could apply for a particular attribute, the Rlogin proxy uses the first one that it finds. See **netperm-table**(5) for a more complete explanation of *netperm-table* syntax and precedence.

The Rlogin proxy recognizes the following attributes:

prompt string
specifies a prompt for **rlogin-gw** to use while it is in command mode.

timeout seconds
specifies the number of seconds of idleness after which the proxy should disconnect. Default is no timeout.

authserver *host port*
Specifies the host running the authentication server (and the port on which it runs) that the proxies use for authenticating users.

host Specifies an IP address or host name.

port Specifies a service name or port number.

denial-msg *file*
Specifies the name of the file the Rlogin proxy displays when it denies access to a user because they do not have permission to use the Rlogin proxy. If no file is specified, the Rlogin proxy generates a default message.

denydest-msg *file*
Specifies the name of the file the Rlogin proxy displays when it denies access to a user because they are trying to access a destination that they are not permitted to access. If no file is specified, the Rlogin proxy generates a default message.

help-msg *file*
Specifies the name of the file the Rlogin proxy displays when the user accesses the help command. If no file is specified, the Rlogin proxy displays a list of internal commands.

hosts*host-pattern***[host-pattern2...]***[options]*
rules specify host and access permissions. Typically, a hosts rule will be in the form of:

rlogin-gw: deny-hosts unknown rlogin-gw: hosts 192.33.112.* 192.94.214.*

There may be several host patterns following the "hosts" keyword, ending with the first optional parameter beginning with '-'. Sub-options are:

−dest pattern
−dest { pattern1 pattern2 ... }
specifies a list of valid destinations. If no list is specified, all destinations are considered valid. The -dest list is processed in order as it appears on the options line. -dest entries preceeded with a '!' character are treated as negation entries. Therefore the rule:

-dest !*.mit.edu -dest *

will permit hosts that are not in the domain "mit.edu" to be connected to.

−auth
specifies that the proxy should require a user to authenticate with a valid user-id prior to being permitted to use the gateway.

−passok
specifies that the proxy should permit users to change their passwords if they are connected from the designated host. Only hosts on a trusted network should be permitted to change passwords, unless token-type authenticators are distributed to all users.

welcome-msg *file*
Specifies the name of the file the proxy displays as a welcome banner upon successful connection to the proxy. If no file is specified, the proxy generates a default message.

xforwarder *program*
Specifies the location of the executable to which the Rlogin proxy passes requests for the X proxy. Generally specifies the location of the X proxy.

Proxy Commands

connect *host* **[***port***]**
Connects to the specified host on the specified port.

host Specifies a host name or IP address.

port Specifies the name or port number, as specified in */etc/services.*

exit Exits the Rlogin proxy.

help Displays a list of valid commands for the Rlogin proxy.

password
Allows you to change your strong authentication password. The Rlogin proxy only allows you to change your strong authentication password from the hosts listed in the **netperm-table**.

quit Exits the Rlogin proxy.

timeout*seconds*
Specifies the number of seconds the Rlogin proxy is idle (with no network activity) before disconnecting. Note that this setting overrides the setting in the *netperm-table*.

x-gw [*hostname***:***display***[.***screennumber***]]**
Invokes the X11 proxy to connect to the specified host and display. See **x-gw**(8) for more information on how the X11 proxy works.

? Displays a list of valid commands for the Rlogin proxy.

EXAMPLES

This example shows the configuration lines in the *netperm-table* that set the default idle time to five minutes (300 seconds):

```
#sets the idle time to five minutes
rlogin-gw: timeout 300
```

A common policy for the Rlogin proxy is to authenticate all requests from untrusted networks to or through the firewall. This example shows a sample Rlogin session from an untrusted network to a trusted network, using S/Key authentication at the firewall.

```
blaze.clientsite.com-28: telnet firewall.yoyodyne.com
Trying 204.255.154.100...
Connected to firewall.yoyodyne.com
Escape character is '^]'.
Username: scooter
Skey Challenge: s/key 651 fi19289 SAFE DUB RISK CUE YARD NIL
Login Accepted
firewall.yoyodyne.com telnet proxy (Version 3.1) ready:
tn-gw> c dimension
Trying 10.0.1.120 port 23...

Connected  to dimension.yoyodyne.com

BSDI BSD/OS 2.0.1 (dimension) (ttyp5)

Welcome to dimension.yoyodyne.com
3:57PM  up 16 days,  5:35, 4 users, load averages: 0.03, 0.01, 0.00
dimension-26:
```

FILES

/etc/rc.local
: Command script that controls automatic reboot, and includes startup information for the Rlogin proxy.

