The Linux-PAM System Administrators' Guide

Andrew G. Morgan <morgan@kernel.org>
Thorsten Kukuk <kukuk@thkukuk.de>

The Linux-PAM System Administrators' Guide

by Andrew G. Morgan and Thorsten Kukuk

Version 1.1.2, 31. August 2010

Abstract

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.

Chapter 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.

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` (or a series of configuration files located in `/etc/pam.d/`) to authenticate a user request via the locally available authentication modules. The modules themselves will usually be located in the directory `/lib/security` or `/lib64/security` and take the form of dynamically loadable object files (see dlopen(3)).

Chapter 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 bibliography"). If you are using a distribution of Linux (or some other operating system) that supports PAM but chooses to distribute these files in a different way you should be careful when copying examples directly from the text.

As an example of the above, where it is explicit, the text assumes that PAM loadable object files (the *modules*) are to be located in the following directory: `/lib/security/` or `/lib64/security` depending on the architecture. This is generally the location that seems to be compatible with the Filesystem Hierarchy Standard (FHS). On Solaris, which has its own licensed version of PAM, and some other implementations of UN*X, these files can be found in `/usr/lib/security`. Please be careful to perform the necessary transcription when using the examples from the text.

Chapter 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 (bash, tcsh, zsh, etc.) running with the identity of the user.

Traditionally, 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 as far as the system is concerned the user is who they claim to be. This is the task that is delegated to *Linux-PAM*.

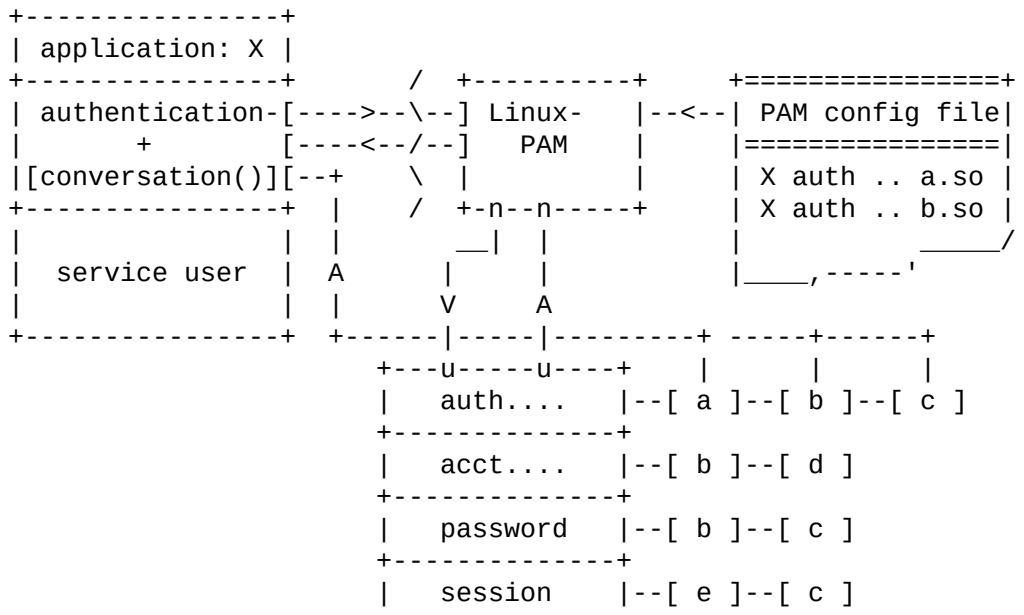
From the perspective of the application programmer (in this case the person that wrote the **login** application), *Linux-PAM* takes care of this authentication task -- verifying the identity of the user.

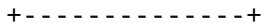
The flexibility of *Linux-PAM* is that *you*, the system administrator, have the freedom to stipulate which authentication scheme is to be used. You have the freedom to set the scheme for any/all PAM-aware applications on your Linux system. That is, you can authenticate from anything as naive as *simple 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 relevant *Linux-PAM* configuration file. The management functions are performed by *modules* specified in the configuration file. The syntax for this file is discussed in the section below.

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





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 PAM configuration 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 configuration 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.

If a program is going to use PAM, then it has to have PAM functions explicitly coded into the program. If you have access to the source code you can add the appropriate PAM functions. If you do not have access to the source code, and the binary does not have the PAM functions included, then it is not possible to use PAM.

Chapter 4. The Linux-PAM configuration file

When a *PAM* aware privilege granting application is started, it activates its attachment to the PAM-API. This activation performs a number of tasks, the most important being the reading of the configuration file(s): `/etc/pam.conf`. Alternatively, this may be the contents of the `/etc/pam.d/` directory. The presence of this directory will cause Linux-PAM to ignore `/etc/pam.conf`.

These files list the *PAM*s that will do the authentication tasks required by this service, and the appropriate behavior of the PAM-API in the event that individual *PAM*s fail.

4.1. Configuration file syntax

The syntax of the `/etc/pam.conf` configuration file is as follows. The file is made up of a list of rules, each rule is typically placed on a single line, but may be extended with an escaped end of line: `\<LF>'. Comments are preceded with `#' marks and extend to the next end of line.

The format of each rule is a space separated collection of tokens, the first three being case-insensitive:

service type control module-path module-arguments

The syntax of files contained in the `/etc/pam.d/` directory, are identical except for the absence of any *service* field. In this case, the *service* is the name of the file in the `/etc/pam.d/` directory. This filename must be in lower case.

An important feature of *PAM*, is that a number of rules may be *stacked* to combine the services of a number of PAMs for a given authentication task.

The *service* is typically the familiar name of the corresponding application: *login* and *su* are good examples. The *service*-name, *other*, is reserved for giving *default* rules. Only lines that mention the current service (or in the absence of such, the *other* entries) will be associated with the given service-application.

The *type* is the management group that the rule corresponds to. It is used to specify which of the management groups the subsequent module is to be associated with. Valid entries are:

account — this module type 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.

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 or other privileges through its credential granting properties.

password — this 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) type.

session — this module type 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.

If the *type* value from the list above is prepended with a - character the PAM library will not log to the system log if it is not possible to load the module because it is missing in the system. This can be useful especially for modules which are not always installed on the system and are not required for correct authentication and authorization of the login session.

The third field, *control*, indicates the behavior of the PAM-API should the module fail to succeed in its authentication task. There are two types of syntax for this control field: the simple one has a single simple keyword; the more complicated one involves a square-bracketed selection of *value=action* pairs.

For the simple (historical) syntax valid *control* values are:

required — failure of such a PAM will ultimately lead to the PAM-API returning failure but only after the remaining *stacked* modules (for this *service* and *type*) have been invoked.

requisite — like *required*, however, in the case that such a module returns a failure, control is directly returned to the application or to the superior PAM stack. The return value is that associated with the first required or requisite module to fail. Note, this flag can be used to protect against the possibility of a user getting the opportunity to enter a password over an unsafe medium. It is conceivable that such behavior might inform an attacker of valid accounts on a system. This possibility should be weighed against the not insignificant concerns of exposing a sensitive password in a hostile environment.

sufficient — if such a module succeeds and no prior *required* module has failed the PAM framework returns success to the application or to the superior PAM stack immediately without calling any further modules in the stack. A failure of a *sufficient* module is ignored and processing of the PAM module stack continues unaffected.

optional — the success or failure of this module is only important if it is the only module in the stack associated with this *service+type*.

include — include all lines of given type from the configuration file specified as an argument to this control.

substack — include all lines of given type from the configuration file specified as an argument to this control. This differs from *include* in that evaluation of the *done* and *die* actions in a substack does not cause skipping the rest of the complete module stack, but only of the substack. Jumps in a substack also can not make evaluation jump out of it, and the whole substack is counted as one module when the jump is done in a parent stack. The *reset* action will reset the state of a module stack to the state it was in as of beginning of the substack evaluation.

For the more complicated syntax valid *control* values have the following form:

```
[value1=action1 value2=action2 ...]
```

Where *valueN* corresponds to the return code from the function invoked in the module for which the line is defined. It is selected from one of these: *success*, *open_err*, *symbol_err*, *service_err*, *system_err*, *buf_err*, *perm_denied*, *auth_err*, *cred_insufficient*, *authinfo_unavail*, *user_unknown*, *maxtries*, *new_authtok_reqd*, *acct_expired*, *session_err*, *cred_unavail*, *cred_expired*, *cred_err*, *no_module_data*, *conv_err*, *authtok_err*, *authtok_recover_err*, *authtok_lock_busy*, *authtok_disable_aging*, *try_again*, *ignore*, *abort*, *authtok_expired*, *module_unknown*, *bad_item*, *conv_again*, *incomplete*, and *default*.

The last of these, *default*, implies 'all *valueN*'s not mentioned explicitly. Note, the full list of PAM errors is available in `/usr/include/security/_pam_types.h`. The *actionN* can take one of the following forms:

ignore	when used with a stack of modules, the module's return status will not contribute to the return code the application obtains.
bad	this action indicates that the return code should be thought of as indicative of the module failing. If this module is the first in the stack to fail, its status value will be used for that of the whole stack.
die	equivalent to bad with the side effect of terminating the module stack and PAM immediately returning to the application.
ok	this tells PAM that the administrator thinks this return code should contribute directly to the return code of the full stack of modules. In other words, if the former state of the stack would lead to a return of *PAM_SUCCESS*, the module's return code will override this value. Note, if the former state of the stack holds some value that is indicative of a modules failure, this 'ok' value will not be used to override that value.
done	equivalent to ok with the side effect of terminating the module stack and PAM immediately returning to the application.
N (an unsigned integer)	equivalent to ok with the side effect of jumping over the next N modules in the stack. Note that N equal to 0 is not allowed (and it would be identical to ok in such case).
reset	clear all memory of the state of the module stack and start again with the next stacked module.

Each of the four keywords: required; requisite; sufficient; and optional, have an equivalent expression in terms of the [...] syntax. They are as follows:

required	[success=ok new_authtok_reqd=ok ignore=ignore default=bad]
requisite	[success=ok new_authtok_reqd=ok ignore=ignore default=die]
sufficient	[success=done new_authtok_reqd=done default=ignore]
optional	[success=ok new_authtok_reqd=ok default=ignore]

module-path is either the full filename of the PAM to be used by the application (it begins with a '/'), or a relative pathname from the default module location: `/lib/security/` or `/lib64/security/`, depending on the architecture.

module-arguments are a space separated list of tokens that can be used to modify the specific behavior of the given PAM. Such arguments will be documented for each individual module. Note, if you wish to include spaces in an argument, you should surround that argument with square brackets.

```
squid auth required pam_mysql.so user=passwd_query passwd=mada \
      db=eminence [query=select user_name from internet_service \
      where user_name='%u' and password=PASSWORD('%p') and \
    service='web_proxy']
```

When using this convention, you can include `[' characters inside the string, and if you wish to include a `]' character inside the string that will survive the argument parsing, you should use `\]'. In other words:

```
[..[..\]..]    -->   ..[..]..
```

Any line in (one of) the configuration file(s), that is not formatted correctly, will generally tend (erring on the side of caution) to make the authentication process fail. A corresponding error is written to the system log files with a call to syslog(3).

4.2. Directory based configuration

More flexible than the single configuration file is it to configure libpam via the contents of the `/etc/pam.d/` directory. In this case the directory is filled with files each of which has a filename equal to a service-name (in lower-case): it is the personal configuration file for the named service.

The syntax of each file in /etc/pam.d/ is similar to that of the `/etc/pam.conf` file and is made up of lines of the following form:

```
type  control  module-path  module-arguments
```

The only difference being that the service-name is not present. The service-name is of course the name of the given configuration file. For example, `/etc/pam.d/login` contains the configuration for the *login* service.

4.3. Example configuration file entries

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.

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       pam_deny.so
other   account  required       pam_deny.so
other   password required       pam_deny.so
other   session  required       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       pam_warn.so
other   password required       pam_warn.so
```

Having two '**other auth**' lines is an example of stacking.

On a system that uses the `/etc/pam.d/` configuration, the corresponding default setup would be achieved with the following file:

```
#
# default configuration: /etc/pam.d/other
#
auth     required       pam_warn.so
auth     required       pam_deny.so
account  required       pam_deny.so
password required       pam_warn.so
password required       pam_deny.so
session  required       pam_deny.so
```

This is the only explicit example we give for an `/etc/pam.d/` file. In general, it should be clear how to transpose the remaining examples to this configuration scheme.

On a less sensitive computer, one on which the system administrator wishes to remain ignorant of much of the power of *Linux-PAM*, the following selection of lines (in `/etc/pam.d/other`) is likely to mimic the historically familiar Linux setup.

