The Linux-PAM System Administrators' Guide

Andrew G. Morgan, morgan@parc.power.net DRAFT v0.54 1996/11/18

This manual documents what a system-administrator needs to know about the **Linux-PAM** library. It covers the correct syntax of the PAM configuration file and discusses strategies for maintaining a secure system.

Contents

1 Introduction

Linux-PAM (Pluggable Authentication Modules for Linux) is a suite of shared libraries that enable the local system administrator to choose how applications authenticate users.

In other words, without (rewriting and) recompiling a PAM-aware application, it is possible to switch between the authentication mechanism(s) it uses. Indeed, one may entirely upgrade the local authentication system without touching the applications themselves.

Historically an application that has required a given user to be authenticated, has had to be compiled to use a specific authentication mechanism. For example, in the case of traditional UN*X systems, the identity of the user is verified by the user entering a correct password. This password, after being prefixed by a two character "salt", is encrypted (with crypt(3)). The user is then authenticated if this encrypted password is identical to the second field of the user's entry in the system password database (the `/etc/passwd` file). On such systems, most if not all forms of privileges are granted based on this single authentication scheme. Privilege comes in the form of a personal user-identifier (`uid`) and membership of various groups. Services and applications are available based on the personal and group identity of the user. Traditionally, group membership has been assigned based on entries in the `/etc/group` file.

Unfortunately, increases in the speed of computers and the widespread introduction of network based computing, have made once secure authentication mechanisms, such as this, vulnerable to attack. In the light of such realities, new methods of authentication are continuously being developed.

It is the purpose of the **Linux-PAM** project to separate the development of privilege granting software from the development of secure and appropriate authentication schemes. This is accomplished by providing a library of functions that an application may use to request that a user be authenticated. This PAM library is configured locally with a system file, `/etc/pam.conf`, to authenticate a user request via the locally available authentication modules. The modules themselves will usually be located in the directory `/usr/lib/security` and take the form of dynamically loadable object files (see `dlopen(3)`).

2 Some comments on the text

Before proceeding to read the rest of this document, it should be noted that the text assumes that certain files are placed in certain directories. Where they have been specified, the conventions we adopt here for locating these files are those of the relevant RFC (RFC-86.0, see 8). If you are using a distribution of Linux (or some other operating system) that supports PAM but chooses to distribute these files in a diferent way, you should be careful when copying examples directly from the text.

As an example of the above, the text assumes that PAM loadable object files (the *modules*) are to be located in the following directory: `/usr/lib/security/`. However, Red Hat Linux, in agreement with the Linux File System Standard (the FSSTND), places these files in `/lib/security`. Please be careful to perform the necessary transcription when using the examples from the text.

3 Overview

For the uninitiated, we begin by considering an example. We take an application that grants some service to users; *login* is one such program. *Login* does two things, it first establishes that the requesting user is whom they claim to be and second provides them with the requested service: in the case of *login* the service is a command shell running with the identity of the user.

Traditinally, the former step is achieved by the *login* application prompting the user for a password and then verifying that it agrees with that located on the system; hence verifying that the so far as the system is concerned the user is who they claim to be. This is the task that is delegated to **Linux-PAM**.

From the perspective of the application programmer (in this case the person that wrote the *login* application), **Linux-PAM** takes care of the task of verifying the identity of the user.

The flexibility of **Linux-PAM** is that *you*, the system administrator, have the freedom to stipulate which authentication scheme is to be used. You have the freedom to set the scheme for any/all PAM-aware applications on your Linux system. That is, you can authenticate from anything as naive as *simple trust* (`pam_permit`) to something as paranoid as a combination of a retinal scan, a voice print and a one-time password!

To illustrate the flexibility you face, consider the following situation: a system administrator (parent) wishes to improve the mathematical ability of her users (children). She can configure their favorite "Shoot 'em up game" (PAM-aware of course) to authenticate them with a request for the product of a couple of random numbers less than 12. It is clear that if the game is any good they will soon learn their *multiplication tables*. As they mature, the authentication can be upgraded to include (long) division!

Linux-PAM deals with four separate types of (management) task. These are: *authentication management*; *account management*; *session management*; and *password management*. The association of the preferred management scheme with the behavior of an application is made with entries in the **Linux-PAM** configuration file, `/etc/pam.conf`.

The management functions are performed by *modules* specified in the `/etc/pam.conf` file. The syntax for this file is discussed in the next section.

Here is a figure that describes the overall organization of **Linux-PAM**.

```
+----------------+
| application: X |
+----------------+       /  +----------+     +================+
| authentication-[---->--\--] Linux-   |--<--| /etc/pam.conf  |
|       +        [----<--/--]   PAM    |     |================|
|[conversation()][--+    \  |          |     | X auth .. a.so |
+----------------+  |    /  +-n--n-----+     | X auth .. b.so |
|                |  |       __|  |           |           _____/
|  service user  |  A      |     |           |____,-----'
|                |  |      V     A
+----------------+  +------|-----|---------+ -----+------+
                       +---u-----u----+    |      |      |
                       |   auth....   |--[ a ]--[ b ]--[ c ]
                       +--------------+
                       |   acct....   |--[ b ]--[ d ]
                       +--------------+
                       |   password   |--[ b ]--[ c ]
                       +--------------+
                       |   session    |--[ e ]--[ c ]
                       +--------------+
```

By way of explanation, the left of the figure represents the application; application X. Such an application interfaces with the **Linux-PAM** library and knows none of the specifics of its configured authentication method. The **Linux-PAM** library (in the center) consults the contents of the `/etc/pam.conf` file and loads the modules that are appropriate for application-X. These modules fall into one of four management groups (lower-center) and are stacked in the order they appear in the configuaration file. These modules, when called by **Linux-PAM**, perform the various authentication tasks for the application. Textual information, required from/or offered to the user, can be exchanged through the use of the application-supplied *conversation* function.

4 The Linux-PAM configuration file

Linux-PAM is designed to provide the system administrator with a great deal of flexibility in configuring the privilege granting applications of their system. The local configuration of those aspects of system security controlled by `Linux-PAM` is contained in the system file, `/etc/pam.conf`. In this section we discuss the correct syntax of and generic options respected by entries to this file.

4.1 Configuration file syntax

The reader should note that the **Linux-PAM** specific tokens in this file are case *insensitive*. The module paths, however, are case sensitive since they indicate a file's *name* and reflect the case dependence of typical Linux file-systems. The case-sensitivity of the arguments to any given module is defined for each module in turn.

In addition to the lines described below, there are two *special* characters provided for the convenience of the system administrator: comments are preceded by a '`#`' and extend to the next end-of-line; also, module specification lines may be extended with a '\'.

A general configuration line of the `/etc/pam.conf` file has the following form:

```
service-name   module-type   control-flag   module-path   arguments
```

Here we explain the meaning of each of these tokens.

`service-name`

The name of the service associated with this entry. Frequently the service name is the conventional name of the given application. For example, '`ftpd`', '`rlogind`' and '`su`', *etc.* .