/usr/local/etc/netperm-table
: The network permissions file contains configuration information for the Firewall Toolkit, including the Rlogin proxy.

BUGS

Report bugs to fwtk-support@tis.com or fwtk-users@tis.com. Include a complete example, explaining what you expected to happen and what actually happened. Be sure to indicate the type of system (operating system, hardware, etc.) you are using, as well as the version of the Rlogin proxy.

AUTHOR

Trusted Information Systems, Inc.

SEE ALSO

netperm-table(5), **rc**(8), **authsrv**(8), **x-gw**(8)

NAME

smap − SMTP proxy (client portion)

SYNOPSIS

smap [-daemon [*port* **]]**

DESCRIPTION

The Firewall Toolkit **smap** proxy is an application level proxy that provides configurable access control and logging mechanisms. The **smap** client proxy together with the **smapd** server proxy, which run on the firewall, transfer mail between internal and external mail servers, using rules you supply.

All packets, and therefore all application requests go to the firewall. On the firewall, the **smap** and **smapd** proxy software relay information from one side of the firewall to the other. They prevent versions of *sendmail* on the external network from talking with versions of *sendmail* on the internal network, and vice versa. No IP packets pass from one side of the firewall to the other. All data is passed at the application level.

The client portion of the SMTP proxy (**smap**) generally runs as a daemon (invoked from */etc/rc.local*) and listens for requests on the standard SMTP port (25, as indicated in */etc/services*). Whenever the system receives an SMTP request on this port, the **smap** client collects the mail from the sender, logs the message, and places the mail in a temporary directory. Periodically, based on a configurable value (by default every 60 seconds), the server daemon (**smapd**) wakes up and checks to see if there is any new mail. The **smapd** daemon checks the headers of the mail for formatting problems. It then calls the configured message transfer agent (usually **sendmail** in delivery mode) for final delivery.

OPTIONS

Command Line Options

The **smap** proxy recognizes the following command line options (whether started from the command line or from within */etc/rc.local*):

-daemon *port*

Indicates that the **smap** proxy runs as a daemon, and the port (name or number) on which the **smap** proxy listens.

Configuration Options

The **smap** proxy reads configuration rules from the */usr/local/etc/netperm-table*. It reads all rules using the **smap** and * (wildcard) keywords. The **smap** proxy reads the *netperm-table* from top to bottom. If there are multiple rules in the table that could apply for a particular attribute, the **smap** proxy uses the first one that it finds. See **netperm-table**(5) for a more complete explanation of *netperm-table* syntax and precedence.

The **smap** proxy recognizes the following attributes:

directory *directory*

Specifies the directory that the **smap** proxy makes its root directory before providing service. This is also the directory where the **smap** proxy stores incoming messages.

groupid *group*

Specifies the name of the group the **smap** proxy uses when running.

group Specifies either a name or numeric id from the */etc/group* file.

timeout *seconds*

Specifies the number of seconds the **smap** proxy is idle (with no network activity) before disconnecting

userid *user*

Specifies the user ID the proxy uses when running.

user Specifies either a name or numeric id from the */etc/passwd* file.

EXAMPLES

This example indicates that the **smap** and **smapd** proxies use the directory */var/spool/smap* as their root directories:

smap, smapd: directory /var/spool/smap

FILES

/etc/rc.local

Command script that controls automatic reboot, and includes startup information for the **smap** proxy.

/usr/local/etc/netperm-table

The network permissions file contains configuration information for the Firewall Toolkit, including the **smap** proxy.

/var/spool/smap

The directory that the **smap** proxy uses as its root directory.

BUGS

Report bugs to fwtk-support@tis.com or fwtk-users@tis.com. Include a complete example, explaining what you expected to happen and what actually happened. Be sure to indicate the type of system (operating system, hardware, etc.) you are using, as well as the version of the smap proxy.

AUTHOR

Trusted Information Systems, Inc.