```
#
# default; standard UN*X access
#
auth     required       pam_unix.so
account  required       pam_unix.so
password required       pam_unix.so
session  required       pam_unix.so
```

In general this will provide a starting place for most applications.

Chapter 5. Security issues

5.1. If something goes wrong

Linux-PAM has the potential to seriously change the security of your system. You can choose to have no security or absolute security (no access permitted). In general, *Linux-PAM* errs towards the latter. Any number of configuration errors can disable access to your system partially, or completely.

The most dramatic problem that is likely to be encountered when configuring *Linux-PAM* is that of *deleting* the configuration file(s): `/etc/pam.d/*` and/or `/etc/pam.conf`. This will lock you out of your own system!

To recover, your best bet is to restore the system from a backup or boot the system into a rescue system and correct things from there.

5.2. Avoid having a weak `other' configuration

It is not a good thing to have a weak default (*other*) entry. This service is the default configuration for all PAM aware applications and if it is weak, your system is likely to be vulnerable to attack.

Here is a sample "other" configuration file. The **pam_deny** module will deny access and the **pam_warn** module will send a syslog message to *auth.notice*:

```
#
# The PAM configuration file for the `other' service
#
auth      required   pam_deny.so
auth      required   pam_warn.so
account   required   pam_deny.so
account   required   pam_warn.so
password  required   pam_deny.so
password  required   pam_warn.so
session   required   pam_deny.so
session   required   pam_warn.so
```

Chapter 6. A reference guide for available modules

Here, we collect together the descriptions of the various modules coming with Linux-PAM.

6.1. pam_access - logdaemon style login access control

`pam_access.so` [debug] [nodefgroup] [noaudit] [accessfile=*`file`*] [fieldsep=*`sep`*] [listsep=*`sep`*]

6.1.1. DESCRIPTION

The pam_access PAM module is mainly for access management. It provides logdaemon style login access control based on login names, host or domain names, internet addresses or network numbers, or on terminal line names, X `$DISPLAY` values, or PAM service names in case of non-networked logins.

By default rules for access management are taken from config file `/etc/security/access.conf` if you don't specify another file. Then individual `*.conf` files from the `/etc/security/access.d/` directory are read. The files are parsed one after another in the order of the system locale. The effect of the individual files is the same as if all the files were concatenated together in the order of parsing. This means that once a pattern is matched in some file no further files are parsed. If a config file is explicitly specified with the `accessfile` option the files in the above directory are not parsed.

If Linux PAM is compiled with audit support the module will report when it denies access based on origin (host, tty, etc.).

6.1.2. DESCRIPTION

The `/etc/security/access.conf` file specifies (*`user/group`*, *`host`*), (*`user/group`*, *`network/netmask`*), (*`user/group`*, *`tty`*), (*`user/group`*, *`X-$DISPLAY-value`*), or (*`user/group`*, *`pam-service-name`*) combinations for which a login will be either accepted or refused.

When someone logs in, the file `access.conf` is scanned for the first entry that matches the (*`user/group`*, *`host`*) or (*`user/group`*, *`network/netmask`*) combination, or, in case of non-networked logins, the first entry that matches the (*`user/group`*, *`tty`*) combination, or in the case of non-networked logins without a tty, the first entry that matches the (*`user/group`*, *`X-$DISPLAY-value`*) or (*`user/group`*, *`pam-service-name/`*) combination. The permissions field of that table entry determines whether the login will be accepted or refused.

Each line of the login access control table has three fields separated by a ":" character (colon):

`permission`:*`users/groups`*:*`origins`*

The first field, the *`permission`* field, can be either a "+" character (plus) for access granted or a "-" character (minus) for access denied.

The second field, the *`users`*/*`group`* field, should be a list of one or more login names, group names, or *ALL* (which always matches). To differentiate user entries from group entries, group entries should be written with brackets, e.g. *(group)*.

The third field, the *origins* field, should be a list of one or more tty names (for non-networked logins), X $DISPLAY values or PAM service names (for non-networked logins without a tty), host names, domain names (begin with "."), host addresses, internet network numbers (end with "."), internet network addresses with network mask (where network mask can be a decimal number or an internet address also), *ALL* (which always matches) or *LOCAL*. The *LOCAL* keyword matches if and only if pam_get_item(3), when called with an *item_type* of *PAM_RHOST*, returns NULL or an empty string (and therefore the *origins* field is compared against the return value of pam_get_item(3) called with an *item_type* of *PAM_TTY* or, absent that, *PAM_SERVICE*).

If supported by the system you can use *@netgroupname* in host or user patterns. The *@@netgroupname* syntax is supported in the user pattern only and it makes the local system hostname to be passed to the netgroup match call in addition to the user name. This might not work correctly on some libc implementations causing the match to always fail.

The *EXCEPT* operator makes it possible to write very compact rules.

If the nodefgroup is not set, the group file is searched when a name does not match that of the logged-in user. Only groups are matched in which users are explicitly listed. However the PAM module does not look at the primary group id of a user.

The "#" character at start of line (no space at front) can be used to mark this line as a comment line.

6.1.3. OPTIONS

accessfile=*/path/to/access.conf*	Indicate an alternative access.conf style configuration file to override the default. This can be useful when different services need different access lists.
debug	A lot of debug information is printed with syslog(3).
noaudit	Do not report logins from disallowed hosts and ttys to the audit subsystem.
fieldsep=*separators*	This option modifies the field separator character that pam_access will recognize when parsing the access configuration file. For example: *fieldsep=|* will cause the default `:' character to be treated as part of a field value and `|' becomes the field separator. Doing this may be useful in conjunction with a system that wants to use pam_access with X based applications, since the *PAM_TTY* item is likely to be of the form "hostname:0" which includes a `:' character in its value. But you should not need this.
listsep=*separators*	This option modifies the list separator character that pam_access will recognize when parsing the access configuration file. For example: *listsep=,* will cause the default ` ' (space) and `\t' (tab) characters to be treated as part of a list element value and `,' becomes the only list element separator. Doing this may be useful on a system with group information obtained from a Windows domain, where the default built-in groups "Domain Users", "Domain Admins" contain a space.
nodefgroup	User tokens which are not enclosed in parentheses will not be matched against the group database. The backwards compatible default is to try the group database match even for tokens not enclosed in parentheses.

6.1.4. MODULE TYPES PROVIDED

All module types (`auth`, `account`, `password` and `session`) are provided.

6.1.5. RETURN VALUES

PAM_SUCCESS	Access was granted.
PAM_PERM_DENIED	Access was not granted.
PAM_IGNORE	`pam_setcred` was called which does nothing.
PAM_ABORT	Not all relevant data or options could be gotten.
PAM_USER_UNKNOWN	The user is not known to the system.

6.1.6. FILES

`/etc/security/access.conf`	Default configuration file

6.1.7. EXAMPLES

These are some example lines which might be specified in `/etc/security/access.conf`.

User *root* should be allowed to get access via *cron*, X11 terminal *:0*, *tty1*, ..., *tty5*, *tty6*.

+:root:crond :0 tty1 tty2 tty3 tty4 tty5 tty6

User *root* should be allowed to get access from hosts which own the IPv4 addresses. This does not mean that the connection have to be a IPv4 one, a IPv6 connection from a host with one of this IPv4 addresses does work, too.

+:root:192.168.200.1 192.168.200.4 192.168.200.9

+:root:127.0.0.1

User *root* should get access from network `192.168.201.` where the term will be evaluated by string matching. But it might be better to use network/netmask instead. The same meaning of `192.168.201.` is *192.168.201.0/24* or *192.168.201.0/255.255.255.0*.

+:root:192.168.201.

User *root* should be able to have access from hosts *foo1.bar.org* and *foo2.bar.org* (uses string matching also).

+:root:foo1.bar.org foo2.bar.org

User *root* should be able to have access from domain *foo.bar.org* (uses string matching also).

+:root:.foo.bar.org

User *root* should be denied to get access from all other sources.

-:root:ALL

User *foo* and members of netgroup *admins* should be allowed to get access from all sources. This will only work if netgroup service is available.

+:@admins foo:ALL

User *john* and *foo* should get access from IPv6 host address.

+:john foo:2001:db8:0:101::1

User *john* should get access from IPv6 net/mask.

+:john:2001:db8:0:101::/64

Disallow console logins to all but the shutdown, sync and all other accounts, which are a member of the wheel group.

-:ALL EXCEPT (wheel) shutdown sync:LOCAL

All other users should be denied to get access from all sources.

-:ALL:ALL

6.1.8. AUTHORS

The logdaemon style login access control scheme was designed and implemented by Wietse Venema. The pam_access PAM module was developed by Alexei Nogin <alexei@nogin.dnttm.ru>. The IPv6 support and the network(address) / netmask feature was developed and provided by Mike Becher <mike.becher@lrz-muenchen.de>.

6.2. pam_cracklib - checks the password against dictionary words

`pam_cracklib.so[...]`

6.2.1. DESCRIPTION

This module can be plugged into the *password* stack of a given application to provide some plug-in strength-checking for passwords.

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 first 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 strength checks works in the following manner: at first the `Cracklib` routine is called to check if the password is part of a dictionary; if this is not the case an additional set of strength checks is done. These checks are:

Palindrome	Is the new password a palindrome?
Case Change Only	Is the new password the old one with only a change of case?

Similar	Is the new password too much like the old one? This is primarily controlled by one argument, `difok` which is a number of character changes (inserts, removals, or replacements) between the old and new password that are enough to accept the new password. This defaults to 5 changes.
Simple	Is the new password too small? This is controlled by 6 arguments `minlen`, `maxclassrepeat`, `dcredit`, `ucredit`, `lcredit`, and `ocredit`. See the section on the arguments for the details of how these work and there defaults.
Rotated	Is the new password a rotated version of the old password?
Same consecutive characters	Optional check for same consecutive characters.
Too long monotonic character sequence	Optional check for too long monotonic character sequence.
Contains user name	Optional check whether the password contains the user's name in some form.

This module with no arguments will work well for standard unix password encryption. With md5 encryption, passwords can be longer than 8 characters and the default settings for this module can make it hard for the user to choose a satisfactory new password. Notably, the requirement that the new password contain no more than 1/2 of the characters in the old password becomes a non-trivial constraint. For example, an old password of the form "the quick brown fox jumped over the lazy dogs" would be difficult to change... In addition, the default action is to allow passwords as small as 5 characters in length. For a md5 systems it can be a good idea to increase the required minimum size of a password. One can then allow more credit for different kinds of characters but accept that the new password may share most of these characters with the old password.

6.2.2. OPTIONS

`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).
`authtok_type=`*`XXX`*	The default action is for the module to use the following prompts when requesting passwords: "New UNIX password: " and "Retype UNIX password: ". The example word *UNIX* can be replaced with this option, by default it is empty.
`retry=`*`N`*	Prompt user at most *N* times before returning with error. The default is *1*.
`difok=`*`N`*	This argument will change the default of *5* for the number of character changes in the new password that differentiate it from the old password.
`minlen=`*`N`*	The minimum acceptable size for the new password (plus one if credits are not disabled which is the default). In addition to the number of characters in the new password, credit (of +1 in length) is given for each different kind of character (*other*, *upper*, *lower* and *digit*). The default for this parameter is *9* which is good for a old style UNIX password all of the same type of character but

may be too low to exploit the added security of a md5 system. Note that there is a pair of length limits in *Cracklib* itself, a "way too short" limit of 4 which is hard coded in and a defined limit (6) that will be checked without reference to `minlen`. If you want to allow passwords as short as 5 characters you should not use this module.

`dcredit=`*`N`*

(N >= 0) This is the maximum credit for having digits in the new password. If you have less than or *`N`* digits, each digit will count +1 towards meeting the current `minlen` value. The default for `dcredit` is 1 which is the recommended value for `minlen` less than 10.

(N < 0) This is the minimum number of digits that must be met for a new password.

`ucredit=`*`N`*

(N >= 0) This is the maximum credit for having upper case letters in the new password. If you have less than or *`N`* upper case letters each letter will count +1 towards meeting the current `minlen` value. The default for `ucredit` is *1* which is the recommended value for `minlen` less than 10.

(N < 0) This is the minimum number of upper case letters that must be met for a new password.

`lcredit=`*`N`*

(N >= 0) This is the maximum credit for having lower case letters in the new password. If you have less than or *`N`* lower case letters, each letter will count +1 towards meeting the current `minlen` value. The default for `lcredit` is 1 which is the recommended value for `minlen` less than 10.

(N < 0) This is the minimum number of lower case letters that must be met for a new password.

`ocredit=`*`N`*

(N >= 0) This is the maximum credit for having other characters in the new password. If you have less than or *`N`* other characters, each character will count +1 towards meeting the current `minlen` value. The default for `ocredit` is 1 which is the recommended value for `minlen` less than 10.

(N < 0) This is the minimum number of other characters that must be met for a new password.

`minclass=`*`N`*

The minimum number of required classes of characters for the new password. The default number is zero. The four classes are digits, upper and lower letters and other characters. The difference to the `credit` check is that a specific class if of characters is not required. Instead *`N`* out of four of the classes are required.

`maxrepeat=`*`N`*

Reject passwords which contain more than N same consecutive characters. The default is 0 which means that this check is disabled.

`maxsequence=`*`N`*

Reject passwords which contain monotonic character sequences longer than N. The default is 0 which means that this check is disabled. Examples of such sequence are '12345' or 'fedcb'. Note that most such passwords will not pass the simplicity check unless the sequence is only a minor part of the password.

`maxclassrepeat=`*`N`*	Reject passwords which contain more than N consecutive characters of the same class. The default is 0 which means that this check is disabled.
`reject_username`	Check whether the name of the user in straight or reversed form is contained in the new password. If it is found the new password is rejected.
`gecoscheck`	Check whether the words from the GECOS field (usually full name of the user) longer than 3 characters in straight or reversed form are contained in the new password. If any such word is found the new password is rejected.
`enforce_for_root`	The module will return error on failed check also if the user changing the password is root. This option is off by default which means that just the message about the failed check is printed but root can change the password anyway. Note that root is not asked for an old password so the checks that compare the old and new password are not performed.
`use_authtok`	This argument is used to *force* the module to not prompt the user for a new password but use the one provided by the previously stacked *password* module.
`dictpath=`*`/path/to/dict`*	Path to the cracklib dictionaries.

6.2.3. MODULE TYPES PROVIDED

Only the `password` module type is provided.

6.2.4. RETURN VALUES

PAM_SUCCESS	The new password passes all checks.
PAM_AUTHTOK_ERR	No new password was entered, the username could not be determined or the new password fails the strength checks.
PAM_AUTHTOK_RECOVERY_ERR	The old password was not supplied by a previous stacked module or got not requested from the user. The first error can happen if `use_authtok` is specified.
PAM_SERVICE_ERR	A internal error occurred.

6.2.5. EXAMPLES

For an example of the use of this module, we show how it may be stacked with the password component of pam_unix(8)