There is a special `service-name`, reserved for defining a default authentication mechanism. It has the name '`OTHER`' and may be specified in either lower or upper case characters. Note, when there is a module specified for a named service, the '`OTHER`' entries are ignored.

`module-type`

One of (currently) four types of module. The four types are as follows:

- `auth`; this module type provides two aspects of authenticating the user. Firstly, it establishes that the user is who they claim to be, by instructing the application to prompt the user for a password or other means of identification. Secondly, the module can grant `group` membership (independently of the `/etc/groups` file discussed above) or other privileges through its *credential* granting properties.
- `account`; this module performs non-authentication based account management. It is typically used to restrict/permit access to a service based on the time of day, currently available system resources (maximum number of users) or perhaps the location of the applicant user—'`root`' login only on the console.
- `session`; primarily, this module is associated with doing things that need to be done for the user before/after they can be given service. Such things include the logging of information concerning the opening/closing of some data exchange with a user, mounting directories, etc. .
- `password`; this last module type is required for updating the authentication token associated with the user. Typically, there is one module for each 'challenge/response' based authentication (`auth`) module-type.

`control-flag`

One of (currently) three tokens that indicate the severity of concern associated with the success or failure of a given module. **Linux-PAM** provides for the *stacking* of similar modules, providing a method of simultaneously exposing the user to more than one authentication mechanism. The application is not made aware of the individual success or failure of modules listed in the '`/etc/pam.conf`' file. Instead, it receives a summary *success* or *fail* response from the **Linux-PAM** library. The order of execution of these modules is that of the entries in the `/etc/pam.conf` file; earlier entries are executed before later ones.

The policy for determining this response is based on these three `control-flags`:

- `required`; this indicates that the success of the module is required for the `module-type` facility to succeed. Failure of this module will not be apparent to the user until all of the remaining modules (of the same `module-type`) have been executed.
- `sufficient`; the success of this module is deemed '*sufficient*' to satisfy the **Linux-PAM** library that this module-type has succeeded in its purpose. In the event that no previous `required` module has failed, no more '*stacked*' modules of this type are invoked. (Note, in this case subsequent `required` modules are **not** invoked.). A failure of this module is not deemed as fatal to satisfying the application that this `module-type` has succeeded.
- `optional`; as its name suggests, this `control-flag` marks the module as not being critical to the success or failure of the user's application for service. However, in the absence of any successes of previous or subsequent stacked modules this module will determine the nature of the response to the application.

`module-path`

The path-name of the dynamically loadable object file; *the pluggable module* itself.

`args`

The `args` are a list of tokens that are passed to the module when it is invoked. Much like arguments to a typical Linux shell command. Generally, valid arguments are optional and are specific to any given module. Invalid arguments are ignored by a module, however, when encountering an invalid argument, the module is required to write an error to `syslog(3)`. For a list of *generic* options see the next section.

Any line, in `/etc/pam.conf`, that is not formatted correctly is ignored by **Linux-PAM**; however, a corresponding error is written to the system log files with a call to `syslog(3)`.

4.2 Generic optional arguments

The following are optional arguments which are likely to be understood by any module. Arguments (including these) are in general *optional*.

`debug`

Use the `syslog(3)` call to log debugging information to the system log files.

`no_warn`

Instruct module to not give warning messages to the application.

`use_first_pass`

The module should not prompt the user for a password. Instead, it should obtain the previously typed password (from the preceding `auth` module), and use that. If that doesn't work, then the user will not be authenticated. (This option is intended for `auth` and `password` modules only).

`try_first_pass`

The module should attempt authentication with the previously typed password (from the preceding `auth` module). If that doesn't work, then the user is prompted for a password. (This option is intended for `auth` modules only).

`use_mapped_pass`

This argument is not currently supported by any of the modules in the **Linux-PAM** distribution because of possible consequences associated with U.S. encryption exporting restrictions. Within the U.S., module developers are, of course, free to implement it (as are developers in other countries). For compatibility reasons we describe its use as suggested in the **DCE-RFC 86.0**, see section 8 for a pointer to this document.

The `use_mapped_pass` argument instructs the module to take the clear text authentication token entered by a previous module (that requests such a token) and use it to generate an encryption/decryption key with which to safely store/retrieve the authentication token required for this module. In this way the user can enter a single authentication token and be quietly authenticated by a number of stacked modules. Obviously a convenient feature that necessarily requires some reliably strong encryption to make it secure. This argument is intended for the `auth` and `password` module types only.

4.3 An example `/etc/pam.conf` file

In this section, we give some examples of entries that can be present in the **Linux-PAM** configuration file. As a first attempt at configuring your system you could do worse than to implement these.

4.3.1 Default policy

If a system is to be considered secure, it had better have a reasonably secure '`OTHER`' entry. The following is a paranoid setting (which is not a bad place to start!):

```
#
# default; deny access
#

OTHER   auth     required       /usr/lib/security/pam_deny.so
OTHER   account  required       /usr/lib/security/pam_deny.so
OTHER   password required       /usr/lib/security/pam_deny.so
OTHER   session  required       /usr/lib/security/pam_deny.so
```

Whilst fundamentally a secure default, this is not very sympathetic to a misconfigured system. For example, such a system is vulnerable to locking everyone out should the rest of the file become badly written.

The module pam_deny (documented in a later section) is not very sophisticated. For example, it logs no information when it is invoked so unless the users of a system contact the administrator when failing to execute a service application, the administrator may go for a long while in ignorance of the fact that his system is misconfigured.

The addition of the following line before those in the above example would provide a suitable warning to the administrator.

```
#
# default; wake up! This application is not configured
#

OTHER   auth     required       /usr/lib/security/pam_warn.so
OTHER   password required       /usr/lib/security/pam_warn.so
```

Having two "OTHER auth" lines is an example of stacking.

On a less sensitive computer, one on which the system administrator wishes to remain ignorant of the power of Linux-PAM, the following selection of lines is likely to mimic the historically familiar Linux setup.

```
#
# default; standard UNIX access
#

OTHER   auth     required       /usr/lib/security/pam_unix_auth.so
OTHER   account  required       /usr/lib/security/pam_unix_acct.so
OTHER   password required       /usr/lib/security/pam_unix_passwd.so
OTHER   session  required       /usr/lib/security/pam_unix_sess.so
```

In general this will provide a starting place for most applications. Unfortunately, most is not all. One application that might require additional lines is *ftpd* if you wish to enable *anonymous-ftp*.

To enable anonymous-ftp, the following lines might be used to replace the default (OTHER) ones. (*WARNING* as of 1996/11/14 this does not work correctly with any ftpd. Consequently, this description may be subject to change or the application will be fixed.)

```
#
# default; add ftp-specifics. These lines enable anonymous ftp over
#          standard UNIX access (the listfile entry blocks access to
#          users listed in /etc/ftpusers)
#

ftpd    auth    sufficient  /usr/lib/security/pam_ftp.so
ftpd    auth    required    /usr/lib/security/pam_unix_auth.so use_first_pass
ftpd    auth    required    /usr/lib/security/pam_listfile.so \
                        onerr=succeed item=user sense=deny file=/etc/ftpusers
```

Note, the second line is necessary since the default entries are ignored by a service application (here *ftpd*) if there are *any* entries in /etc/pam.conf for that specified service. Again, this is an example of authentication module stacking. Note the use of the `sufficient` control-flag. It says that "if this module authenticates the user, ignore the subsequent `auth` modules". Also note the use of the "`use_first_pass`" module-argument, this instructs the UNIX authentication module that it is not to prompt for a password but rely one already having been obtained by the ftp module.

