NAME

pam.conf – configuration file for pluggable authentication modules

SYNOPSIS

/etc/pam.conf

DESCRIPTION

pam.conf is the configuration file for the Pluggable Authentication Module architecture, or PAM. A PAM module provides functionality for one or more of four possible services: authentication, account management, session management, and password management. An authentication service module provides functionality to authenticate a user and set up user credentials. A account management module provides functionality to determine if the current user’s account is valid. This includes checking for password and account expiration, as well as verifying access hour restrictions. A session management module provides functionality to set up and terminate login sessions. A password management module provides functionality to change a user’s authentication token or password. Each of the four service modules can be implemented as a shared library object which can be referenced in the **pam.conf** configuration file.

Simplified PAM.CONF configuration file

The **pam.conf** file contains a listing of services. Each service is paired with a corresponding service module. When a service is requested, its associated module is invoked. Each entry has the following format:

<*service_name*> <*module_type*> <*control_flag*> <*module_path*> <*options*>

Below is an example of the **pam.conf** configuration file with support for authentication, account management, and session management modules.

```
login     auth      required    /usr/lib/security/pam_unix.so.1     debug
login     session   required    /usr/lib/security/pam_unix.so.1
login     account   required    /usr/lib/security/pam_unix.so.1
telnet    session   required    /usr/lib/security/pam_unix.so.1
other     auth      required    /usr/lib/security/pam_unix.so.1
other     passwd    required    /usr/lib/security/pam_unix.so.1
```

The *service_name* denotes the service (for example, **login**, **passwd**, or **rlogin**). The keyword, *other*, indicates the module all other applications which have not been specified should use. The *other* keyword can also be used if all services of the same *module_type* have the same requirements. In the example above, since all of the services use the same session module, they could have been replace by a single *other* line.

module_type denotes the service module type: authentication (*auth*), account management (*account*), session management (*session*), or password management (*password*).

The *control_flag* field determines the behavior of stacking, and will be discussed in more detail below.

The *module_path* field specifies the pathname to a shared library object which implements the service functionality. If the pathname is not absolute, it is assumed to be relative to **/usr/lib/security.**

The *options* field is used by the PAM framework layer to pass module specific options to the modules. It is up to the module to parse and interpret the options. This field can be used by the modules to turn on debugging or to pass any module specific parameters such as a TIMEOUT value. It can also be used to support unified login. The options supported by the modules are documented in their respective manual pages. For example, **pam_unix**(5) lists the options accepted by the UNIX module.

Integrating Multiple Authentication Services With Stacking

When a service_name of the same *module_type* is defined more than once, the service is said to be *stacked.* Each module referenced in the *module_path* for that service is then processed in the order that it occurs in the configuration file. The *control_flag* field specifies the continuation and failure semantics of the modules, and may be *required*, *optional*, or *sufficient*.

The PAM framework processes each service module in the stack. If all *required* modules in the stack succeed, then success is returned (*optional* and *sufficient* error values are ignored). If one or more *required* modules fail, then the error value from the first *required* module that failed is returned.

If none of the service modules in the stack are designated as *required,* then the PAM framework requires that at least one *optional* or *sufficient* module succeed. If all fail then the error value from the first service module in the stack is returned.

The only exception to the above is caused by the *sufficient* flag. If a service module that is designated as *sufficient* succeeds, then the PAM framework immediately returns success to the application (all subsequent services modules, even *required* ones, in the stack are ignored), given that all prior *required* modules had also succeeded. If a prior *required* module failed, then the error value from that module is returned.

If a module does not exist or can not be opened, then the **pam.conf** entry is ignored and an error will be logged through **syslog**(3) at the LOG_CRIT level.

Below is a sample configuration file that stacks the **login** and **rlogin** services.

```
login     auth     required     /usr/lib/security/pam_unix.so.1          debug
login     auth     optional     /usr/lib/security/pam_inhouse.so.1
rlogin    auth     sufficient   /usr/lib/security/pam_rhosts_auth.so.1
rlogin    auth     required     /usr/lib/security/pam_unix.so.1
```

In the case of **login,** the user is authenticated by the UNIX and inhouse authentication modules. The *required* keyword for *control_flag* requires that the user be allowed to login only if the user is authenticated by the UNIX service module. Inhouse authentication is optional by virtue of the *optional* keyword in the *control_flag* field. The user can still log in even if inhouse authentication fails.

In the case of **rlogin**, the *sufficient* keyword for *control_flag* specifies that if the *rhosts* authentication check succeeds, then PAM should return success to **rlogin** and **rlogin** should not prompt the user for a password. The UNIX authentication module (the next module in the stack) will only be invoked if the *rhosts* check fails. This gives the system administrator the flexibility to determine if *rhosts* alone is sufficient enough to authenticate a remote user.

Some modules may return PAM_IGNORE in certain situations. In these cases the PAM framework ignores the entire entry in **pam.conf** regardless of whether or not it is *required*, *optional* or *sufficient*.

NOTES

If an error is found in an entry due to invalid *service_name*, *module_type*, or *control_flag*, then the entry is ignored. If there are no valid entries for the given *module_type,* the PAM framework returns an error to the application.

EXAMPLES

The following is a sample pam.conf configuration file. Lines that begin with the # symbol are treated as comments, and therefore ignored.

```
#
# PAM configuration
#
# Authentication management for login service is stacked.
# Both UNIX and inhouse authentication functions are invoked.
login     auth     required     /usr/lib/security/pam_unix.so.1
login     auth     required     /usr/lib/security/pam_inhouse.so.1  try_first_pass
#
# Authentication management for rlogin service is stacked.
# If the rhost check succeeds, do not continue
rlogin      auth        sufficient   /usr/lib/security/pam_rhosts_auth.so.1
rlogin      auth        required     /usr/lib/security/pam_unix.so.1
#
# Other services use UNIX authentication
other       auth        required     /usr/lib/security/pam_unix.so.1
#
# Account management for login service is stacked.
# UNIX account management is required; inhouse account management is optional
login       account     required     /usr/lib/security/pam_unix.so.1
login       account     optional     /usr/lib/security/pam_inhouse.so.1
other       account     required     /usr/lib/security/pam_unix.so.1
#
# Session management
other       session     required     /usr/lib/security/pam_unix.so.1
#
# Password management
other       password    required     /usr/lib/security/pam_unix.so.1
```

Utilities and files

A list of utilities that are known to use PAM include: **login**, **passwd**, **su**, **dtlogin**, **rlogind**, **rshd**, **telnetd**, **ftpd**, **rpc.rexd**, **uucpd**, **init**, **sac**, and **ttymon**.

The PAM configuration file does not dictate either the name or the location of the service specific modules. The convention, however, is the following:

/usr/lib/security/pam_<service_name>_<module_name>.so.x
implements various function of specific authentication services.

/etc/pam.conf
configuration file

/usr/lib/libpam.so.1
implements the PAM framework library

SEE ALSO

dtlogin(1), **init**(1) **in.ftpd**(1M), **in.rexd**(1M), **in.rshd**(1M), **in.rlogind**(1M), **in.telnetd**(1M), **in.uucpd**(1), **login**(1), **passwd**(1), **sac**(1M), **su**(1M), **ttymon**(1M), **pam**(3)