SEE ALSO

netperm-table(5), **rc**(8), **smap**(8)

NAME

smapd − SMTP proxy (server portion)

SYNOPSIS

smapd [-d]

DESCRIPTION

The Firewall Toolkit **smapd** proxy is an application level proxy that provides configurable access control and logging mechanisms. The **smap** client proxy together with the **smapd** server proxy, which run on the firewall, transfer mail between internal and external mail servers, using rules you supply.

All packets, and therefore all application requests go to the firewall. On the firewall, the **smap** and **smapd** proxy software relay information from one side of the firewall to the other. They prevent versions of *sendmail* on the external network from talking with versions of *sendmail* on the internal network, and vice versa. No IP packets pass from one side of the firewall to the other. All data is passed at the application level.

The server portion of the SMTP proxy (**smapd**) generally runs as a daemon (invoked from */etc/rc.local*). Periodically, based on a configurable value (by default every 60 seconds), the server daemon (**smapd**) wakes up and checks to see if there is any new mail that the SMTP client proxy (**smap**) has placed in the spool directory. The **smapd** daemon checks the headers of the mail for formatting problems. It then calls the configured message transfer agent (usually **sendmail** in delivery mode) for final delivery

OPTIONS

Command Line Options

The **smapd** proxy recognizes the following command line options (whether started from the command line or from within */etc/rc.local*):

-d Starts the smapd proxy in debug mode, which causes the proxy to display status and progress messages as it processes mail messages.

Configuration Options

The **smapd** proxy reads configuration rules from the */usr/local/etc/netperm-table*. It reads all rules using the **smapd** and ***** (wildcard) keywords. The **smapd** proxy reads the *netperm-table* from top to bottom. If there are multiple rules in the table that could apply for a particular attribute, the **smapd** proxy uses the first one that it finds. See **netperm-table**(5) for a more complete explanation of *netperm-table* syntax and precedence.

The **smapd** proxy recognizes the following attributes:

badadmin *user*

Specifies the username to which the **smapd** server forwards mail that it cannot deliver.

user Specifies a user name or alias.

baddir *directory*

Specifies the directory in which the **smapd** server places any spooled mail that it cannot deliver normally.

directory

Specifies a directory on the same device as the spool directory. Do not include a trailing slash (/) character. Ensure that this directory has the same owner and permission as the normal directory that **smap** uses.

directory *directory*

Specifies the directory that the **smapd** proxy makes its root directory before providing service.

executable *program*

Specifies the name of the **smapd** program itself. This attribute is required, as the **smapd** process forks and execs new copies of itself to handle each mail message.

groupid *group*
Specifies the name of the group the **smapd** proxy uses when running.

group Specifies either a name or numeric id from the */etc/group* file.

maxchildren *children*
Specifies the maximum number of child processes the **smapd** server can fork to handle mail.

sendmail *program*
Specifies an alternate path for **sendmail**, or another mail delivery program you are using to deliver your mail inside your perimeter.

userid *user*
Specifies the user ID the proxy uses when running.

user Specifies either a name or numeric id from the */etc/passwd* file.

wakeup *seconds*
Specifies the number of seconds that the **smapd** server sleeps between scans of the spool directory. If no value is specified, **smapd** uses a default value of 60 seconds.

EXAMPLES

This example places the undelivered mail in the */var/spool/smap/badmail* directory:

```
smapd:  baddir /var/spool/smap/badmail
```

FILES

/etc/rc.local
Command script that controls automatic reboot, and includes startup information for the **smapd** proxy.

/usr/local/etc/netperm-table
The network permissions file contains configuration information for the Firewall Toolkit, including the **smapd** proxy.

/var/spool/smap
The directory that the **smapd** proxy uses as its root directory.

BUGS

Report bugs to fwtk-support@tis.com or fwtk-users@tis.com. Include a complete example, explaining what you expected to happen and what actually happened. Be sure to indicate the type of system (operating system, hardware, etc.) you are using, as well as the version of the smapd proxy.

AUTHOR

Trusted Information Systems, Inc.