```
#
# 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_unix module does not
# prompt for a password, but instead uses the one provided by
# pam_cracklib.
```

```
#
passwd  password required       pam_cracklib.so retry=3
passwd  password required       pam_unix.so use_authtok
```

Another example (in the `/etc/pam.d/passwd` format) is for the case that you want to use md5 password encryption:

```
#%PAM-1.0
#
# These lines allow a md5 systems to support passwords of at least 14
# bytes with extra credit of 2 for digits and 2 for others the new
# password must have at least three bytes that are not present in the
# old password
#
password  required pam_cracklib.so \
              difok=3 minlen=15 dcredit= 2 ocredit=2
password  required pam_unix.so use_authtok nullok md5
```

And here is another example in case you don't want to use credits:

```
#%PAM-1.0
#
# These lines require the user to select a password with a minimum
# length of 8 and with at least 1 digit number, 1 upper case letter,
# and 1 other character
#
password  required pam_cracklib.so \
               dcredit=-1 ucredit=-1 ocredit=-1 lcredit=0 minlen=8
password  required pam_unix.so use_authtok nullok md5
```

6.2.6. AUTHOR

pam_cracklib was written by Cristian Gafton <gafton@redhat.com>

6.3. pam_debug - debug the PAM stack

`pam_debug.so` [auth=*`value`*] [cred=*`value`*] [acct=*`value`*] [prechauthtok=*`value`*] [chauthtok=*`value`*] [auth=*`value`*] [open_session=*`value`*] [close_session=*`value`*]

6.3.1. DESCRIPTION

The pam_debug PAM module is intended as a debugging aide for determining how the PAM stack is operating. This module returns what its module arguments tell it to return.

6.3.2. OPTIONS

`auth=`*`value`*	The pam_sm_authenticate(3) function will return *`value`*.
`cred=`*`value`*	The pam_sm_setcred(3) function will return *`value`*.

`acct=value` The pam_sm_acct_mgmt(3) function will return *value*.

`prechauthtok=value` The pam_sm_chauthtok(3) function will return *value* if the *PAM_PRELIM_CHECK* flag is set.

`chauthtok=value` The pam_sm_chauthtok(3) function will return *value* if the *PAM_PRELIM_CHECK* flag is *not* set.

`open_session=value` The pam_sm_open_session(3) function will return *value*.

`close_session=value` The pam_sm_close_session(3) function will return *value*.

Where *value* can be one of: success, open_err, symbol_err, service_err, system_err, buf_err, perm_denied, auth_err, cred_insufficient, authinfo_unavail, user_unknown, maxtries, new_authtok_reqd, acct_expired, session_err, cred_unavail, cred_expired, cred_err, no_module_data, conv_err, authtok_err, authtok_recover_err, authtok_lock_busy, authtok_disable_aging, try_again, ignore, abort, authtok_expired, module_unknown, bad_item, conv_again, incomplete.

6.3.3. MODULE TYPES PROVIDED

All module types (`auth`, `account`, `password` and `session`) are provided.

6.3.4. RETURN VALUES

PAM_SUCCESS Default return code if no other value was specified, else specified return value.

6.3.5. EXAMPLES

```
auth    requisite       pam_permit.so
auth    [success=2 default=ok]  pam_debug.so auth=perm_denied cred=success
auth    [default=reset]         pam_debug.so auth=success cred=perm_denied
auth    [success=done default=die] pam_debug.so
auth    optional        pam_debug.so auth=perm_denied cred=perm_denied
auth    sufficient      pam_debug.so auth=success cred=success
```

6.3.6. AUTHOR

pam_debug was written by Andrew G. Morgan <morgan@kernel.org>.

6.4. pam_deny - locking-out PAM module

`pam_deny.so`

6.4.1. DESCRIPTION

This module can be used to deny access. It always indicates a failure to the application through the PAM framework. It might be suitable for using for default (the *OTHER*) entries.

6.4.2. OPTIONS

This module does not recognise any options.

6.4.3. MODULE TYPES PROVIDED

All module types (`account`, `auth`, `password` and `session`) are provided.

6.4.4. RETURN VALUES

PAM_AUTH_ERR This is returned by the account and auth services.

PAM_CRED_ERR This is returned by the setcred function.

PAM_AUTHTOK_ERR This is returned by the password service.

PAM_SESSION_ERR This is returned by the session service.

6.4.5. EXAMPLES

```
#%PAM-1.0
#
# If we don't have config entries for a service, the
# OTHER entries are used. To be secure, warn and deny
# access to everything.
other auth     required       pam_warn.so
other auth     required       pam_deny.so
other account  required       pam_warn.so
other account  required       pam_deny.so
other password required       pam_warn.so
other password required       pam_deny.so
other session  required       pam_warn.so
other session  required       pam_deny.so
```

6.4.6. AUTHOR

pam_deny was written by Andrew G. Morgan <morgan@kernel.org>

6.5. pam_echo - print text messages

`pam_echo.so` [file=*`/path/message`*]

6.5.1. DESCRIPTION

The *pam_echo* PAM module is for printing text messages to inform user about special things. Sequences starting with the *%* character are interpreted in the following way:

%H The name of the remote host (PAM_RHOST).

%h The name of the local host.

%s The service name (PAM_SERVICE).

%t The name of the controlling terminal (PAM_TTY).

%U The remote user name (PAM_RUSER).

%u The local user name (PAM_USER).

All other sequences beginning with % expands to the characters following the % character.

6.5.2. OPTIONS

`file=`*`/path/message`* The content of the file `/path/message` will be printed with the PAM conversion function as PAM_TEXT_INFO.

6.5.3. MODULE TYPES PROVIDED

All module types (`auth`, `account`, `password` and `session`) are provided.

6.5.4. RETURN VALUES

PAM_BUF_ERR Memory buffer error.

PAM_SUCCESS Message was successful printed.

PAM_IGNORE PAM_SILENT flag was given or message file does not exist, no message printed.

6.5.5. EXAMPLES

For an example of the use of this module, we show how it may be used to print information about good passwords:

```
password optional pam_echo.so file=/usr/share/doc/good-password.txt
password required pam_unix.so
```

6.5.6. AUTHOR

Thorsten Kukuk <kukuk@thkukuk.de>

6.6. pam_env - set/unset environment variables

`pam_env.so` [debug] [conffile=*`conf-file`*] [envfile=*`env-file`*] [readenv=*`0|1`*] [user_envfile=*`env-file`*] [user_readenv=*`0|1`*]

6.6.1. DESCRIPTION

The pam_env PAM module allows the (un)setting of environment variables. Supported is the use of previously set environment variables as well as *PAM_ITEM*s such as *PAM_RHOST*.

By default rules for (un)setting of variables are taken from the config file `/etc/security/pam_env.conf`. An alternate file can be specified with the *conffile* option.

Second a file (`/etc/environment` by default) with simple *KEY=VAL* pairs on separate lines will be read. With the *envfile* option an alternate file can be specified. And with the *readenv* option this can be completely disabled.

Third it will read a user configuration file (`$HOME/.pam_environment` by default). The default file file can be changed with the *user_envfile* option and it can be turned on and off with the *user_readenv* option.

Since setting of PAM environment variables can have side effects to other modules, this module should be the last one on the stack.

6.6.2. DESCRIPTION

The `/etc/security/pam_env.conf` file specifies the environment variables to be set, unset or modified by pam_env(8). When someone logs in, this file is read and the environment variables are set according.

Each line starts with the variable name, there are then two possible options for each variable DEFAULT and OVERRIDE. DEFAULT allows and administrator to set the value of the variable to some default value, if none is supplied then the empty string is assumed. The OVERRIDE option tells pam_env that it should enter in its value (overriding the default value) if there is one to use. OVERRIDE is not used, "" is assumed and no override will be done.

VARIABLE [*DEFAULT=[value]*] [*OVERRIDE=[value]*]

(Possibly non-existent) environment variables may be used in values using the ${string} syntax and (possibly non-existent) PAM_ITEMs as well as HOME and SHELL may be used in values using the @{string} syntax. Both the $ and @ characters can be backslash escaped to be used as literal values values can be delimited with "", escaped " not supported. Note that many environment variables that you would like to use may not be set by the time the module is called. For example, ${HOME} is used below several times, but many PAM applications don't make it available by the time you need it. The special variables @{HOME} and @{SHELL} are expanded to the values for the user from his *passwd* entry.

The "#" character at start of line (no space at front) can be used to mark this line as a comment line.

The `/etc/environment` file specifies the environment variables to be set. The file must consist of simple *NAME=VALUE* pairs on separate lines. The pam_env(8) module will read the file after the `pam_env.conf` file.

6.6.3. OPTIONS

`conffile=`*`/path/to/pam_env.conf`*	Indicate an alternative `pam_env.conf` style configuration file to override the default. This can be useful when different services need different environments.
`debug`	A lot of debug information is printed with syslog(3).
`envfile=`*`/path/to/environment`*	Indicate an alternative `environment` file to override the default. The syntax are simple *KEY=VAL* pairs on separate lines. The *export* instruction can be specified for bash compatibility, but will be ignored. This can be useful when different services need different environments.
`readenv=`*`0|1`*	Turns on or off the reading of the file specified by envfile (0 is off, 1 is on). By default this option is on.
`user_envfile=`*`filename`*	Indicate an alternative `.pam_environment` file to override the default.The syntax is the same as for */etc/environment*. The filename is relative to the user home directory. This can be useful when different services need different environments.

`user_readenv=`*`0|1`* Turns on or off the reading of the user specific environment file. 0 is off, 1 is on. By default this option is on.

6.6.4. MODULE TYPES PROVIDED

The `auth` and `session` module types are provided.

6.6.5. RETURN VALUES

PAM_ABORT Not all relevant data or options could be gotten.

PAM_BUF_ERR Memory buffer error.

PAM_IGNORE No pam_env.conf and environment file was found.

PAM_SUCCESS Environment variables were set.

6.6.6. FILES

`/etc/security/pam_env.conf` Default configuration file

`/etc/environment` Default environment file

`$HOME/.pam_environment` User specific environment file

6.6.7. EXAMPLES

These are some example lines which might be specified in `/etc/security/pam_env.conf`.

Set the REMOTEHOST variable for any hosts that are remote, default to "localhost" rather than not being set at all

```
REMOTEHOST     DEFAULT=localhost OVERRIDE=@{PAM_RHOST}
```

Set the DISPLAY variable if it seems reasonable

```
DISPLAY        DEFAULT=${REMOTEHOST}:0.0 OVERRIDE=${DISPLAY}
```

Now some simple variables

```
PAGER          DEFAULT=less
MANPAGER       DEFAULT=less
LESS           DEFAULT="M q e h15 z23 b80"
NNTPSERVER     DEFAULT=localhost
PATH           DEFAULT=${HOME}/bin:/usr/local/bin:/bin\
:/usr/bin:/usr/local/bin/X11:/usr/bin/X11
XDG_DATA_HOME  @{HOME}/share/
```

Silly examples of escaped variables, just to show how they work.