The standard UNIX modules used above are strongly tied to using the default '`libc`' user database functions (see for example, `man getpwent`). It is the opinion of the author that these functions are not sufficently flexible to make full use of the power of **Linux-PAM**. For this reason, and as a small plug, I mention in passing that there is a pluggable replacement for the `pam_unix_..` modules; `pam_pwdb`. See the section below for a more complete description.

5 Security issues of Linux-PAM

This section will discuss good practices for constructing a secure `/etc/pam.conf` file. *It is currently sadly lacking...*

It is not a good thing to have a weak default (`OTHER`) entry.

6 A reference guide for available modules

Here, we collect together some descriptions of the various modules available for **Linux-PAM**. In general these modules should be freely available. Where this is not the case, it will be indicated.

Also please note the comments contained in the section 2 when copying the examples listed below.

6.1 Chroot

6.1.1 Synopsis

Module Name:

`pam_chroot`

Author:

Bruce Campbell <brucec@humbug.org.au>

Maintainer:

Author; proposed on 20/11/96 - email for status

Management groups provided:

account; session; authentication

Cryptographically sensitive:

Security rating:

Clean code base:

Unwritten.

System dependencies:

Network aware:

Expects localhost.

6.1.2 Overview of module

This module is intended to provide a transparent wrapper around the average user, one that puts them in a fake file-system (eg, their '/' is really `/some/where/else`).

Useful if you have several classes of users, and are slightly paranoid about security. Can be used to limit who else users can see on the system, and to limit the selection of programs they can run.

6.1.3 Account component:

Need more info here.

6.1.4 Authentication component:

Need more info here.

6.1.5 Session component:

Need more info here.

Recognized arguments:

Arguments and logging levels for the PAM version are being worked on.

Description:

Examples/suggested usage:

Do provide a reasonable list of programs - just tossing 'cat', 'ls', 'rm', 'cp' and 'ed' in there is a bit...

Don't take it to extremes (eg, you can set up a separate environment for each user, but its a big waste of your disk space.)

6.2 Cracklib pluggable password strength-checker

6.2.1 Synopsis

Module Name:

pam_cracklib

Author:

Cristian Gafton <gafton@sorosis.ro>

Maintainer:

Author.

Management groups provided:

password

Cryptographically sensitive:

Security rating:

Clean code base:

System dependencies:

Requires the system library `libcrack` and a system dictionary: `/usr/lib/cracklib_dict`.

Network aware:

6.2.2 Overview of module

This module can be plugged into the `password` stack of a given application to provide some plug-in strength-checking for passwords. (XXX - note this does not necessarily work with the pam_unix module, although it is known to work with the pam_pwdb replacement for the unix module – see example and pam_pwdb write up for more information).

6.2.3 Password component

Recognized arguments:

`debug`; `type=XXX`; `retry=N`

Description:

The action of this module is to prompt the user for a password and check its strength against a system dictionary and a set of rules for identifying poor choices.

The default action is to prompt for a single password, check its strength and then, if it is considered strong, prompt for the password a second time (to verify that it was typed correctly on the first occasion). All being well, the password is passed on to subsequent modules to be installed as the new authentication token.

The default action may be modified in a number of ways using the arguments recognized by the module:

- `debug` -
  this option makes the module write information to syslog(3) indicating the behavior of the module (this option does **not** write password information to the log file).

- `type=XXX` -

  the default action is for the module to use the following prompts when requesting passwords: "New UNIX password: " and "Retype UNIX password: ". Using this option you can replace the word UNIX with `XXX`.

- `retry=N` -

  the default number of times this module will request a new password (for strength-checking) from the user is 1. Using this argument this can be increased to `N`.

Examples/suggested usage:

(At the time of writing, this module can only be stacked before the `pam_pwdb` module. Cracklib strength checking may be compiled by default into the `pam_unix` module.)

For an example of the use of this module, we show how it may be stacked with the password component of `pam_pwdb`:

```
#
# These lines stack two password type modules. In this example the user
# is given 3 opportunities to enter a strong password. The
# "use_authtok" argument ensures that the pam_pwdb module does not
# prompt for a password, but instead uses the one provided by
# pam_cracklib.
#
passwd  password required       /usr/lib/security/pam_cracklib.so retry=3
passwd  password required       /usr/lib/security/pam_pwdb.so use_authtok
```

6.3 The locking-out module

6.3.1 Synopsis

Module Name:

pam_deny

Author:

Andrew G. Morgan <morgan@parc.power.net>

Maintainer:

current **Linux-PAM** maintainer

Management groups provided:

account; authentication; password; session

Cryptographically sensitive:

Security rating:

Clean code base:

clean.

System dependencies:

Network aware:

6.3.2 Overview of module

This module can be used to deny access. It always indicates a failure to the application through the PAM framework. As is commented in the overview section 3, this module might be suitable for using for default (the `OTHER`) entries.

6.3.3 Account component

Recognized arguments:

Description:

This component does nothing other than return a failure. The failure type is `PAM_ACCT_EXPIRED`.

Examples/suggested usage:

Stacking this module with type `account` will prevent the user from gaining access to the system via applications that refer to **Linux-PAM**'s account management function `pam_acct_mgmt()`.

The following example would make it impossible to login:

```
#
# add this line to your other login entries to disable all accounts
#
login   account  required       /usr/lib/security/pam_deny.so
```

6.3.4 Authentication component

Recognized arguments:

Description:

This component does nothing other than return a failure. The failure type is `PAM_AUTH_ERR` in the case that `pam_authenticate()` is called (when the application tries to authenticate the user), and is `PAM_CRED_UNAVAIL` when the application calls `pam_setcred()` (to establish and set the credentials of the user – it is unlikely that this function will ever be called in practice).

Examples/suggested usage:

To deny access to default applications with this component of the `pam_deny` module, you might include the following line in your **Linux-PAM** configuration file:

```
#
# add this line to your existing OTHER entries to prevent authentication
# succeeding with default applications.
#
OTHER   auth     required       /usr/lib/security/pam_deny.so
```

6.3.5 Password component

Recognized arguments:

Description:

This component of the module denies the user the opportunity to change their password. It always responds with `PAM_AUTHTOK_ERR` when invoked.

Examples/suggested usage:

This module should be used to prevent an application from updating the applicant user's password. For example, to prevent `login` from automatically prompting for a new password when the old one has expired you should include the following line in your configuration file:

```
#
# add this line to your other login entries to prevent the login
# application from being able to change the user's password.
#
login   password required       /usr/lib/security/pam_deny.so
```

6.3.6 Session component

Recognized arguments:

Description:

This aspect of the module prevents an application from starting a session on the host computer.