SEE ALSO

netperm-table(5), **rc**(8), **smap**(8)

NAME

tn-gw − TELNET proxy

SYNOPSIS

tn-gw [-daemon *port*] **[-version]**

DESCRIPTION

The Firewall Toolkit TELNET proxy is an application level proxy that provides configurable access control, authentication and logging mechanisms. The TELNET proxy, which runs on the firewall, passes TELNET requests through the firewall (at the application level), using rules you supply. You can configure the proxies to allow connections based on:

- source IP address
- source host name
- destination IP address
- destination host name

All packets, and therefore all application requests go to the firewall. On the firewall, the TELNET proxy software relays information from one side of the firewall to the other. The proxy prevents the applications on outside networks from talking directly with the applications on your inside network, and vice versa. No IP packets pass from one side of the firewall to the other. All data is passed at the application level.

The TELNET proxy (**tn-gw**) generally runs as a daemon (invoked from */etc/rc.local*) and listens for requests on the standard TELNET port (23, as indicated in */etc/services*). Whenever the system receives a TELNET request on this port, the TELNET proxy checks its configuration information (in the *netperm-table*) and determines whether the initiating host has permission to use TELNET. If the host does not have permission, the TELNET daemon logs the connection attempt and displays an error message.

If the host has permission, the proxy authenticates the user (if required), logs the transaction and passes the request to the destination host. If the host is permitted to use the proxy, **tn-gw** enters a command loop in which it awaits for a user to specify a) the system they wish to connect with, b) the X-gateway they wish to invoke.

The default configuration for the TELNET proxy actually uses the network access control daemon (**netacl**) to start the proxy. This configuration allows the firewall to start the TELNET proxy or the UNIX TELNET daemon (**telnetd**) to handle TELNET requests. This allows people to TELNET directly to and from the firewall itself, if needed. The TELNET proxy remains active until either side closes the connection.

OPTIONS

Command Line Options

The TELNET proxy recognizes the following command line options (whether started from the command line or from within */etc/rc.local*):

-daemon *port*
Indicates that the TELNET proxy runs as a daemon, and the port (name or number) on which the TELNET proxy listens.

-version
Displays version information for the TELNET proxy on stdout.

Configuration Options

The TELNET proxy reads configuration rules from the */usr/local/etc/netperm-table*. It reads all rules using the **tn-gw** and * (wildcard) keywords. If there are multiple rules in the table that could apply for a particular attribute, the TELNET proxy uses the first one that it finds. See **netperm-table**(5) for a more complete explanation of *netperm-table* syntax and precedence.

The TELNET proxy recognizes the following attributes:

authserver *host port*
Specifies the host running the authentication server (and the port on which it runs) that the proxies use for authenticating users.

host Specifies an IP address or host name.

port Specifies a service name or port number.

denial-msg *file*
Specifies the name of the file the TELNET proxy displays when it denies access to a user because they do not have permission to use the TELNET proxy. If no file is specified, the TELNET proxy generates a default message.

denydest-msg *file*
Specifies the name of the file the TELNET proxy displays when it denies access to a user because they are trying to access a destination that they are not permitted to access. If no file is specified, the TELNET proxy generates a default message.

directory *directory*
Specifies the directory that the TELNET proxy makes its root directory before providing service. This option is equivalent to the **chroot** option in previous versions.

groupid *group*
Specifies the name of the group the TELNET proxy uses when running.

group Specifies either a name or numeric id from the */etc/group* file.

help-msg *file*
Specifies the name of the file the TELNET proxy displays when the user accesses the help command. If no file is specified, the TELNET proxy displays a list of internal commands.

prompt *prompt*
Specifies the string the TELNET proxy uses as a prompt in command mode. Quotes are not required, but are recommended for strings that include spaces.

timeout *seconds*
Specifies the number of seconds the TELNET proxy is idle (with no network activity) before disconnecting

userid *user*
Specifies the user ID the proxy uses when running.

user Specifies either a name or numeric id from the */etc/passwd* file. This option is equivalent to the **-user** command in previous versions.

welcome-msg *file*
Specifies the name of the file the proxy displays as a welcome banner upon successful connection to the proxy. If no file is specified, the proxy generates a default message.

hosts host-pattern [host-pattern2...] [options]
rules specify host and access permissions. Typically, a hosts rule will be in the form of:

tn-gw: deny-hosts unknown
tn-gw: hosts 192.33.112.* 192.94.214.*

There may be several host patterns following the "hosts" keyword, ending with the first optional parameter beginning with '-'. Optional parameters are:

–dest pattern
–dest { pattern1 pattern2 ... }
specifies a list of valid destinations. If no list is specified, all destinations are considered valid. The -dest list is processed in order as it appears on the options line. -dest entries preceeded with a '!' character are treated as negation entries. Therefore the rule:

-dest !*.mit.edu -dest *

will permit hosts that are not in the domain "mit.edu" to be connected to.