```
DOLLAR          DEFAULT=\$
DOLLARDOLLAR    DEFAULT=        OVERRIDE=\$${DOLLAR}
DOLLARPLUS      DEFAULT=\${REMOTEHOST}${REMOTEHOST}
ATSIGN          DEFAULT=""      OVERRIDE=\@
```

6.6.8. AUTHOR

pam_env was written by Dave Kinchlea <kinch@kinch.ark.com>.

6.7. pam_exec - call an external command

`pam_exec.so` [debug] [expose_authtok] [seteuid] [quiet] [stdout] [log=*`file`*] [type=*`type`*] *`command`* [`...`]

6.7.1. DESCRIPTION

pam_exec is a PAM module that can be used to run an external command.

The child's environment is set to the current PAM environment list, as returned by pam_getenvlist(3) In addition, the following PAM items are exported as environment variables: *PAM_RHOST*, *PAM_RUSER*, *PAM_SERVICE*, *PAM_TTY*, *PAM_USER* and *PAM_TYPE*, which contains one of the module types: `account`, `auth`, `password`, `open_session` and `close_session`.

Commands called by pam_exec need to be aware of that the user can have control over the environment.

6.7.2. OPTIONS

`debug`	Print debug information.
`expose_authtok`	During authentication the calling command can read the password from stdin(3). Only first *PAM_MAX_RESP_SIZE* bytes of a password are provided to the command.
`log=`*`file`*	The output of the command is appended to `file`
`type=`*`type`*	Only run the command if the module type matches the given type.
`stdout`	Per default the output of the executed command is written to `/dev/null`. With this option, the stdout output of the executed command is redirected to the calling application. It's in the responsibility of this application what happens with the output. The `log` option is ignored.
`quiet`	Per default pam_exec.so will echo the exit status of the external command if it fails. Specifying this option will suppress the message.
`seteuid`	Per default pam_exec.so will execute the external command with the real user ID of the calling process. Specifying this option means the command is run with the effective user ID.

6.7.3. MODULE TYPES PROVIDED

All module types (`auth`, `account`, `password` and `session`) are provided.

6.7.4. RETURN VALUES

PAM_SUCCESS	The external command was run successfully.
PAM_SERVICE_ERR	No argument or a wrong number of arguments were given.
PAM_SYSTEM_ERR	A system error occurred or the command to execute failed.
PAM_IGNORE	`pam_setcred` was called, which does not execute the command. Or, the value given for the type= parameter did not match the module type.

6.7.5. EXAMPLES

Add the following line to `/etc/pam.d/passwd` to rebuild the NIS database after each local password change:

```
password optional pam_exec.so seteuid /usr/bin/make -C /var/yp
```

This will execute the command

`make -C /var/yp`

with effective user ID.

6.7.6. AUTHOR

pam_exec was written by Thorsten Kukuk <kukuk@thkukuk.de> and Josh Triplett <josh@joshtriplett.org>.

6.8. pam_faildelay - change the delay on failure per-application

`pam_faildelay.so` [debug] [delay=*`microseconds`*]

6.8.1. DESCRIPTION

pam_faildelay is a PAM module that can be used to set the delay on failure per-application.

If no `delay` is given, pam_faildelay will use the value of FAIL_DELAY from `/etc/login.defs`.

6.8.2. OPTIONS

`debug`	Turns on debugging messages sent to syslog.
`delay=`*`N`*	Set the delay on failure to N microseconds.

6.8.3. MODULE TYPES PROVIDED

Only the `auth` module type is provided.

6.8.4. RETURN VALUES

PAM_IGNORE Delay was successful adjusted.

PAM_SYSTEM_ERR The specified delay was not valid.

6.8.5. EXAMPLES

The following example will set the delay on failure to 10 seconds:

```
auth  optional  pam_faildelay.so  delay=10000000
```

6.8.6. AUTHOR

pam_faildelay was written by Darren Tucker <dtucker@zip.com.au>.

6.9. pam_filter - filter module

`pam_filter.so` [debug] [new_term] [non_term] run1|run2 *`filter`* [`...`]

6.9.1. DESCRIPTION

This module is intended to be a platform for providing access to all of the input/output that passes between the user and the application. It is only suitable for tty-based and (stdin/stdout) applications.

To function this module requires *filters* to be installed on the system. 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).

Each component of the module has the potential to invoke the desired filter. The filter is always execv(2) 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.

6.9.2. OPTIONS

`debug`	Print debug information.
`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.

Permitted values for *X* are *1* and *2*. These indicate the precise time that the filter is to be run. To understand this concept it will be useful to have read the pam(3) manual page. 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 pam_authenticate(3) and pam_setcred(3), here `run1` means run the filter from the `pam_authenticate` function and `run2` means run the filter from `pam_setcred`. In the case of the session modules, *run1* implies that the filter is invoked at the pam_open_session(3) stage, and *run2* for pam_close_session(3).

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 of pam_chauthtok(3) (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).

`filter` The full pathname of the filter to be run and any command line arguments that the filter might expect.

6.9.3. MODULE TYPES PROVIDED

All module types (`auth`, `account`, `password` and `session`) are provided.

6.9.4. RETURN VALUES

PAM_SUCCESS The new filter was set successfully.

PAM_ABORT Critical error, immediate abort.

6.9.5. EXAMPLES

Add the following line to `/etc/pam.d/login` to see how to configure login to transpose upper and lower case letters once the user has logged in:

```
session required pam_filter.so run1 /lib/security/pam_filter/upperLOWER
```

6.9.6. AUTHOR

pam_filter was written by Andrew G. Morgan <morgan@kernel.org>.

6.10. pam_ftp - module for anonymous access

`pam_ftp.so` [debug] [ignore] [users=*XXX*, *YYY*, ...]

6.10.1. DESCRIPTION

pam_ftp is a PAM module which provides a pluggable anonymous ftp mode of access.

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 (*PAM_USER*) is set to *ftp*. In this case the module succeeds. Alternatively, the module sets the *PAM_AUTHTOK* item with the entered password and fails.

This module is not safe and easily spoofable.

6.10.2. OPTIONS

`debug`	Print debug information.
`ignore`	Pay no attention to the email address of the user (if supplied).
`ftp=`*`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*.

6.10.3. MODULE TYPES PROVIDED

Only the `auth` module type is provided.

6.10.4. RETURN VALUES

PAM_SUCCESS	The authentication was successful.
PAM_USER_UNKNOWN	User not known.

6.10.5. EXAMPLES

Add the following line to `/etc/pam.d/ftpd` to handle ftp style anonymous login:

```
#
# ftpd; add ftp-specifics. These lines enable anonymous ftp over
#       standard UN*X access (the listfile entry blocks access to
#       users listed in /etc/ftpusers)
#
auth    sufficient  pam_ftp.so
auth    required    pam_unix.so use_first_pass
auth    required    pam_listfile.so \
          onerr=succeed item=user sense=deny file=/etc/ftpusers
```

6.10.6. AUTHOR

pam_ftp was written by Andrew G. Morgan <morgan@kernel.org>.

6.11. pam_group - module to modify group access

`pam_group.so`

6.11.1. DESCRIPTION

The pam_group PAM 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.

By default rules for group memberships are taken from config file `/etc/security/group.conf`.

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*.

The pam_group module functions in parallel with the `/etc/group` file. If the user is granted any groups based on the behavior of this module, they are granted *in addition* to those entries `/etc/group` (or equivalent).

6.11.2. DESCRIPTION

The pam_group PAM 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.

For this module to function correctly there must be a correctly formatted `/etc/security/group.conf` file present. White spaces are ignored and lines maybe extended with '\' (escaped newlines). Text following a '#' is ignored to the end of the line.

The syntax of the lines is as follows:

services;*ttys*;*users*;*times*;*groups*

The first field, the *services* field, is a logic list of PAM service names that the rule applies to.

The second field, the *tty* field, is a logic list of terminal names that this rule applies to.

The third field, the *users* field, is a logic list of users, or a UNIX group, or a netgroup of users to whom this rule applies. Group names are preceded by a '%' symbol, while netgroup names are preceded by a '@' symbol.

For these items the simple wildcard '*' may be used only once. With UNIX groups or netgroups no wildcards or logic operators are allowed.

The *times* field is used to indicate "when" these groups are to be given to the user. The format here is a logic list of day/time-range entries. The days are specified by a sequence of two character entries, MoTuSa for example is Monday Tuesday and Saturday. Note that repeated days are unset MoMo = no day, and

MoWk = all weekdays bar Monday. The two character combinations accepted are Mo Tu We Th Fr Sa Su Wk Wd Al, the last two being week-end days and all 7 days of the week respectively. As a final example, AlFr means all days except Friday.

Each day/time-range can be prefixed with a '!' to indicate "anything but". The time-range part is two 24-hour times HHMM, separated by a hyphen, indicating the start and finish time (if the finish time is smaller than the start time it is deemed to apply on the following day).

The *groups* field is a comma or space separated list of groups that the user inherits membership of. These groups are added if the previous fields are satisfied by the user's request.

For a rule to be active, ALL of service+ttys+users must be satisfied by the applying process.

6.11.3. OPTIONS

This module does not recognise any options.

6.11.4. MODULE TYPES PROVIDED

Only the `auth` module type is provided.

6.11.5. RETURN VALUES

PAM_SUCCESS	group membership was granted.
PAM_ABORT	Not all relevant data could be gotten.
PAM_BUF_ERR	Memory buffer error.
PAM_CRED_ERR	Group membership was not granted.
PAM_IGNORE	`pam_sm_authenticate` was called which does nothing.
PAM_USER_UNKNOWN	The user is not known to the system.

6.11.6. FILES

`/etc/security/group.conf` Default configuration file

6.11.7. EXAMPLES

These are some example lines which might be specified in `/etc/security/group.conf`.

Running 'xsh' on tty* (any ttyXXX device), the user 'us' is given access to the floppy (through membership of the floppy group)

```
xsh;tty*&!ttyp*;us;Al0000-2400;floppy
```

Running 'xsh' on tty* (any ttyXXX device), the users 'sword', 'pike' and 'shield' are given access to games (through membership of the floppy group) after work hours.

```
xsh; tty* ;sword|pike|shield;!Wk0900-1800;games, sound
xsh; tty* ;*;Al0900-1800;floppy
```

Any member of the group 'admin' running 'xsh' on tty*, is granted access (at any time) to the group 'plugdev'

```
xsh; tty* ;%admin;Al0000-2400;plugdev
```

6.11.8. AUTHORS

pam_group was written by Andrew G. Morgan <morgan@kernel.org>.

6.12. pam_issue - add issue file to user prompt

`pam_issue.so` [noesc] [issue=*`issue-file-name`*]

6.12.1. DESCRIPTION

pam_issue is a PAM module to prepend an issue file to the username prompt. It also by default parses escape codes in the issue file similar to some common getty's (using \x format).

Recognized escapes:

d current day

l name of this tty

m machine architecture (uname -m)

n machine's network node hostname (uname -n)

o domain name of this system

r release number of operating system (uname -r)

t current time

s operating system name (uname -s)

u number of users currently logged in

U same as \u except it is suffixed with "user" or "users" (eg. "1 user" or "10 users")

v operating system version and build date (uname -v)

6.12.2. OPTIONS

`noesc`	Turns off escape code parsing.
`issue=`*`issue-file-name`*	The file to output if not using the default.

6.12.3. MODULE TYPES PROVIDED

Only the `auth` module type is provided.

6.12.4. RETURN VALUES

PAM_BUF_ERR	Memory buffer error.
PAM_IGNORE	The prompt was already changed.
PAM_SERVICE_ERR	A service module error occurred.
PAM_SUCCESS	The new prompt was set successfully.

6.12.5. EXAMPLES

Add the following line to `/etc/pam.d/login` to set the user specific issue at login:

```
auth optional pam_issue.so issue=/etc/issue
```

6.12.6. AUTHOR

pam_issue was written by Ben Collins <bcollins@debian.org>.

6.13. pam_keyinit - display the keyinit file

`pam_keyinit.so` [debug] [force] [revoke]

6.13.1. DESCRIPTION

The pam_keyinit PAM module ensures that the invoking process has a session keyring other than the user default session keyring.

The module checks to see if the process's session keyring is the user-session-keyring(7), and, if it is, creates a new session-keyring(7) with which to replace it. If a new session keyring is created, it will install a link to the user-keyring(7) in the session keyring so that keys common to the user will be automatically accessible through it. The session keyring of the invoking process will thenceforth be inherited by all its children unless they override it.

In order to allow other PAM modules to attach tokens to the keyring, this module provides both an *auth* (limited to pam_setcred(3) and a *session* component. The session keyring is created in the module called. Moreover this module should be included as early as possible in a PAM configuration.

This module is intended primarily for use by login processes. Be aware that after the session keyring has been replaced, the old session keyring and the keys it contains will no longer be accessible.

This module should not, generally, be invoked by programs like *su*, since it is usually desirable for the key set to percolate through to the alternate context. The keys have their own permissions system to manage this.

The keyutils package is used to manipulate keys more directly. This can be obtained from:

Keyutils [http://people.redhat.com/~dhowells/keyutils/]

6.13.2. OPTIONS

`debug`	Log debug information with syslog(3).

`force` Causes the session keyring of the invoking process to be replaced unconditionally.

`revoke` Causes the session keyring of the invoking process to be revoked when the invoking process exits if the session keyring was created for this process in the first place.

6.13.3. MODULE TYPES PROVIDED

Only the `session` module type is provided.

6.13.4. RETURN VALUES

PAM_SUCCESS	This module will usually return this value
PAM_AUTH_ERR	Authentication failure.
PAM_BUF_ERR	Memory buffer error.
PAM_IGNORE	The return value should be ignored by PAM dispatch.
PAM_SERVICE_ERR	Cannot determine the user name.
PAM_SESSION_ERR	This module will return this value if its arguments are invalid or if a system error such as ENOMEM occurs.
PAM_USER_UNKNOWN	User not known.

6.13.5. EXAMPLES

Add this line to your login entries to start each login session with its own session keyring:

```
session  required  pam_keyinit.so
```

This will prevent keys from one session leaking into another session for the same user.

6.13.6. AUTHOR

pam_keyinit was written by David Howells, <dhowells@redhat.com>.

6.14. pam_lastlog - display date of last login

`pam_lastlog.so` [debug] [silent] [never] [nodate] [nohost] [noterm] [nowtmp] [noupdate] [showfailed] [inactive=<days>] [unlimited]

6.14.1. DESCRIPTION

pam_lastlog is a PAM module to display a line of information about the last login of the user. In addition, the module maintains the `/var/log/lastlog` file.

Some applications may perform this function themselves. In such cases, this module is not necessary.

The module checks `LASTLOG_UID_MAX` option in `/etc/login.defs` and does not update or display last login records for users with UID higher than its value. If the option is not present or its value is invalid, no user ID limit is applied.

If the module is called in the auth or account phase, the accounts that were not used recently enough will be disallowed to log in. The check is not performed for the root account so the root is never locked out. It is also not performed for users with UID higher than the `LASTLOG_UID_MAX` value.

6.14.2. OPTIONS

`debug`	Print debug information.
`silent`	Don't inform the user about any previous login, just update the `/var/log/lastlog` file. This option does not affect display of bad login attempts.
`never`	If the `/var/log/lastlog` file does not contain any old entries for the user, indicate that the user has never previously logged in with a welcome message.
`nodate`	Don't display the date of the last login.
`noterm`	Don't display the terminal name on which the last login was attempted.
`nohost`	Don't indicate from which host the last login was attempted.
`nowtmp`	Don't update the wtmp entry.
`noupdate`	Don't update any file.
`showfailed`	Display number of failed login attempts and the date of the last failed attempt from btmp. The date is not displayed when `nodate` is specified.
`inactive=<days>`	This option is specific for the auth or account phase. It specifies the number of days after the last login of the user when the user will be locked out by the module. The default value is 90.
`unlimited`	If the *fsize* limit is set, this option can be used to override it, preventing failures on systems with large UID values that lead lastlog to become a huge sparse file.

6.14.3. MODULE TYPES PROVIDED

The `auth` and `account` module type allows one to lock out users who did not login recently enough. The `session` module type is provided for displaying the information about the last login and/or updating the lastlog and wtmp files.

6.14.4. RETURN VALUES

PAM_SUCCESS	Everything was successful.
PAM_SERVICE_ERR	Internal service module error.

PAM_USER_UNKNOWN User not known.

PAM_AUTH_ERR User locked out in the auth or account phase due to inactivity.

PAM_IGNORE There was an error during reading the lastlog file in the auth or account phase and thus inactivity of the user cannot be determined.

6.14.5. EXAMPLES

Add the following line to `/etc/pam.d/login` to display the last login time of an user:

```
session  required  pam_lastlog.so nowtmp
```

To reject the user if he did not login during the previous 50 days the following line can be used:

```
auth  required  pam_lastlog.so inactive=50
```

6.14.6. AUTHOR

pam_lastlog was written by Andrew G. Morgan <morgan@kernel.org>.

Inactive account lock out added by Tomáš Mráz <tm@t8m.info>.

6.15. pam_limits - limit resources

`pam_limits.so` [conf=*`/path/to/limits.conf`*] [debug] [set_all] [utmp_early] [noaudit]

6.15.1. DESCRIPTION

The pam_limits PAM module sets limits on the system resources that can be obtained in a user-session. Users of *uid=0* are affected by this limits, too.

By default limits are taken from the `/etc/security/limits.conf` config file. Then individual *.conf files from the `/etc/security/limits.d/` directory are read. The files are parsed one after another in the order of "C" locale. The effect of the individual files is the same as if all the files were concatenated together in the order of parsing. If a config file is explicitly specified with a module option then the files in the above directory are not parsed.

The module must not be called by a multithreaded application.

If Linux PAM is compiled with audit support the module will report when it denies access based on limit of maximum number of concurrent login sessions.

6.15.2. DESCRIPTION

The *pam_limits.so* module applies ulimit limits, nice priority and number of simultaneous login sessions limit to user login sessions. This description of the configuration file syntax applies to the `/etc/security/limits.conf` file and `*.conf` files in the `/etc/security/limits.d` directory.

The syntax of the lines is as follows:

<domain> *<type>* *<item>* *<value>*

The fields listed above should be filled as follows:

`<domain>`

- a username
- a groupname, with *@group* syntax. This should not be confused with netgroups.
- the wildcard *, for default entry.
- the wildcard *%*, for maxlogins limit only, can also be used with *%group* syntax. If the *%* wildcard is used alone it is identical to using * with maxsyslogins limit. With a group specified after *%* it limits the total number of logins of all users that are member of the group.
- an uid range specified as *<min_uid>:<max_uid>*. If min_uid is omitted, the match is exact for the max_uid. If max_uid is omitted, all uids greater than or equal min_uid match.
- a gid range specified as *@<min_gid>:<max_gid>*. If min_gid is omitted, the match is exact for the max_gid. If max_gid is omitted, all gids greater than or equal min_gid match. For the exact match all groups including the user's supplementary groups are examined. For the range matches only the user's primary group is examined.
- a gid specified as *%:<gid>* applicable to maxlogins limit only. It limits the total number of logins of all users that are member of the group with the specified gid.

`<type>`

`hard` for enforcing *hard* resource limits. These limits are set by the superuser and enforced by the 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-existing *hard* limits. The values specified with this token can be thought of as *default* values, for normal system usage.

`-` for enforcing both *soft* and *hard* resource limits together.

Note, if you specify a type of '-' but neglect to supply the item and value fields then the module will never enforce any limits on the specified user/group etc. .

`<item>`

`core`	limits the core file size (KB)
`data`	maximum data size (KB)
`fsize`	maximum filesize (KB)
`memlock`	maximum locked-in-memory address space (KB)

`nofile`	maximum number of open file descriptors
`rss`	maximum resident set size (KB) (Ignored in Linux 2.4.30 and higher)
`stack`	maximum stack size (KB)
`cpu`	maximum CPU time (minutes)
`nproc`	maximum number of processes
`as`	address space limit (KB)
`maxlogins`	maximum number of logins for this user (this limit does not apply to user with *uid=0*)
`maxsyslogins`	maximum number of all logins on system; user is not allowed to log-in if total number of all user logins is greater than specified number (this limit does not apply to user with *uid=0*)
`priority`	the priority to run user process with (negative values boost process priority)
`locks`	maximum locked files (Linux 2.4 and higher)
`sigpending`	maximum number of pending signals (Linux 2.6 and higher)
`msgqueue`	maximum memory used by POSIX message queues (bytes) (Linux 2.6 and higher)
`nice`	maximum nice priority allowed to raise to (Linux 2.6.12 and higher) values: [-20,19]
`rtprio`	maximum realtime priority allowed for non-privileged processes (Linux 2.6.12 and higher)

All items support the values *-1*, *unlimited* or *infinity* indicating no limit, except for *priority* and *nice*.

If a hard limit or soft limit of a resource is set to a valid value, but outside of the supported range of the local system, the system may reject the new limit or unexpected behavior may occur. If the control value *required* is used, the module will reject the login if a limit could not be set.

In general, 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. One exception is the *maxlogin* option, this one is system wide. But there is a race, concurrent logins at the same time will not always be detect as such but only counted as one.

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

The pam_limits module does report configuration problems found in its configuration file and errors via syslog(3).

6.15.3. OPTIONS

`conf=`*`/path/to/limits.conf`*	Indicate an alternative limits.conf style configuration file to override the default.
`debug`	Print debug information.
`set_all`	Set the limits for which no value is specified in the configuration file to the one from the process with the PID 1.
`utmp_early`	Some broken applications actually allocate a utmp entry for the user before the user is admitted to the system. If some of the services you are configuring PAM for do this, you can selectively use this module argument to compensate for this behavior and at the same time maintain system-wide consistency with a single limits.conf file.
`noaudit`	Do not report exceeded maximum logins count to the audit subsystem.

6.15.4. MODULE TYPES PROVIDED

Only the `session` module type is provided.

6.15.5. RETURN VALUES

PAM_ABORT	Cannot get current limits.
PAM_IGNORE	No limits found for this user.
PAM_PERM_DENIED	New limits could not be set.
PAM_SERVICE_ERR	Cannot read config file.
PAM_SESSION_ERR	Error recovering account name.
PAM_SUCCESS	Limits were changed.
PAM_USER_UNKNOWN	The user is not known to the system.

6.15.6. FILES

`/etc/security/limits.conf`	Default configuration file

6.15.7. EXAMPLES

These are some example lines which might be specified in `/etc/security/limits.conf`.

```
*               soft    core            0
*               hard    nofile          512
@student        hard    nproc           20
@faculty        soft    nproc           20
```

```
@faculty        hard    nproc           50
ftp             hard    nproc           0
@student        -       maxlogins       4
:123            hard    cpu             5000
@500:           soft    cpu             10000
600:700         hard    locks           10
```

6.15.8. AUTHORS

pam_limits was initially written by Cristian Gafton <gafton@redhat.com>

6.16. pam_listfile - deny or allow services based on an arbitrary file

`pam_listfile.so` item=[tty|user|rhost|ruser|group|shell] sense=[allow|deny] file=*`/path/filename`* onerr=[succeed|fail] [apply=[*`user`*|*`@group`*]] [quiet]

6.16.1. DESCRIPTION

pam_listfile is a PAM module which provides a way to deny or allow services based on an arbitrary file.

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.

An additional argument, `apply=`, can be used to restrict the application of the above to a specific user (`apply=`*`username`*) or a given group (`apply=`*`@groupname`*). This added restriction is only meaningful when used with the *tty*, *rhost* and *shell* items.

Besides this last one, all arguments should be specified; do not count on any default behavior.

No credentials are awarded by this module.

6.16.2. OPTIONS

`item=[tty|user|rhost|ruser|group|shell]`	What is listed in the file and should be checked for.
`sense=[allow|deny]`	Action to take if found in file, if the item is NOT found in the file, then the opposite action is requested.
`file=`*`/path/filename`*	File containing one item per line. The file needs to be a plain file and not world writable.

`onerr=[succeed|fail]`	What to do if something weird happens like being unable to open the file.
`apply=[`*`user`*`|`*`@group`*`]`	Restrict the user class for which the restriction apply. Note that with `item=[user|ruser|group]` this does not make sense, but for `item=[tty|rhost|shell]` it have a meaning.
`quiet`	Do not treat service refusals or missing list files as errors that need to be logged.

6.16.3. MODULE TYPES PROVIDED

All module types (`auth`, `account`, `password` and `session`) are provided.

6.16.4. RETURN VALUES

PAM_AUTH_ERR	Authentication failure.
PAM_BUF_ERR	Memory buffer error.
PAM_IGNORE	The rule does not apply to the `apply` option.
PAM_SERVICE_ERR	Error in service module.
PAM_SUCCESS	Success.

6.16.5. EXAMPLES

Classic 'ftpusers' authentication can be implemented with this entry in `/etc/pam.d/ftpd`:

```
#
# deny ftp-access to users listed in the /etc/ftpusers file
#
auth    required       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 a `/etc/pam.d/login` entry like this:

```
#
# permit login to users listed in /etc/loginusers
#
auth    required       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.

6.16.6. AUTHOR

pam_listfile was written by Michael K. Johnson <johnsonm@redhat.com> and Elliot Lee <sopwith@cuc.edu>.

6.17. pam_localuser - require users to be listed in /etc/passwd

`pam_localuser.so` [debug] [file=*`/path/passwd`*]

6.17.1. DESCRIPTION

pam_localuser is a PAM module to help implementing site-wide login policies, where they typically include a subset of the network's users and a few accounts that are local to a particular workstation. Using pam_localuser and pam_wheel or pam_listfile is an effective way to restrict access to either local users and/or a subset of the network's users.

This could also be implemented using pam_listfile.so and a very short awk script invoked by cron, but it's common enough to have been separated out.

6.17.2. OPTIONS

`debug`	Print debug information.
`file=`*`/path/passwd`*	Use a file other than `/etc/passwd`.

6.17.3. MODULE TYPES PROVIDED

All module types (`account`, `auth`, `password` and `session`) are provided.

6.17.4. RETURN VALUES

PAM_SUCCESS	The new localuser was set successfully.
PAM_SERVICE_ERR	No username was given.
PAM_PERM_DENIED	The user is not listed in the passwd file.

6.17.5. EXAMPLES

Add the following lines to `/etc/pam.d/su` to allow only local users or group wheel to use su.