Examples/suggested usage:

Together with another session module, that displays a message of the day perhaps (XXX - such a module needs to be written), this module can be used to block a user from starting a shell. Given the presence of a `pam_motd` module, we might use the following entries in the configuration file to inform the user it is system time:

```
#
# An example to see how to configure login to refuse the user a
# session (politely)
#
login   session  required       /usr/lib/security/pam_motd.so \
                                        file="/etc/system_time"
login   session  required       /usr/lib/security/pam_deny.so
```

6.4 The filter module

6.4.1 Synopsis

Module Name:

pam_filter

Author:

Andrew G. Morgan <morgan@parc.power.net>

Maintainer:

Author.

Management groups provided:

account; authentication; password; session

Cryptographically sensitive:

Not yet.

Security rating:

Clean code base:

This module compiles cleanly on Linux based systems.

System dependencies:

To function it requires *filters* to be installed on the system.

Network aware:

6.4.2 Overview of module

This module was written to offer a plug-in alternative to programs like ttysnoop (XXX - need a reference). Since writing a filter that performs this function has not occurred, it is currently only a toy. The single filter provided with the module simply transposes upper and lower case letters in the input and output streams. (This can be very annoying and is not kind to termcap based editors).

6.4.3 Account+Authentication+Password+Session components

Recognized arguments:

`debug`; `new_term`; `non_term`; `runX`

Description:

Each component of the module has the potential to invoke the desired filter. The filter is always `execv(2)`d with the privilege of the calling application and **not** that of the user. For this reason it cannot usually be killed by the user without closing their session.

The behavior of the module can be significantly altered by the arguments passed to it in the **Linux-PAM** configuration file:

- `debug` -
  this option increases the amount of information logged to `syslog(3)` as the module is executed.

- `new_term` -

  the default action of the filter is to set the `PAM_TTY` item to indicate the terminal that the user is using to connect to the application. This argument indicates that the filter should set `PAM_TTY` to the filtered pseudo-terminal.

- `non_term` - don't try to set the `PAM_TTY` item.

- `runX` -

  in order that the module can invoke a filter it should know when to invoke it. This argument is required to tell the filter when to do this. The arguments that follow this one are respectively the full pathname of the filter to be run and any command line arguments that the filter might expect.

  Permitted values for `X` are `1` and `2`. These indicate the precise time the that filter is to be run. To explain this concept it will be useful to have read the Linux-PAM Module developer's guide. Basically, for each management group there are up to two ways of calling the module's functions.

  In the case of the *authentication* and *session* components there are actually two separate functions. For the case of authentication, these functions are `_authenticate` and `_setcred` – here `run1` means run the filter from the `_authenticate` function and `run2` means run the filter from `_setcred`. In the case of the session modules, `run1` implies that the filter is invoked at the `_open_session` stage, and `run2` for `_close_session`.

  For the case of the account component. Either `run1` or `run2` may be used.

  For the case of the password component, `run1` is used to indicate that the filter is run on the first occasion `_chauthtok` is run (the `PAM_PRELIM_CHECK` phase) and `run2` is used to indicate that the filter is run on the second occasion (the `PAM_UPDATE_AUTHTOK` phase).

Examples/suggested usage:

At the time of writing there is little real use to be made of this module. For fun you might try adding the following line to your login's configuration entries

```
#
# An example to see how to configure login to transpose upper and
# lower case letters once the user has logged in(!)
#
login   session  required       /usr/lib/security/pam_filter.so \
                                run1 /usr/sbin/pam_filter/upperLOWER
```

6.5 Anonymous access module

6.5.1 Synopsis

Module Name:

`pam_ftp.so`

Author:

Andrew G. Morgan <morgan@parc.power.net>

Maintainer:

Author.

Management groups provided:

authentication

Cryptographically sensitive:

Security rating:

Clean code base:

System dependencies:

Network aware:

prompts for email address of user; easily spoofed (XXX - needs work)

6.5.2 Overview of module

The purpose of this module is to provide a pluggable anonymous ftp mode of access.

6.5.3 Authentication component

Recognized arguments:

`debug`; `users=XXX,YYY,...`; `ignore`

Description:

This module intercepts the user's name and password. If the name is "`ftp`" or "`anonymous`", the user's password is broken up at the '`@`' delimiter into a `PAM_RUSER` and a `PAM_RHOST` part; these pam-items being set accordingly. The username is set to "`ftp`". In this case the module succeeds. Alternatively, the module sets the `PAM_AUTHTOK` item with the entered password and fails.

The behavior of the module can be modified with the following flags:

- `debug` - log more information to with `syslog(3)`.
- `users=XXX,YYY,...` - instead of "`ftp`" or "`anonymous`", provide anonymous login to the comma separated list of users; "`XXX,YYY,...`". Should the applicant enter one of these usernames the returned username is set to the first in the list; "`XXX`".
- `ignore` - pay no attention to the email address of the user (if supplied).

Examples/suggested usage:

An example of the use of this module is provided in the configuration file section 4. With care, this module could be used to provide new/temporary account anonymous login.

6.6 The group access module

6.6.1 Synopsis

Module Name:

pam_group

Author:

Andrew G. Morgan <morgan@parc.power.net>

Maintainer:

Author.

Management groups provided:

authentication

Cryptographically sensitive:

Security rating:

Sensitive to *setgid* status of file-systems accessible to users.

Clean code base:

System dependencies:

Requires an `/etc/security/group.conf` file.

Network aware:

Only through correctly set `PAM_TTY` item.

6.6.2 Overview of module

This module provides group-settings based on the user's name and the terminal they are requesting a given service from. It takes note of the time of day.

6.6.3 Authentication component

Recognized arguments:

Description:

This module does not authenticate the user, but instead it grants group memberships (in the credential setting phase of the authentication module) to the user. Such memberships are based on the service they are applying for. The group memberships are listed in text form in the `/etc/security/group.conf` file.

Examples/suggested usage:

For this module to function correctly there must be a correctly formatted `/etc/security/groups.conf` file present. The format of this file is as follows. Group memberships are given based on the service application satisfying any combination of lines in the configuration file. Each line (barring comments which are preceded by '#' marks) has the following syntax:

```
services  ;  ttys  ;  users  ;  times  ;  groups
```

Here the first four fields share the syntax of the `pam_time` configuration file; `/etc/security/pam_time.conf`, and the last field, the `groups` field, is a comma (or space) separated list of the text-names of a selection of groups. If the users application for service satisfies the first four fields, the user is granted membership of the listed groups.

As stated in above this module's usefulness relies on the file-systems accessible to the user. The point being that once granted the membership of a group, the user may attempt to create a *setgid* binary with a restricted group ownership. Later, when the user is not given membership to this group, they can recover group membership with the precompiled binary. The reason that the file-systems that the user has access to are so significant, is the fact that when a system is mounted *nosuid* the user is unable to create or execute such a binary file. For this module to provide any level of security, all file-systems that the user has write access to should be mounted *nosuid*.