−auth
specifies that the proxy should require a user to authenticate with a valid user-id prior to being permitted to use the gateway.

−passok
specifies that the proxy should permit users to change their passwords if they are connected from the designated host. Only hosts on a trusted network should be permitted to change passwords, unless token-type authenticators are distributed to all users.

xforwarder *program*
Specifies the location of the executable to which the TELNET proxy passes requests for the X proxy. Generally specifies the location of the X proxy.

Proxy Commands

connect *host* **[***port***]**
Connects to the specified host on the specified port.

host Specifies a host name or IP address.

port Specifies the name or port number, as specified in */etc/services.*

exit Exits the TELNET proxy.

help Displays a list of valid commands for the TELNET proxy.

password
Allows you to change your strong authentication password. The TELNET proxy only allows you to change your strong authentication password from the hosts listed in the **netperm-table**.

quit Exits the TELNET proxy.

timeout*seconds*
Specifies the number of seconds the TELNET proxy is idle (with no network activity) before disconnecting. Note that this setting overrides the setting in the *netperm-table*.

x-gw [*hostname***:***display***[.***screennumber***]]**
Invokes the X11 proxy to connect to the specified host and display. See **x-gw**(8) for more information on how the X11 proxy works.

? Displays a list of valid commands for the TELNET proxy.

EXAMPLES

This example shows the configuration lines in the *netperm-table* that sets the prompt for the proxy:

```
# set the prompt
tn-gw: prompt "Yoyodyne TELNET proxy > "
```

A common policy for the TELNET proxy is to authenticate all requests from untrusted networks to or through the firewall. This example shows a sample TELNET session from an untrusted network to a trusted network, using S/Key authentication at the firewall.

```
blaze.clientsite.com-28: telnet firewall.yoyodyne.com
Trying 204.255.154.100...
Connected to firewall.yoyodyne.com
Escape character is '^]'.
Username: scooter
Skey Challenge: s/key 651 fi19289 SAFE DUB RISK CUE YARD NIL
Login Accepted
firewall.yoyodyne.com telnet proxy (Version 3.1) ready:
tn-gw> c dimension
Trying 10.0.1.120 port 23...
```

Connected to dimension.yoyodyne.com

BSDI BSD/OS 2.0.1 (dimension) (ttyp5)

login: **scooter**
Password: **#########**

Welcome to dimension.yoyodyne.com
3:57PM up 16 days, 5:35, 4 users, load averages: 0.03, 0.01, 0.00
dimension-26:

FILES

/etc/rc.local
: Command script that controls automatic reboot, and includes startup information for the TELNET proxy.

/usr/local/etc/netperm-table
: The network permissions file contains configuration information for the Firewall Toolkit, including the TELNET proxy.

BUGS

Report bugs fwtk-support@tis.com or fwtk-users@tis.com. Include a complete example, explaining what you expected to happen and what actually happened. Be sure to indicate the type of system (operating system, hardware, etc.) you are using, as well as the version of the TELNET proxy.

AUTHOR

Trusted Information Systems, Inc.

SEE ALSO

netperm-table(5), **rc**(8), **authsrv**(8), **x-gw**(8)

NAME

x-gw – X11 proxy

SYNOPSIS

x-gw [-daemon *port***] [-version]**

DESCRIPTION

The Firewall Toolkit X11 proxy is an application level proxy that provides configurable access control. The X11 proxy, which runs on the firewall, passes X11 requests through the firewall (at the application level), using rules you supply. You can configure the proxies to allow connections based on:

- display name
- user

All packets, and therefore all application requests go to the firewall. On the firewall, the X11 proxy software relays information from one side of the firewall to the other. The proxy prevents the applications on outside networks from talking directly with the applications on your inside network, and vice versa. No IP packets pass from one side of the firewall to the other. All data is passed at the application level.

The default configuration for the X11 proxy allows both inside and outside hosts to start the X11 proxy.

Unlike some of the other toolkit proxies, the firewall does NOT run the X11 proxy as a daemon listening for display requests on the standard X11 port (6000). Instead, users must explicitly start the X11 proxy from either the TELNET or Rlogin proxy. The firewall logs and denies all requests for services on the standard X11 port.