```
account sufficient pam_localuser.so
account required pam_wheel.so
```

6.17.6. AUTHOR

pam_localuser was written by Nalin Dahyabhai <nalin@redhat.com>.

6.18. pam_loginuid - record user's login uid to the process attribute

`pam_loginuid.so` [require_auditd]

6.18.1. DESCRIPTION

The pam_loginuid module sets the loginuid process attribute for the process that was authenticated. This is necessary for applications to be correctly audited. This PAM module should only be used for entry point applications like: login, sshd, gdm, vsftpd, crond and atd. There are probably other entry point applications besides these. You should not use it for applications like sudo or su as that defeats the purpose by changing the loginuid to the account they just switched to.

6.18.2. OPTIONS

`require_auditd`	This option, when given, will cause this module to query the audit daemon status and deny logins if it is not running.

6.18.3. MODULE TYPES PROVIDED

Only the `session` module type is provided.

6.18.4. RETURN VALUES

PAM_SUCCESS	The loginuid value is set and auditd is running if check requested.
PAM_IGNORE	The /proc/self/loginuid file is not present on the system or the login process runs inside uid namespace and kernel does not support overwriting loginuid.
PAM_SESSION_ERR	Any other error prevented setting loginuid or auditd is not running.

6.18.5. EXAMPLES

```
#%PAM-1.0
auth       required     pam_unix.so
auth       required     pam_nologin.so
account    required     pam_unix.so
password   required     pam_unix.so
session    required     pam_unix.so
session    required     pam_loginuid.so
```

6.18.6. AUTHOR

pam_loginuid was written by Steve Grubb <sgrubb@redhat.com>

6.19. pam_mail - inform about available mail

`pam_mail.so` [close] [debug] [dir=*`maildir`*] [empty] [hash=*`count`*] [noenv] [nopen] [quiet] [standard]

6.19.1. DESCRIPTION

The pam_mail PAM module provides the "you have new mail" service to the user. It can be plugged into any application that has credential or session hooks. It gives a single message indicating the *newness* of any mail it finds in the user's mail folder. This module also sets the PAM environment variable, *MAIL*, to the user's mail directory.

If the mail spool file (be it `/var/mail/$USER` or a pathname given with the `dir=` parameter) is a directory then pam_mail assumes it is in the *Maildir* format.

6.19.2. OPTIONS

`close`	Indicate if the user has any mail also on logout.
`debug`	Print debug information.
`dir=`*`maildir`*	Look for the user's mail in an alternative location defined by `maildir/<login>`. The default location for mail is `/var/mail/<login>`. Note, if the supplied `maildir` is prefixed by a '~', the directory is interpreted as indicating a file in the user's home directory.
`empty`	Also print message if user has no mail.
`hash=`*`count`*	Mail directory hash depth. For example, a *hashcount* of 2 would make the mail file be `/var/spool/mail/u/s/user`.
`noenv`	Do not set the *MAIL* environment variable.
`nopen`	Don't print any mail information on login. This flag is useful to get the *MAIL* environment variable set, but to not display any information about it.
`quiet`	Only report when there is new mail.
`standard`	Old style "You have..." format which doesn't show the mail spool being used. This also implies "empty".

6.19.3. MODULE TYPES PROVIDED

The `session` and `auth` (on establishment and deletion of credentials) module types are provided.

6.19.4. RETURN VALUES

PAM_BUF_ERR	Memory buffer error.
PAM_SERVICE_ERR	Badly formed arguments.
PAM_SUCCESS	Success.
PAM_USER_UNKNOWN	User not known.

6.19.5. EXAMPLES

Add the following line to `/etc/pam.d/login` to indicate that the user has new mail when they login to the system.

```
session  optional  pam_mail.so standard
```

6.19.6. AUTHOR

pam_mail was written by Andrew G. Morgan <morgan@kernel.org>.

6.20. pam_mkhomedir - create users home directory

`pam_mkhomedir.so` [silent] [debug] [umask=*`mode`*] [skel=*`skeldir`*]

6.20.1. DESCRIPTION

The pam_mkhomedir PAM module will create a users home directory if it does not exist when the session begins. This allows users to be present in central database (such as NIS, kerberos or LDAP) without using a distributed file system or pre-creating a large number of directories. The skeleton directory (usually `/etc/skel/`) is used to copy default files and also sets a umask for the creation.

The new users home directory will not be removed after logout of the user.

6.20.2. OPTIONS

`silent`	Don't print informative messages.
`debug`	Turns on debugging via syslog(3).
`umask=`*`mask`*	The user file-creation mask is set to *`mask`*. The default value of mask is 0022.
`skel=`*`/path/to/skel/directory`*	Indicate an alternative `skel` directory to override the default `/etc/skel`.

6.20.3. MODULE TYPES PROVIDED

Only the `session` module type is provided.

6.20.4. RETURN VALUES

PAM_BUF_ERR	Memory buffer error.
PAM_CRED_INSUFFICIENT	Insufficient credentials to access authentication data.
PAM_PERM_DENIED	Not enough permissions to create the new directory or read the skel directory.
PAM_USER_UNKNOWN	User not known to the underlying authentication module.
PAM_SUCCESS	Environment variables were set.

6.20.5. EXAMPLES

A sample /etc/pam.d/login file:

```
auth       requisite   pam_securetty.so
auth       sufficient  pam_ldap.so
auth       required    pam_unix.so
auth       required    pam_nologin.so
account    sufficient  pam_ldap.so
account    required    pam_unix.so
password   required    pam_unix.so
session    required    pam_mkhomedir.so skel=/etc/skel/ umask=0022
session    required    pam_unix.so
session    optional    pam_lastlog.so
session    optional    pam_mail.so standard
```

6.20.6. AUTHOR

pam_mkhomedir was written by Jason Gunthorpe <jgg@debian.org>.

6.21. pam_motd - display the motd file

`pam_motd.so` [motd=*`/path/filename`*] [motd_dir=*`/path/dirname.d`*]

6.21.1. DESCRIPTION

pam_motd is a PAM module that can be used to display arbitrary motd (message of the day) files after a successful login. By default, pam_motd shows files in the following locations:

```
/etc/motd
/run/motd
/usr/lib/motd
/etc/motd.d/
/run/motd.d/
/usr/lib/motd.d/
```

Each message size is limited to 64KB.

If `/etc/motd` does not exist, then `/run/motd` is shown. If `/run/motd` does not exist, then `/usr/lib/motd` is shown.

Similar overriding behavior applies to the directories. Files in `/etc/motd.d/` override files with the same name in `/run/motd.d/` and `/usr/lib/motd.d/`. Files in `/run/motd.d/` override files with the same name in `/usr/lib/motd.d/`.

Files the in the directories listed above are displayed in lexicographic order by name.

To silence a message, a symbolic link with target `/dev/null` may be placed in `/etc/motd.d` with the same filename as the message to be silenced. Example: Creating a symbolic link as follows silences `/usr/lib/motd.d/my_motd`.

ln -s /dev/null /etc/motd.d/my_motd

The *MOTD_SHOWN=pam* environment variable is set after showing the motd files, even when all of them were silenced using symbolic links.

6.21.2. OPTIONS

`motd=`*`/path/filename`* The `/path/filename` file is displayed as message of the day. Multiple paths to try can be specified as a colon-separated list. By default this option is set to `/etc/motd:/run/motd:/usr/lib/motd`.

`motd_dir=`*`/path/dirname.d`* The `/path/dirname.d` directory is scanned and each file contained inside of it is displayed. Multiple directories to scan can be specified as a colon-separated list. By default this option is set to `/etc/motd.d:/run/motd.d:/usr/lib/motd.d`.

When no options are given, the default behavior applies for both options. Specifying either option (or both) will disable the default behavior for both options.

6.21.3. MODULE TYPES PROVIDED

Only the `session` module type is provided.

6.21.4. RETURN VALUES

PAM_ABORT Not all relevant data or options could be obtained.

PAM_BUF_ERR Memory buffer error.

PAM_IGNORE This is the default return value of this module.

6.21.5. EXAMPLES

The suggested usage for `/etc/pam.d/login` is:

```
session  optional  pam_motd.so
```

To use a `motd` file from a different location:

```
session  optional  pam_motd.so motd=/elsewhere/motd
```

To use a `motd` file from elsewhere, along with a corresponding `.d` directory:

```
session  optional  pam_motd.so motd=/elsewhere/motd motd_dir=/elsewhere/motd.d
```

6.21.6. AUTHOR

pam_motd was written by Ben Collins <bcollins@debian.org>.

The `motd_dir=` option was added by Allison Karlitskaya <allison.karlitskaya@redhat.com>.

6.22. pam_namespace - setup a private namespace

`pam_namespace.so` [debug] [unmnt_remnt] [unmnt_only] [require_selinux] [gen_hash] [ignore_config_error] [ignore_instance_parent_mode] [unmount_on_close] [use_current_context] [use_default_context] [mount_private]

6.22.1. DESCRIPTION

The pam_namespace PAM module sets up a private namespace for a session with polyinstantiated directories. A polyinstantiated directory provides a different instance of itself based on user name, or when using SELinux, user name, security context or both. If an executable script `/etc/security/namespace.init` exists, it is used to initialize the instance directory after it is set up and mounted on the polyinstantiated directory. The script receives the polyinstantiated directory path, the instance directory path, flag whether the instance directory was newly created (0 for no, 1 for yes), and the user name as its arguments.

The pam_namespace module disassociates the session namespace from the parent namespace. Any mounts/unmounts performed in the parent namespace, such as mounting of devices, are not reflected in the session namespace. To propagate selected mount/unmount events from the parent namespace into the disassociated session namespace, an administrator may use the special shared-subtree feature. For additional information on shared-subtree feature, please refer to the mount(8) man page and the shared-subtree description at http://lwn.net/Articles/159077 and http://lwn.net/Articles/159092.

6.22.2. DESCRIPTION

The *pam_namespace.so* module allows setup of private namespaces with polyinstantiated directories. Directories can be polyinstantiated based on user name or, in the case of SELinux, user name, sensitivity level or complete security context. If an executable script `/etc/security/namespace.init` exists, it is used to initialize the namespace every time an instance directory is set up and mounted. The script receives the polyinstantiated directory path and the instance directory path as its arguments.

The `/etc/security/namespace.conf` file specifies which directories are polyinstantiated, how they are polyinstantiated, how instance directories would be named, and any users for whom polyinstantiation would not be performed.

When someone logs in, the file `namespace.conf` is scanned. Comments are marked by # characters. Each non comment line represents one polyinstantiated directory. The fields are separated by spaces but can be quoted by *"* characters also escape sequences *\b*, *\n*, and *\t* are recognized. The fields are as follows:

`polydir instance_prefix method list_of_uids`

The first field, *`polydir`*, is the absolute pathname of the directory to polyinstantiate. The special string *$HOME* is replaced with the user's home directory, and *$USER* with the username. This field cannot be blank.

The second field, *`instance_prefix`* is the string prefix used to build the pathname for the instantiation of <polydir>. Depending on the polyinstantiation *`method`* it is then appended with "instance differentiation string" to generate the final instance directory path. This directory is created if it did not exist already, and is then bind mounted on the <polydir> to provide an instance of <polydir> based on

the <method> column. The special string *$HOME* is replaced with the user's home directory, and *$USER* with the username. This field cannot be blank.

The third field, *method*, is the method used for polyinstantiation. It can take these values; "user" for polyinstantiation based on user name, "level" for polyinstantiation based on process MLS level and user name, "context" for polyinstantiation based on process security context and user name, "tmpfs" for mounting tmpfs filesystem as an instance dir, and "tmpdir" for creating temporary directory as an instance dir which is removed when the user's session is closed. Methods "context" and "level" are only available with SELinux. This field cannot be blank.

The fourth field, *list_of_uids*, is a comma separated list of user names for whom the polyinstantiation is not performed. If left blank, polyinstantiation will be performed for all users. If the list is preceded with a single "~" character, polyinstantiation is performed only for users in the list.

The *method* field can contain also following optional flags separated by *:* characters.

create=mode,owner,group - create the polyinstantiated directory. The mode, owner and group parameters are optional. The default for mode is determined by umask, the default owner is the user whose session is opened, the default group is the primary group of the user.

iscript=path - path to the instance directory init script. The base directory for relative paths is `/etc/security/namespace.d`.

noinit - instance directory init script will not be executed.

shared - the instance directories for "context" and "level" methods will not contain the user name and will be shared among all users.

mntopts=value - value of this flag is passed to the mount call when the tmpfs mount is done. It allows for example the specification of the maximum size of the tmpfs instance that is created by the mount call. In addition to options specified in the tmpfs(5) manual the *nosuid*, *noexec*, and *nodev* flags can be used to respectively disable setuid bit effect, disable running executables, and disable devices to be interpreted on the mounted tmpfs filesystem.

The directory where polyinstantiated instances are to be created, must exist and must have, by default, the mode of 0000. The requirement that the instance parent be of mode 0000 can be overridden with the command line option *ignore_instance_parent_mode*

In case of context or level polyinstantiation the SELinux context which is used for polyinstantiation is the context used for executing a new process as obtained by getexeccon. This context must be set by the calling application or `pam_selinux.so` module. If this context is not set the polyinstatiation will be based just on user name.

The "instance differentiation string" is <user name> for "user" method and <user name>_<raw directory context> for "context" and "level" methods. If the whole string is too long the end of it is replaced with md5sum of itself. Also when command line option *gen_hash* is used the whole string is replaced with md5sum of itself.

6.22.3. OPTIONS

`debug`	A lot of debug information is logged using syslog
`unmnt_remnt`	For programs such as su and newrole, the login session has already setup a polyinstantiated namespace. For these

	programs, polyinstantiation is performed based on new user id or security context, however the command first needs to undo the polyinstantiation performed by login. This argument instructs the command to first undo previous polyinstantiation before proceeding with new polyinstantiation based on new id/context
`unmnt_only`	For trusted programs that want to undo any existing bind mounts and process instance directories on their own, this argument allows them to unmount currently mounted instance directories
`require_selinux`	If selinux is not enabled, return failure
`gen_hash`	Instead of using the security context string for the instance name, generate and use its md5 hash.
`ignore_config_error`	If a line in the configuration file corresponding to a polyinstantiated directory contains format error, skip that line process the next line. Without this option, pam will return an error to the calling program resulting in termination of the session.
`ignore_instance_parent_mode`	Instance parent directories by default are expected to have the restrictive mode of 000. Using this option, an administrator can choose to ignore the mode of the instance parent. This option should be used with caution as it will reduce security and isolation goals of the polyinstantiation mechanism.
`unmount_on_close`	Explicitly unmount the polyinstantiated directories instead of relying on automatic namespace destruction after the last process in a namespace exits. This option should be used only in case it is ensured by other means that there cannot be any processes running in the private namespace left after the session close. It is also useful only in case there are multiple pam session calls in sequence from the same process.
`use_current_context`	Useful for services which do not change the SELinux context with setexeccon call. The module will use the current SELinux context of the calling process for the level and context polyinstantiation.
`use_default_context`	Useful for services which do not use pam_selinux for changing the SELinux context with setexeccon call. The module will use the default SELinux context of the user for the level and context polyinstantiation.
`mount_private`	This option can be used on systems where the / mount point or its submounts are made shared (for example with a **mount --make-rshared /** command). The module will mark the whole directory tree so any mount and unmount operations in the polyinstantiation namespace are private. Normally the pam_namespace will try to detect the shared / mount point and make the polyinstantiated directories private automatically. This option has to be used just when only a subtree is shared and / is not. Note that mounts and unmounts done in the private namespace will not affect the parent namespace if this option is used or when the shared / mount point is autodetected.

6.22.4. MODULE TYPES PROVIDED

Only the `session` module type is provided. The module must not be called from multithreaded processes.

6.22.5. RETURN VALUES

PAM_SUCCESS	Namespace setup was successful.
PAM_SERVICE_ERR	Unexpected system error occurred while setting up namespace.
PAM_SESSION_ERR	Unexpected namespace configuration error occurred.

6.22.6. FILES

`/etc/security/namespace.conf`	Main configuration file
`/etc/security/namespace.d`	Directory for additional configuration files
`/etc/security/namespace.init`	Init script for instance directories

6.22.7. EXAMPLES

These are some example lines which might be specified in `/etc/security/namespace.conf`.