At this time, the `pam_group` *module does not fuction in parallel with the* `/etc/group` *file. If the user is granted any groups based on the behavior of this module, the user's entries in* `/etc/group` *are ignored. This will be fixed in the future.*

6.7 The Kerberos 4 module.

6.7.1 Synopsis

Module Name:

`pam_krb4`

Author:

Derrick J. Brashear <shadow@dementia.org>

Maintainer:

Author.

Management groups provided:

authentication; password; session

Cryptographically sensitive:

uses API

Security rating:

Clean code base:

System dependencies:

libraries - `libkrb`, `libdes`, `libcom_err`, `libkadm`; and a set of Kerberos include files.

Network aware:

Gets Kerberos ticket granting ticket via a Kerberos key distribution center reached via the network.

6.7.2 Overview of module

This module provides an interface for doing Kerberos verification of a user's password, getting the user a Kerberos ticket granting ticket for use with the Kerberos ticket granting service, destroying the user's tickets at logout time, and changing a Kerberos password.

6.7.3 Session component

Recognized arguments:

Description:

This component of the module currently sets the user's `KRBTKFILE` environment variable (although there is currently no way to export this), as well as deleting the user's ticket file upon logout (until `PAM_CRED_DELETE` is supported by *login*).

Examples/suggested usage:

This part of the module won't be terribly useful until we can change the environment from within a `Linux-PAM` module.

6.7.4 Password component

Recognized arguments:

`use_first_pass`; `try_first_pass`

Description:

This component of the module changes a user's Kerberos password by first getting and using the user's old password to get a session key for the password changing service, then sending a new password to that service.

Examples/suggested usage:

This should only be used with a real Kerberos v4 `kadmind`. It cannot be used with an AFS kaserver unless special provisions are made. Contact the module author for more information.

6.7.5 Authentication component

Recognized arguments:

`use_first_pass`; `try_first_pass`

Description:

This component of the module verifies a user's Kerberos password by requesting a ticket granting ticket from the Kerberos server and optionally using it to attempt to retrieve the local computer's host key and verifying using the key file on the local machine if one exists.

It also writes out a ticket file for the user to use later, and deletes the ticket file upon logout (not until `PAM_CRED_DELETE` is called from *login*).

Examples/suggested usage:

This module can be used with a real Kerberos server using MIT v4 Kerberos keys. The module or the system Kerberos libraries may be modified to support AFS style Kerberos keys. Currently this is not supported to avoid cryptography constraints.

6.8 The resource limits module

6.8.1 Synopsis

Module Name:

`pam_limits`

Authors:

Cristian Gafton <gafton@sorosis.ro>
Thanks are also due to Elliot Lee <sopwith@redhat.com> for his comments on improving this module.

Maintainer:

Cristian Gafton - 1996/11/20

Management groups provided:

session

Cryptographically sensitive:

Security rating:

Clean code base:

System dependencies:

requires an `/etc/security/limits.conf` file and kernel support for resource limits. Also uses the library, `libpwdb`.

Network aware:

6.8.2 Overview of module

This module, through the **Linux-PAM** *open*-session hook, sets limits on the system resources that can be obtained in a user-session. Its actions are dictated more explicitly through the configuration file discussed below.

6.8.3 Session component

Recognized arguments:

`debug`; `conf=/path/to/file.conf`

Description:

Through the contents of the configuration file, `/etc/security/limits.conf`, resource limits are placed on users' sessions. Users of `uid=0` are not affected by this restriction.

The behavior of this module can be modified with the following arguments:

- `debug` - verbose logging to `syslog(3)`.
- `conf=/path/to/file.conf` - indicate an alternative *limits* configuration file to the default.

Examples/suggested usage:

In order to use this module the system administrator must first create a *root-only-readable* file (default is `/etc/security/limits.conf`). This file describes the resource limits the superuser wishes to impose on users and groups. No limits are imposed on `uid=0` accounts.

Each line of the configuration file describes a limit for a user in the form:

```
<domain>        <type>  <item>        <value>
```

The fields listed above should be filled as follows...
<`domain`> can be:

- a username
- a groupname, with `@group` syntax
- the wild-card `*`, for default entry

<`type`> can have the two values:

- `hard` for enforcing *hard* resource limits. These limits are set by the superuser and enforced by the Linux Kernel. The user cannot raise his requirement of system resources above such values.
- `soft` for enforcing *soft* resource limits. These limits are ones that the user can move up or down within the permitted range by any pre-exisiting *hard* limits. The values specified with this token can be thought of as *default* values, for normal system usage.

<`item`> can be one of the following:

- `core` - limits the core file size (KB)
- `data` - max data size (KB)
- `fsize` - maximum filesize (KB)
- `memlock` - max locked-in-memory address space (KB)
- `nofile` - max number of open files
- `rss` - max resident set size (KB)
- `stack` - max stack size (KB)

- `cpu` - max CPU time (MIN)
- `nproc` - max number of processes
- `as` - address space limit
- `maxlogins` - max number of logins for this user.

To completely disable limits for a user (or a group), a single dash (-) will do (Example: "`bin -`", "`@admin -`"). Please remember that individual limits have priority over group limits, so if you impose no limits for `admin` group, but one of the members in this group have a limits line, the user will have its limits set according to this line.

Also, please note that all limit settings are set *per login*. They are not global, nor are they permanent; existing only for the duration of the session.

In the *limits* configuration file, the "`#`" character introduces a comment - after which the rest of the line is ignored.

The `pam_limits` module does its best to report configuration problems found in its configuration file via `syslog(3)`.

The following is an example configuration file:

```
# EXAMPLE /etc/security/limits.conf file:
# =======================================
# <domain>      <type>  <item>          <value>
*               soft    core            0
*               hard    rss             10000
@student        hard    nproc           20
@faculty        soft    nproc           20
@faculty        hard    nproc           50
ftp             hard    nproc           0
@student        -       maxlogins       4
```

Note, the use of `soft` and `hard` limits for the same resource (see `@faculty`) – this establishes the *default* and permitted *extreme* level of resources that the user can can obtain in a given service-session.

For the services that need resources limits (login for example) put a the following line in `/etc/pam.conf` as the last line for that service (usually after the pam_unix session line:

```
#
# Resource limits imposed on login sessions via pam_limits
#
login   session    required     /usr/lib/security/pam_limits.so
```

6.9 The list-file module

6.9.1 Synopsis

Module Name:

`pam_listfile`

Author:

Elliot Lee <sopwith@cuc.edu>

Maintainer:

Red Hat Software:
Michael K. Johnson <johnsonm@redhat.com> 1996/11/18
(if unavailable, contact Elliot Lee <sopwith@cuc.edu>).

Management groups provided:

authentication

Cryptographically sensitive:

Security rating:

Clean code base:

1 warning; discards "`const`" from pointer when `free()`'ing

System dependencies:

Network aware:

6.9.2 Overview of module

The list-file module provides a way to deny or allow services based on an arbitrary file.