To use the X11 proxy, a user TELNETs to the firewall, which runs the TELNET proxy. After checking permissions and authenticating users, the TELNET proxy displays a prompt for the user. At the prompt, the user indicates they wish to allow X displays across the firewall. The TELNET proxy starts the X11 proxy (**x-gw**) on port 6010 or higher. The X11 proxy checks its configuration information (in the *netperm-table*) and determines whether the initiating user has permission to use X11 services related to the desired display. If the host does not have permission, the X11 daemon logs the connection attempt and displays an error message.

If the user has permission, the proxy creates a "virtual display" on the firewall for the requesting client. When the outside X client requests access to the user's display, the virtual display server passes a query display to the X server on the display machine. This X server displays the query window on the real display, prompting the user to confirm the request. After the user confirms the request, the real X server receives the display information from the virtual X server. The proxy remains active until either end closes the connection.

OPTIONS

Command Line Options

The X11 proxy recognizes the following command line options (whether started from the command line or from within the TELNET or Rlogin proxies):

-version

Displays version information for the X11 proxy on stdout.

Configuration Options

The X11 proxy reads configuration rules from the */usr/local/etc/netperm-table*. It reads all rules using the **x-gw** and ***** (wildcard) keywords. The X11 proxy reads the *netperm-table* from top to bottom. If there are multiple rules in the table that could apply for a particular attribute, the X11 proxy uses the first one that it finds. See **netperm-table**(5) for a more complete explanation of *netperm-table* syntax and precedence.

The X11 proxy recognizes the following attributes:

directory *directory*

Specifies the directory that the X11 proxy makes its root directory before providing service. This option is equivalent to the **chroot** option in previous versions.

display *host***:***displaynumber***[.***screennumber***]**
Specifies the destination display on which applications display.

host Specifies the name of the machine to which the display is physically connected.

displaynumber
Specifies the number of the display on the machine.

screennumber
Specifies the number of the screen for the display.

groupid *group*
Specifies the name of the group the X11 proxy uses when running.

group Specifies either a name or numeric id from the */etc/group* file.

prompt *prompt*
Specifies the string the X11 proxy uses as a prompt in command mode. Quotes are not required, but are recommended for strings that include spaces.

timeout *seconds*
Specifies the number of seconds the X11 proxy is idle (with no network activity) before disconnecting

userid *user*
Specifies the user ID the proxy uses when running.

user Specifies either a name or numeric id from the */etc/passwd* file. This option is equivalent to the **-user** command in previous versions.

EXAMPLES

This example indicates that the X gateway displays all X applications on the display attached to dimension:

x-gw: display dimension:10.0

This example shows a user working on the inside network who needs to display information from a program running on a machine on an outside network.

dimension-27: **xhost +firewall**
dimension-28: **telnet firewall**
Trying 204.255.154.100...
Connected to firewall.yoyodyne.com
Escape character is '^]'.
firewall.yoyodyne.com telnet proxy (Version 3.1) ready:
tn-gw> **x**
tn-gw> display is firewall.yoyodyne.com: 10
tn-gw> **c blaze.clientsite.com**

Connecting to blaze.clientsite.com .... connected

HP-UX blaze A.09.01 E 9000/710 (ttys1)

login: **crawhide**
Password: #########

Please wait...checking for disk quotas You have mail.
blaze.clientsite.com-1:

After starting the X proxy, the user starts the X application:

blaze.clientsite.com-1: **setenv DISPLAY firewall.yoyodyne.com:10.0**
blaze.clientsite.com-2: **xclock &**

blaze.clientsite.com-3:

Finally, the user confirms the display request on his machine:

FILES

/etc/rc.local
: Command script that controls automatic reboot, and includes startup information for the X11 proxy.

/usr/local/etc/netperm-table
: The network permissions file contains configuration information for the Firewall Toolkit, including the X11 proxy.

BUGS

Report bugs to fwtk-support@tis.com or fwtk-users@tis.com. Include a complete example, explaining what you expected to happen and what actually happened. Be sure to indicate the type of system (operating system, hardware, etc.) you are using, as well as the version of the X11 proxy.

AUTHOR

Trusted Information Systems, Inc.

SEE ALSO

netperm-table(5), **rc**(8), **rlogin-gw**(8), **tn-gw**(8)