```
# The following three lines will polyinstantiate /tmp,
# /var/tmp and user's home directories. /tmp and /var/tmp
# will be polyinstantiated based on the security level
# as well as user name, whereas home directory will be
# polyinstantiated based on the full security context and user name.
# Polyinstantiation will not be performed for user root
# and adm for directories /tmp and /var/tmp, whereas home
# directories will be polyinstantiated for all users.
#
# Note that instance directories do not have to reside inside
# the polyinstantiated directory. In the examples below,
# instances of /tmp will be created in /tmp-inst directory,
# where as instances of /var/tmp and users home directories
# will reside within the directories that are being
# polyinstantiated.
#
/tmp     /tmp-inst/              level      root,adm
/var/tmp /var/tmp/tmp-inst/     level      root,adm
$HOME    $HOME/$USER.inst/inst- context
```

For the <service>s you need polyinstantiation (login for example) put the following line in /etc/pam.d/<service> as the last line for session group:

session required pam_namespace.so [arguments]

This module also depends on pam_selinux.so setting the context.

6.22.8. AUTHORS

The namespace setup scheme was designed by Stephen Smalley, Janak Desai and Chad Sellers. The pam_namespace PAM module was developed by Janak Desai <janak@us.ibm.com>, Chad Sellers <csellers@tresys.com> and Steve Grubb <sgrubb@redhat.com>. Additional improvements by Xavier Toth <txtoth@gmail.com> and Tomas Mraz <tmraz@redhat.com>.

6.23. pam_nologin - prevent non-root users from login

`pam_nologin.so` [file=*`/path/nologin`*] [successok]

6.23.1. DESCRIPTION

pam_nologin is a PAM module that prevents users from logging into the system when `/var/run/nologin` or `/etc/nologin` exists. The contents of the file are displayed to the user. The pam_nologin module has no effect on the root user's ability to log in.

6.23.2. OPTIONS

`file=`*`/path/nologin`*	Use this file instead the default `/var/run/nologin` or `/etc/nologin`.
`successok`	Return PAM_SUCCESS if no file exists, the default is PAM_IGNORE.

6.23.3. MODULE TYPES PROVIDED

The `auth` and `account` module types are provided.

6.23.4. RETURN VALUES

PAM_AUTH_ERR	The user is not root and `/etc/nologin` exists, so the user is not permitted to log in.
PAM_BUF_ERR	Memory buffer error.
PAM_IGNORE	This is the default return value.
PAM_SUCCESS	Success: either the user is root or the nologin file does not exist.
PAM_USER_UNKNOWN	User not known to the underlying authentication module.

6.23.5. EXAMPLES

The suggested usage for `/etc/pam.d/login` is:

```
auth  required  pam_nologin.so
```

6.23.6. AUTHOR

pam_nologin was written by Michael K. Johnson <johnsonm@redhat.com>.

6.24. pam_permit - the promiscuous module

`pam_permit.so`

6.24.1. DESCRIPTION

pam_permit is a PAM module that always permit access. It does nothing else.

In the case of authentication, the user's name will be set to *nobody* if the application didn't set one. Many applications and PAM modules become confused if this name is unknown.

This module is very dangerous. It should be used with extreme caution.

6.24.2. OPTIONS

This module does not recognise any options.

6.24.3. MODULE TYPES PROVIDED

The `auth`, `account`, `password` and `session` module types are provided.

6.24.4. RETURN VALUES

PAM_SUCCESS This module always returns this value.

6.24.5. EXAMPLES

Add this line to your other login entries to disable account management, but continue to permit users to log in.

```
account  required  pam_permit.so
```

6.24.6. AUTHOR

pam_permit was written by Andrew G. Morgan, <morgan@kernel.org>.

6.25. pam_pwhistory - grant access using .pwhistory file

`pam_pwhistory.so` [debug] [use_authtok] [enforce_for_root] [remember=*N*] [retry=*N*] [authtok_type=*STRING*]

6.25.1. DESCRIPTION

This module saves the last passwords for each user in order to force password change history and keep the user from alternating between the same password too frequently.

This module does not work together with kerberos. In general, it does not make much sense to use this module in conjunction with NIS or LDAP, since the old passwords are stored on the local machine and are not available on another machine for password history checking.

6.25.2. OPTIONS

`debug`	Turns on debugging via syslog(3).
`use_authtok`	When password changing enforce the module to use the new password provided by a previously stacked `password` module (this is used in the example of the stacking of the **pam_cracklib** module documented below).
`enforce_for_root`	If this option is set, the check is enforced for root, too.
`remember=`*`N`*	The last *`N`* passwords for each user are saved in `/etc/security/opasswd`. The default is *10*. Value of *0* makes the module to keep the existing contents of the `opasswd` file unchanged.
`retry=`*`N`*	Prompt user at most *`N`* times before returning with error. The default is *1*.
`authtok_type=`*`STRING`*	See pam_get_authtok(3) for more details.

6.25.3. MODULE TYPES PROVIDED

Only the `password` module type is provided.

6.25.4. RETURN VALUES

PAM_AUTHTOK_ERR	No new password was entered, the user aborted password change or new password couldn't be set.
PAM_IGNORE	Password history was disabled.
PAM_MAXTRIES	Password was rejected too often.
PAM_USER_UNKNOWN	User is not known to system.

6.25.5. FILES

`/etc/security/opasswd`	File with password history

6.25.6. EXAMPLES

An example password section would be:

```
#%PAM-1.0
password     required       pam_pwhistory.so
password     required       pam_unix.so        use_authtok
```

In combination with **pam_cracklib**:

```
#%PAM-1.0
password     required       pam_cracklib.so    retry=3
password     required       pam_pwhistory.so   use_authtok
password     required       pam_unix.so        use_authtok
```

6.25.7. AUTHOR

pam_pwhistory was written by Thorsten Kukuk <kukuk@thkukuk.de>

6.26. pam_rhosts - grant access using .rhosts file

`pam_rhosts.so`

6.26.1. DESCRIPTION

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

The authentication mechanism of this module is based on the contents of two files; `/etc/hosts.equiv` (or 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.

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.

6.26.2. OPTIONS

`debug`	Print debug information.
`silent`	Don't print informative messages.
`superuser=`*`account`*	Handle *`account`* as root.

6.26.3. MODULE TYPES PROVIDED

Only the `auth` module type is provided.

6.26.4. RETURN VALUES

PAM_AUTH_ERR	The remote host, remote user name or the local user name couldn't be determined or access was denied by `.rhosts` file.

PAM_USER_UNKNOWN User is not known to system.

6.26.5. EXAMPLES

To grant a remote user access by `/etc/hosts.equiv` or `.rhosts` for **rsh** add the following lines to `/etc/pam.d/rsh`:

```
#%PAM-1.0
#
auth     required       pam_rhosts.so
auth     required       pam_nologin.so
auth     required       pam_env.so
auth     required       pam_unix.so
```

6.26.6. AUTHOR

pam_rhosts was written by Thorsten Kukuk <kukuk@thkukuk.de>

6.27. pam_rootok - gain only root access

`pam_rootok.so` [debug]

6.27.1. DESCRIPTION

pam_rootok is a PAM module that 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.

6.27.2. OPTIONS

`debug` Print debug information.

6.27.3. MODULE TYPES PROVIDED

The `auth`, `account` and `password` module types are provided.

6.27.4. RETURN VALUES

PAM_SUCCESS The *UID* is *0*.

PAM_AUTH_ERR The *UID* is *not 0*.

6.27.5. EXAMPLES

In the case of the su(1) 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 with PAM the following pair of lines are needed for the corresponding entry in the `/etc/pam.d/su` configuration file:

```
# su authentication. Root is granted access by default.
auth  sufficient   pam_rootok.so
auth  required     pam_unix.so
```

6.27.6. AUTHOR

pam_rootok was written by Andrew G. Morgan, <morgan@kernel.org>.

6.28. pam_securetty - limit root login to special devices

`pam_securetty.so` [debug]

6.28.1. DESCRIPTION

pam_securetty is a PAM module that allows root logins only if the user is logging in on a "secure" tty, as defined by the listing in the `securetty` file. pam_securetty checks at first, if `/etc/securetty` exists. If not and it was built with vendordir support, it will use `%vendordir%/securetty`. pam_securetty also checks that the `securetty` files are plain files and not world writable. It will also allow root logins on the tty specified with `console=` switch on the kernel command line and on ttys from the `/sys/class/tty/console/active`.

This module has no effect on non-root users and requires that the application fills in the *PAM_TTY* item correctly.

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

6.28.2. OPTIONS

`debug`	Print debug information.
`noconsole`	Do not automatically allow root logins on the kernel console device, as specified on the kernel command line or by the sys file, if it is not also specified in the `securetty` file.

6.28.3. MODULE TYPES PROVIDED

Only the `auth` module type is provided.

6.28.4. RETURN VALUES

PAM_SUCCESS	The user is allowed to continue authentication. Either the user is not root, or the root user is trying to log in on an acceptable device.
PAM_AUTH_ERR	Authentication is rejected. Either root is attempting to log in via an unacceptable device, or the `securetty` file is world writable or not a normal file.
PAM_INCOMPLETE	An application error occurred. pam_securetty was not able to get information it required from the application that called it.

PAM_SERVICE_ERR An error occurred while the module was determining the user's name or tty, or the module could not open the `securetty` file.

PAM_USER_UNKNOWN The module could not find the user name in the `/etc/passwd` file to verify whether the user had a UID of 0. Therefore, the results of running this module are ignored.

6.28.5. EXAMPLES

```
auth  required  pam_securetty.so
auth  required  pam_unix.so
```

6.28.6. AUTHOR

pam_securetty was written by Elliot Lee <sopwith@cuc.edu>.

6.29. pam_selinux - set the default security context

`pam_selinux.so` [open] [close] [restore] [nottys] [debug] [verbose] [select_context] [env_params] [use_current_range]

6.29.1. DESCRIPTION

pam_selinux is a PAM module that sets up the default SELinux security context for the next executed process.

When a new session is started, the open_session part of the module computes and sets up the execution security context used for the next execve(2) call, the file security context for the controlling terminal, and the security context used for creating a new kernel keyring.

When the session is ended, the close_session part of the module restores old security contexts that were in effect before the change made by the open_session part of the module.

Adding pam_selinux into the PAM stack might disrupt behavior of other PAM modules which execute applications. To avoid that, *pam_selinux.so open* should be placed after such modules in the PAM stack, and *pam_selinux.so close* should be placed before them. When such a placement is not feasible, *pam_selinux.so restore* could be used to temporary restore original security contexts.

6.29.2. OPTIONS

`open` Only execute the open_session part of the module.

`close` Only execute the close_session part of the module.

`restore` In open_session part of the module, temporarily restore the security contexts as they were before the previous call of the module. Another call of this module without the restore option will set up the new security contexts again.

`nottys`	Do not setup security context of the controlling terminal.
`debug`	Turn on debug messages via syslog(3).
`verbose`	Attempt to inform the user when security context is set.
`select_context`	Attempt to ask the user for a custom security context role. If MLS is on, ask also for sensitivity level.
`env_params`	Attempt to obtain a custom security context role from PAM environment. If MLS is on, obtain also sensitivity level. This option and the select_context option are mutually exclusive. The respective PAM environment variables are *SELINUX_ROLE_REQUESTED*, *SELINUX_LEVEL_REQUESTED*, and *SELINUX_USE_CURRENT_RANGE*. The first two variables are self describing and the last one if set to 1 makes the PAM module behave as if the use_current_range was specified on the command line of the module.
`use_current_range`	Use the sensitivity level of the current process for the user context instead of the default level. Also suppresses asking of the sensitivity level from the user or obtaining it from PAM environment.

6.29.3. MODULE TYPES PROVIDED

Only the `session` module type is provided.

6.29.4. RETURN VALUES

PAM_SUCCESS	The security context was set successfully.
PAM_SESSION_ERR	Unable to get or set a valid context.
PAM_USER_UNKNOWN	The user is not known to the system.
PAM_BUF_ERR	Memory allocation error.

6.29.5. EXAMPLES