6.9.3 Authentication component

Recognized arguments:

`onerr=succeed|fail`; `sense=allow|deny`; `file=`*filename*; `item=user|tty|rhost|ruser`

Description:

The module gets the item of the type specified – `user` specifies the username, `PAM_USER`; tty specifies the name of the terminal over which the request has been made, `PAM_TTY`; rhost specifies the name of the remote host (if any) from which the request was made, `PAM_RHOST`; and ruser specifies the name of the remote user (if available) who made the request, `PAM_RUSER` – and looks for an instance of that item in the file *filename*. *filename* contains one line per item listed. If the item is found, then if `sense=allow`, `PAM_SUCCESS` is returned, causing the authorization request to succeed; else if `sense=deny`, `PAM_AUTH_ERR` is returned, causing the authorization request to fail.

If an error is encountered (for instance, if *filename* does not exist, or a poorly-constructed argument is encountered), then if `onerr=succeed`, `PAM_SUCCESS` is returned, otherwise if `onerr=fail`, `PAM_AUTH_ERR` or `PAM_SERVICE_ERR` (as appropriate) will be returned.

All arguments should be specified; do not count on any default behavior, as it is subject to change.

No credentials are awarded by this module.

Examples/suggested usage:

Classic "ftpusers" authentication can be implemented with this entry in `/etc/pam.conf`:

```
#
# deny ftp-access to users listed in the /etc/ftpusers file
#
ftp     auth     required       /usr/lib/security/pam_listfile.so \
                        onerr=succeed item=user sense=deny file=/etc/ftpusers
```

Note, users listed in `/etc/ftpusers` file are (counterintuitively) **not** allowed access to the ftp service.

To allow login access only for certain users, you can use an pam.conf entry like this:

```
#
# permit login to users listed in /etc/loginusers
#
login   auth     required       /usr/lib/security/pam_listfile.so \
                        onerr=fail item=user sense=allow file=/etc/loginusers
```

For this example to work, all users who are allowed to use the login service should be listed in the file `/etc/loginusers`. Unless you are explicitly trying to lock out root, make sure that when you do this, you leave a way for root to log in, either by listing root in `/etc/loginusers`, or by listing a user who is able to *su* to the root account.

6.10 The no-login module

6.10.1 Synopsis

Module Name:

`pam_nologin`

Author:

Written by Michael K. Johnson <johnsonm@redhat.com>
(based on code taken from a module written by Andrew G. Morgan <morgan@parc.power.net>).

Maintainer:

Michael K. Johnson <johnsonm@redhat.com>

Management groups provided:

authentication

Cryptographically sensitive:

Security rating:

Clean code base:

System dependencies:

Network aware:

6.10.2 Overview of module

Provides standard Unix *nologin* authentication.

6.10.3 Authentication component

Recognized arguments:

Description:

Provides standard Unix *nologin* authentication. If the file `/etc/nologin` exists, only root is allowed to log in; other users are turned away with an error message. All users (root or otherwise) are shown the contents of `/etc/nologin`.

If the file `/etc/nologin` does not exist, this module succeeds silently.

Examples/suggested usage:

In order to make this module effective, all login methods should be secured by it. It should be used as a `required` method listed before any `sufficient` methods in order to get standard Unix nologin semantics.

6.11 The promiscuous module

6.11.1 Synopsis

Module Name:

pam_permit

Author:

Andrew G. Morgan, <morgan@parc.power.net>

Maintainer:

Linux-PAM maintainer.

Management groups provided:

account; authentication; password; session

Cryptographically sensitive:

Security rating:

VERY LOW. Use with extreme caution.

Clean code base:

Clean.

System dependencies:

Network aware:

6.11.2 Overview of module

This module is very dangerous. It should be used with extreme caution. Its action is always to permit access. It does nothing else.

6.11.3 Account+Authentication+Password+Session components

Recognized arguments:

Description:

No matter what management group, the action of this module is to simply return `PAM_SUCCESS` – operation successful.

Examples/suggested usage:

It is seldom a good idea to use this module. However, it does have some legitimate uses. For example, if the system-administrator wishes to turn off the account management on a workstation, and at the same time continue to allow logins, then she might use the following configuration file entry for login:

```
#
# add this line to your other login entries to disable account
# management, but continue to permit users to log in...
#
login   account  required       /usr/lib/security/pam_permit.so
```

6.12 The Password-Database module

6.12.1 Synopsis

Module Name:

pam_pwdb

Author:

Cristian Gafton <gafton@sorosis.ro>
and Andrew G. Morgan <morgan@parc.power.net>

Maintainer:

Authors.

Management groups provided:

account; authentication; password; session

Cryptographically sensitive:

Security rating:

Clean code base:

System dependencies:

Requires properly configured `libpwdb`

Network aware:

6.12.2 Overview of module

This module is a pluggable replacement for the `pam_unix_..` modules. It uses the generic interface of the *Password Database* library `http://parc.power.net/morgan/libpwdb/index.html`.

6.12.3 Account component

Recognized arguments:

`debug`

Description:

The `debug` argument makes the accounting functions of this module `syslog(3)` more information on its actions. (Remaining arguments supported by the other functions of this module are silently ignored, but others are logged as errors through `syslog(3)`).

Based on the following `pwdb_elements`: `expire`; `last_change`; `max_change`; `defer_change`; `warn_change`, this module performs the task of establishing the status of the user's account and password. In the case of the latter, it may offer advice to the user on changing their password or, through the `PAM_AUTHTOKEN_REQD` return, delay giving service to the user until they have established a new password. The entries listed above are documented in the *Password Database Library Guide* (see pointer above). Should the user's record not contain one or more of these entries, the corresponding *shadow* check is not performed.

Examples/suggested usage:

In its accounting mode, this module can be inserted as follows:

```
#
# Ensure users account and password are still active
#
login   account  required       /usr/lib/secrity/pam_pwdb.so
```

6.12.4 Authentication component

Recognized arguments:

`debug`; `use_first_pass`; `try_first_pass`; `nullok`

Description:

The `debug` argument makes the authentication functions of this module `syslog(3)` more information on its actions.

The default action of this module is to not permit the user access to a service if their *official* password is blank. The `nullok` argument overrides this default.

When given the argument `try_first_pass`, before prompting the user for their password, the module first tries the previous stacked `auth`-module's password in case that satisfies this module as well. The argument `use_first_pass` forces the module to use such a recalled password and will never prompt the user - if no password is available or the password is not appropriate, the user will be denied access.

Remaining arguments, supported by the other functions of this module, are silently ignored. Other arguments are logged as errors through `syslog(3)`.

Examples/suggested usage:

The correct functionality of this module is dictated by having an appropriate `/etc/pwdb.conf` file, the user databases specified there dictate the source of the authenticated user's record.

6.12.5 Password component

Recognized arguments:

`debug`; `nullok`; `not_set_pass`; `use_authtok`; `try_first_pass`; `use_first_pass`; `md5`; `shadow`; `radius`

Description:

This part of the `pam_pwdb` module performs the task of updating the user's password. Thanks to the flexibility of `libpwdb` this module is able to move the user's password from one database to another, perhaps securing the user's database entry in a dynamic manner (*this is very ALPHA code at the moment!*) - this is the purpose of the `shadow` and `radius` arguments.

In the case of conventional unix databases (which store the password encrypted) the `md5` argument is used to do the encryption with the MD5 function as opposed to the *conventional* `crypt(3)` call.

The `nullok` module is used to permit the changing of a password *from* an empty one. Without this argument, empty passwords are treated as account-locking ones.

The argument `use_first_pass` is used to lock the choice of old and new passwords to that dictated by the previously stacked `password` module. The `try_first_pass` argument is used to avoid the user having to re-enter an old password when `pam_pwdb` follows a module that possibly shared the user's old password - if this old password is not correct the user will be prompted for the correct one. The argument `use_authtok` is used to *force* this module to set the new password to the one provided by the previously stacked `password` module (this is used in an example of the stacking of the *Cracklib* module documented above).

The `not_set_pass` argument is used to inform the module that it is not to pay attention to/make available the old or new passwords from/to other (stacked) password modules.

The `debug` argument makes the password functions of this module `syslog(3)` more information on its actions. Other arguments may be logged as erroneous to `syslog(3)`.

Examples/suggested usage:

An example of the stacking of this module with respect to the pluggable password checking module, `pam_cracklib`, is given in that modules section above.

6.12.6 Session component

Recognized arguments:

Description:

No arguments are recognized by this module component. Its action is simply to log the username and the service-type to syslog(3). Messages are logged at the beginning and end of the user's session.

Examples/suggested usage:

The use of the session modules is straightforward:

```
#
# pwdb - unix like session opening and closing
#
login   session  required       /usr/lib/security/pam_pwdb.so
```

6.13 The RADIUS session module

6.13.1 Synopsis

Module Name:

pam_radius

Author:

Cristian Gafton <gafton@sorosis.ro>

Maintainer:

Author.

Management groups provided:

session

Cryptographically sensitive:

This module does not deal with passwords

Security rating:

Clean code base:

gcc reports 1 warning when compiling /usr/include/rpc/clnt.h. Hey, is not my fault !

System dependencies:

requires /etc/security/radius.conf - configuration file which holds the secret shared with the RADIUS server.

Network aware:

yes; this is a network module (independent of application).

6.13.2 Overview of module

This module is intended to provide the session service for users autheticated with a RADIUS server. At the present stage, the only option supported is the use of the RADIUS server as an accounting server.

6.13.3 Session component

Recognized arguments:

`debug` - verbose logging to `syslog(3)`.

Description:

This module is intended to provide the session service for users autheticated with a RADIUS server. At the present stage, the only option supported is the use of the RADIUS server as an *accounting* server.

(There are few things which needs to be cleared out first in the PAM project until one will be able to use this module and expect it to magically start pppd in response to a RADIUS server command to use PPP for this user, or to initiate a telnet connection to another host, or to hang and call back the user using parameters provided in the RADIUS server response. Most of these things are better suited for the radius login application. I hope to make available Real Soon (tm) patches for the login apps to make it work this way.)

When opening a session, this module sends an "Accounting-Start" message to the RADIUS server, which will log/update/whatever a database for this user. On close, an "Accounting-Stop" message is sent to the RADIUS server.

This module has no other prerequisites for making it work. One can install a RADIUS server just for fun and use it as a centralized accounting server and forget about wtmp/last/sac etc. .

Examples/suggested usage:

For the services that need this module (*login* for example) put the following line in `/etc/pam.conf` as the last line for that service (usually after the pam_unix session line):

```
login   session  required       /lib/security/pam_radius.so
```

Replace `login` for each service you are using this module.

This module expects to find a RADIUS client configuration installed. Well, it sounds complicated, but all you need to do is to update your root-read-only `/etc/security/radius.conf` file. **Remember, this file MUST be readable only by root!**

6.14 The rhosts module

6.14.1 Synopsis

Module Name:

`pam_rhosts_auth`

Author:

Al Longyear <longyear@netcom.com>

Maintainer:

Management groups provided:

authentication

Cryptographically sensitive:

Security rating:

Clean code base:

Clean.

System dependencies:

Network aware:

Standard `inet_addr()`, `gethostbyname()` function calls.

6.14.2 Overview of module

This module performs the standard network authentication for services, as used by traditional implementations of *rlogin* and *rsh* etc.

6.14.3 Authentication component

Recognized arguments:

`no_hosts_equiv`; `no_rhosts`; `debug`; `no_warn`

Description:

The authentication mechanism of this module is based on the contents of two files; `/etc/hosts.equiv` (or `_PATH_HEQUIV` in `#include <netdb.h>`) and `~/.rhosts`. Firstly, hosts listed in the former file are treated as equivalent to the localhost. Secondly, entries in the user's own copy of the latter file is used to map "`remote-host remote-user`" pairs to that user's account on the current host. Access is granted to the user if their host is present in `/etc/hosts.equiv` and their remote account is identical to their local one, or if their remote account has an entry in their personal configuration file.

Some restrictions are applied to the attributes of the user's personal configuration file: it must be a regular file (as defined by `S_ISREG(x)` of POSIX.1); it must be owned by the *superuser* or the user; it must not be writable by any user besides its owner.

The module authenticates a remote user (internally specified by the item `PAM_RUSER`) connecting from the remote host (internally specified by the item `PAM_RHOST`). Accordingly, for applications to be compatible this authentication module they must set these items prior to calling `pam_authenticate()`. The module is not capable of independently probing the network connection for such information.

Note, the name "`root`" is hard-wired into the program (XXX - this should be reconfigurable with a configuration option). Caution should be used in the case that the local superuser goes by another name. In the case of `root`-access, the `/etc/host.equiv` file is *ignored*. Instead, the superuser must have a correctly configured personal configuration file.

The behavior of the module is modified by flags:

- `debug` - log more information to `syslog(3)`. (XXX - actually, this module does not do any logging currently, please volunteer to fix this!)
- `no_warn` - do not give verbal warnings to the user about failures etc. (XXX - this module currently does not issue any warnings, please volunteer to fix this!)
- `no_hosts_equiv` - ignore the contents of the `/etc/hosts.equiv` file.
- `no_rhosts` - ignore the contents of all user's personal configuration file `~/.rhosts`.

Examples/suggested usage:

To allow users to login from trusted remote machines, you should try adding the following line to your `/etc/pam.conf` file *before* the line that would otherwise prompt the user for a password:

```
#
# No passwords required for users from hosts listed above.
#
login   auth    sufficient      /usr/lib/security/pam_rhosts_auth.so
no_rhosts
```

Note, in this example, the system administrator has turned off all *personal rhosts* configuration files. Also note, that this module can be used to *only* allow remote login from hosts specified in the `/etc/host.equiv` file, by replacing `sufficient` in the above example with `required`.

6.15 The root access module

6.15.1 Synopsis

Module Name:

pam_rootok

Author:

Andrew G. Morgan <morgan@parc.power.net>

Maintainer:

Linux-PAM maintainer

Management groups provided:

authentication

Cryptographically sensitive:

Security rating:

Clean code base:

Clean.

System dependencies:

Network aware:

6.15.2 Overview of module

This module is for use in situations where the superuser wishes to gain access to a service without having to enter a password.

6.15.3 Authentication component

Recognized arguments:

`debug`

Description:

This module authenticates the user if their `uid` is `0`. Applications that are created *setuid*-root generally retain the `uid` of the user but run with the authority of an enhanced *effective*-`uid`. It is the real `uid` that is checked.

Examples/suggested usage:

In the case of the `su` application the historical usage is to permit the superuser to adopt the identity of a lesser user without the use of a password. To obtain this behavior under `Linux-PAM` the following pair of lines are needed for the corresponding entry in the configuration file:

```
#
# su authentication. Root is granted access by default.
#
su      auth     sufficient     /usr/lib/security/pam_rootok.so
su      auth     required       /usr/lib/security/pam_unix_auth.so
```

Note. For programs that are run by the superuser (or started when the system boots) this module should not be used to authenticate users.

6.16 The securetty module

6.16.1 Synopsis

Module Name:

`pam_securetty`

Author*s*:

Elliot Lee <sopwith@cuc.edu>

Maintainer:

Red Hat Software:
currently Michael K. Johnson <johnsonm@redhat.com>
(if unavailable, contact Elliot Lee <sopwith@cuc.edu>).

Management groups provided:

authentication

Cryptographically sensitive:

Security rating:

Clean code base:

System dependencies:

`/etc/securetty` file

Network aware:

Requires the application to fill in the `PAM_TTY` item correctly in order to act meaningfully.

6.16.2 Overview of module

Provides standard Unix securetty checking.

6.16.3 Authentication component

Recognized arguments:

Description:

Provides standard Unix securetty checking, which causes authentication for root to fail unless `PAM_TTY` is set to a string listed in the `/etc/securetty` file. For all other users, it succeeds.

Examples/suggested usage:

For canonical usage, should be listed as a `required` authentication method before any `sufficient` authentication methods.

6.17 Warning logger module

6.17.1 Synopsis

Module Name:

`pam_warn`

Author:

Andrew G. Morgan <morgan@parc.power.net>

Maintainer:

Author.

Management groups provided:

authentication; password

Cryptographically sensitive:

Security rating:

Clean code base:

System dependencies:

Network aware:

logs information about the remote user and host (if pam-items are known)

6.17.2 Overview of module

This module is principally for logging information about a proposed authentication or application to update a password.

6.17.3 Authentication+Password component

Recognized arguments:

Description:

Log the service, terminal, user, remote user and remote host to `syslog(3)`. The items are not probed for, but instead obtained from the standard pam-items.

Examples/suggested usage:

an example is provided in the configuration file section 4.

6.18 The wheel module

6.18.1 Synopsis

Module Name:

`pam_wheel`

Author:

Cristian Gafton <gafton@sorosis.ro>

Maintainer:

Author.

Management groups provided:

authentication

Cryptographically sensitive:

Security rating:

Clean code base:

System dependencies:

Requires libpwdb.

Network aware:

6.18.2 Overview of module

Only permit root access to members of the wheel (`gid=0`) group.

6.18.3 Authentication component

Recognized arguments:

`debug`; `use_uid`; `trust`; `deny`; `group=XXXX`

Description:

This module is used to enforce the so-called wheel group. By default, it permits root access to the system if the applicant user is a member of the `wheel` group (better described as the group with group-id `0`).

The action of the module may be modified from this default by one or more of the following flags in the `/etc/pam.conf` file.

- `debug` - Supply more debugging information to `syslog(3)`.
- `use_id` - This option modifies the behavior of the module by using the current `uid` of the process and not the `getlogin(3)` name of the user. This option is useful for being able to jump from one account to another, for example with 'su'.
- `trust` - This option instructs the module to return `PAM_SUCCESS` should it find the user applying for root privilege is a member of the wheel group. The default action is to return `PAM_IGNORE` in this situation. By using the `trust` option it is possible to arrange for `wheel`-group members to become root without typing a password. **USE WITH CARE**.
- `deny` - This is used to reverse the logic of the module's behavior. If the user is trying to get `uid=0` access and is a member of the wheel group, deny access (for the wheel group, this is perhaps nonsense!): it is intended for use in conjunction with the `group=` argument...
- `group=XXX` - Instead of checking the `gid=0` group, use the user's `XXX` group membership for the authentication.

Examples/suggested usage:

To restrict access to superuser status to the members of the `wheel` group, use the following entries in your configuration file:

```
#
# root gains access by default (rootok), only wheel members can become
# root (wheel) but Unix authenticate non-root applicants.
#
su      auth     sufficient     /usr/lib/security/pam_rootok.so
su      auth     required       /usr/lib/security/pam_wheel.so
su      auth     required       /usr/lib/security/pam_unix_auth.so
```

7 Files

`/usr/lib/libpam.so.*`

the shared library providing applications with access to **Linux-PAM**.

`/etc/pam.conf`

the **Linux-PAM** configuration file.

`/usr/lib/security/pam_*.so`

the primary location for **Linux-PAM** dynamically loadable object files; the modules.

8 See also

- The **Linux-PAM** Application Writers' Guide.
- The **Linux-PAM** Module Writers' Guide.
- The V. Samar and R. Schemers (SunSoft), "UNIFIED LOGIN WITH PLUGGABLE AUTHENTICATION MODULES", Open Software Foundation Request For Comments 86.0, October 1995. See this url: `http://www.pilgrim.umass.edu/pub/osf_dce/RFC/rfc86.0.txt`

9 Notes

I intend to put development comments here... like "at the moment this isn't actually supported". At release time what ever is in this section will be placed in the Bugs section below! :)

Are we going to be able to support the `use_mapped_pass` module argument? Anyone know a cheap (free) good lawyer?!

- This issue may go away, as Sun have investigated adding a new management group for mappings. In this way, libpam would have mapping modules that could securely store passwords using strong cryptography and in such a way that they need not be distributed with Linux-PAM.

10 Author/acknowledgments

This document was written by Andrew G. Morgan (morgan@parc.power.net) with many contributions from Craig S. Bell, Oliver Crow, Marc Ewing, Cristian Gafton, Michael K. Johnson, Al Longyear, Marek Michalkiewicz, Theodore Ts'o, Jeff Uphoff, Joseph S. D. Yao and Alex O. Yuriev.

Thanks are also due to Sun Microsystems, especially to Vipin Samar and Charlie Lai for their advice. At an early stage in the development of **Linux-PAM**, Sun graciously made the documentation for their implementation of PAM available. This act greatly accelerated the development of **Linux-PAM**.

11 Bugs/omissions

More PAM modules are being developed all the time. It is unlikely that this document will ever be truely up to date!

Currently there is no documentation for PAM-aware applications.

This manual is unfinished. Only a partial list of people is credited for all the good work they have done.