```
auth     required  pam_unix.so
session  required  pam_permit.so
session  optional  pam_selinux.so
```

6.29.6. AUTHOR

pam_selinux was written by Dan Walsh <dwalsh@redhat.com>.

6.30. pam_shells - check for valid login shell

`pam_shells.so`

6.30.1. DESCRIPTION

pam_shells is a PAM module that only allows access to the system if the user's shell is listed in `/etc/shells`.

It also checks if `/etc/shells` is a plain file and not world writable.

6.30.2. OPTIONS

This module does not recognise any options.

6.30.3. MODULE TYPES PROVIDED

The `auth` and `account` module types are provided.

6.30.4. RETURN VALUES

PAM_AUTH_ERR	Access to the system was denied.
PAM_SUCCESS	The user's login shell was listed as valid shell in `/etc/shells`.
PAM_SERVICE_ERR	The module was not able to get the name of the user.

6.30.5. EXAMPLES

```
auth  required  pam_shells.so
```

6.30.6. AUTHOR

pam_shells was written by Erik Troan <ewt@redhat.com>.

6.31. pam_succeed_if - test account characteristics

`pam_succeed_if.so` [*`flag`*...] [*`condition`*...]

6.31.1. DESCRIPTION

pam_succeed_if.so is designed to succeed or fail authentication based on characteristics of the account belonging to the user being authenticated or values of other PAM items. One use is to select whether to load other modules based on this test.

The module should be given one or more conditions as module arguments, and authentication will succeed only if all of the conditions are met.

6.31.2. OPTIONS

The following *flag*s are supported:

`debug`	Turns on debugging messages sent to syslog.
`use_uid`	Evaluate conditions using the account of the user whose UID the application is running under instead of the user being authenticated.
`quiet`	Don't log failure or success to the system log.
`quiet_fail`	Don't log failure to the system log.
`quiet_success`	Don't log success to the system log.
`audit`	Log unknown users to the system log.

*Condition*s are three words: a field, a test, and a value to test for.

Available fields are *user*, *uid*, *gid*, *shell*, *home*, *ruser*, *rhost*, *tty* and *service*:

`field < number`	Field has a value numerically less than number.
`field <= number`	Field has a value numerically less than or equal to number.
`field eq number`	Field has a value numerically equal to number.
`field >= number`	Field has a value numerically greater than or equal to number.
`field > number`	Field has a value numerically greater than number.
`field ne number`	Field has a value numerically different from number.
`field = string`	Field exactly matches the given string.
`field != string`	Field does not match the given string.
`field =~ glob`	Field matches the given glob.
`field !~ glob`	Field does not match the given glob.
`field in item:item:...`	Field is contained in the list of items separated by colons.
`field notin item:item:...`	Field is not contained in the list of items separated by colons.
`user ingroup group`	User is in given group.
`user notingroup group`	User is not in given group.
`user innetgr netgroup`	(user,host) is in given netgroup.
`user notinnetgr group`	(user,host) is not in given netgroup.

6.31.3. MODULE TYPES PROVIDED

All module types (`account`, `auth`, `password` and `session`) are provided.

6.31.4. RETURN VALUES

PAM_SUCCESS	The condition was true.

PAM_AUTH_ERR The condition was false.

PAM_SERVICE_ERR A service error occurred or the arguments can't be parsed correctly.

6.31.5. EXAMPLES

To emulate the behaviour of *pam_wheel*, except there is no fallback to group 0:

```
auth required pam_succeed_if.so quiet user ingroup wheel
```

Given that the type matches, only loads the othermodule rule if the UID is over 500. Adjust the number after default to skip several rules.

```
type [default=1 success=ignore] pam_succeed_if.so quiet uid > 500
type required othermodule.so arguments...
```

6.31.6. AUTHOR

Nalin Dahyabhai <nalin@redhat.com>

6.32. pam_tally - login counter (tallying) module

`pam_tally.so` [file=*`/path/to/counter`*] [onerr=[*`fail`*|*`succeed`*]] [magic_root] [even_deny_root_account] [deny=*`n`*] [lock_time=*`n`*] [unlock_time=*`n`*] [per_user] [no_lock_time] [no_reset] [audit] [silent] [no_log_info]

`pam_tally` [--file */`path/to/counter`*] [--user *`username`*] [--reset[=*`n`*]] [--quiet]

6.32.1. DESCRIPTION

This module maintains a count of attempted accesses, can reset count on success, can deny access if too many attempts fail.

pam_tally has several limitations, which are solved with pam_tally2. For this reason pam_tally is deprecated and will be removed in a future release.

pam_tally comes in two parts: *pam_tally.so* and **pam_tally**. The former is the PAM module and the latter, a stand-alone program. **pam_tally** is an (optional) application which can be used to interrogate and manipulate the counter file. It can display user counts, set individual counts, or clear all counts. Setting artificially high counts may be useful for blocking users without changing their passwords. For example, one might find it useful to clear all counts every midnight from a cron job. The faillog(8) command can be used instead of pam_tally to to maintain the counter file.

Normally, failed attempts to access *root* will *not* cause the root account to become blocked, to prevent denial-of-service: if your users aren't given shell accounts and root may only login via **su** or at the machine console (not telnet/rsh, etc), this is safe.

6.32.2. OPTIONS

GLOBAL OPTIONS

This can be used for *auth* and *account* module types.

`onerr=[`*`fail`*`|`*`succeed`*`]`
: If something weird happens (like unable to open the file), return with PAM_SUCCESS if `onerr=`*`succeed`* is given, else with the corresponding PAM error code.

`file=`*`/path/to/counter`*
: File where to keep counts. Default is `/var/log/faillog`.

`audit`
: Will log the user name into the system log if the user is not found.

`silent`
: Don't print informative messages.

`no_log_info`
: Don't log informative messages via syslog(3).

AUTH OPTIONS

Authentication phase first checks if user should be denied access and if not it increments attempted login counter. Then on call to pam_setcred(3) it resets the attempts counter.

`deny=`*`n`*
: Deny access if tally for this user exceeds *n*.

`lock_time=`*`n`*
: Always deny for *n* seconds after failed attempt.

`unlock_time=`*`n`*
: Allow access after *n* seconds after failed attempt. If this option is used the user will be locked out for the specified amount of time after he exceeded his maximum allowed attempts. Otherwise the account is locked until the lock is removed by a manual intervention of the system administrator.

`magic_root`
: If the module is invoked by a user with uid=0 the counter is not incremented. The sysadmin should use this for user launched

services, like **su**, otherwise this argument should be omitted.

`no_lock_time` Do not use the .fail_locktime field in `/var/log/faillog` for this user.

`no_reset` Don't reset count on successful entry, only decrement.

`even_deny_root_account` Root account can become unavailable.

`per_user` If `/var/log/faillog` contains a non-zero fail_max/.fail_locktime field for this user then use it instead of `deny=`*n*/`lock_time=`*n* parameter.

`no_lock_time` Don't use .fail_locktime filed in `/var/log/faillog` for this user.

ACCOUNT OPTIONS Account phase resets attempts counter if the user is *not* magic root. This phase can be used optionally for services which don't call pam_setcred(3) correctly or if the reset should be done regardless of the failure of the account phase of other modules.

`magic_root` If the module is invoked by a user with uid=0 the counter is not incremented. The sysadmin should use this for user launched services, like **su**, otherwise this argument should be omitted.

`no_reset` Don't reset count on successful entry, only decrement.

6.32.3. MODULE TYPES PROVIDED

The `auth` and `account` module types are provided.

6.32.4. RETURN VALUES

PAM_AUTH_ERR A invalid option was given, the module was not able to retrieve the user name, no valid counter file was found, or too many failed logins.

PAM_SUCCESS Everything was successful.

PAM_USER_UNKNOWNUser not known.

6.32.5. EXAMPLES

Add the following line to `/etc/pam.d/login` to lock the account after too many failed logins. The number of allowed fails is specified by `/var/log/faillog` and needs to be set with pam_tally or faillog(8) before.

```
auth     required       pam_securetty.so
auth     required       pam_tally.so per_user
auth     required       pam_env.so
auth     required       pam_unix.so
auth     required       pam_nologin.so
account  required       pam_unix.so
password required       pam_unix.so
session  required       pam_limits.so
session  required       pam_unix.so
session  required       pam_lastlog.so nowtmp
session  optional       pam_mail.so standard
```

6.32.6. AUTHOR

pam_tally was written by Tim Baverstock and Tomas Mraz.

6.33. pam_tally2 - login counter (tallying) module

`pam_tally2.so` [file=*`/path/to/counter`*] [onerr=[*`fail`*|*`succeed`*]] [magic_root] [even_deny_root] [deny=*`n`*] [lock_time=*`n`*] [unlock_time=*`n`*] [root_unlock_time=*`n`*] [serialize] [audit] [silent] [no_log_info] [debug]

`pam_tally2` [--file */`path/to/counter`*] [--user *`username`*] [--reset[=*`n`*]] [--quiet]

6.33.1. DESCRIPTION

This module maintains a count of attempted accesses, can reset count on success, can deny access if too many attempts fail.

pam_tally2 comes in two parts: *pam_tally2.so* and **pam_tally2**. The former is the PAM module and the latter, a stand-alone program. **pam_tally2** is an (optional) application which can be used to interrogate and manipulate the counter file. It can display user counts, set individual counts, or clear all counts. Setting artificially high counts may be useful for blocking users without changing their passwords. For example, one might find it useful to clear all counts every midnight from a cron job.

Normally, failed attempts to access *root* will *not* cause the root account to become blocked, to prevent denial-of-service: if your users aren't given shell accounts and root may only login via **su** or at the machine console (not telnet/rsh, etc), this is safe.

6.33.2. OPTIONS

GLOBAL OPTIONS This can be used for *auth* and *account* module types.

	`onerr=[`*`fail`*`|`*`succeed`*`]`	If something weird happens (like unable to open the file), return with PAM_SUCCESS if `onerr=`*`succeed`* is given, else with the corresponding PAM error code.
	`file=`*`/path/to/counter`*	File where to keep counts. Default is `/var/log/tallylog`.
	`audit`	Will log the user name into the system log if the user is not found.
	`silent`	Don't print informative messages.
	`no_log_info`	Don't log informative messages via syslog(3).
	`debug`	Always log tally count when it is incremented as a debug level message to the system log.
AUTH OPTIONS	Authentication phase first increments attempted login counter and checks if user should be denied access. If the user is authenticated and the login process continues on call to pam_setcred(3) it resets the attempts counter.	
	`deny=`*`n`	Deny access if tally for this user exceeds *n*.
	`lock_time=`*`n`	Always deny for *n* seconds after failed attempt.
	`unlock_time=`*`n`	Allow access after *n* seconds after failed attempt. If this option is used the user will be locked out for the specified amount of time after he exceeded his maximum allowed attempts. Otherwise the account is locked until the lock is removed by a manual intervention of the system administrator.
	`magic_root`	If the module is invoked by a user with uid=0 the counter is not incremented.

The sysadmin should use this for user launched services, like **su**, otherwise this argument should be omitted.

`even_deny_root` Root account can become unavailable.

`root_unlock_time=`*n* This option implies `even_deny_root` option. Allow access after *n* seconds to root account after failed attempt. If this option is used the root user will be locked out for the specified amount of time after he exceeded his maximum allowed attempts.

`serialize` Serialize access to the tally file using locks. This option might be used only for non-multithreaded services because it depends on the fcntl locking of the tally file. Also it is a good idea to use this option only in such configurations where the time between auth phase and account or setcred phase is not dependent on the authenticating client. Otherwise the authenticating client will be able to prevent simultaneous authentications by the same user by simply artificially prolonging the time the file record lock is held.

ACCOUNT OPTIONS

Account phase resets attempts counter if the user is *not* magic root. This phase can be used optionally for services which don't call pam_setcred(3) correctly or if the reset should be done regardless of the failure of the account phase of other modules.

`magic_root` If the module is invoked by a user with uid=0 the counter is not changed. The sysadmin should use this for user launched services, like **su**, otherwise this argument should be omitted.

6.33.3. MODULE TYPES PROVIDED

The `auth` and `account` module types are provided.

6.33.4. RETURN VALUES

PAM_AUTH_ERR	A invalid option was given, the module was not able to retrieve the user name, no valid counter file was found, or too many failed logins.
PAM_SUCCESS	Everything was successful.
PAM_USER_UNKNOWN	User not known.

6.33.5. NOTES

pam_tally2 is not compatible with the old pam_tally faillog file format. This is caused by requirement of compatibility of the tallylog file format between 32bit and 64bit architectures on multiarch systems.

There is no setuid wrapper for access to the data file such as when the *pam_tally2.so* module is called from xscreensaver. As this would make it impossible to share PAM configuration with such services the following workaround is used: If the data file cannot be opened because of insufficient permissions (EACCES) the module returns PAM_IGNORE.

6.33.6. EXAMPLES

Add the following line to `/etc/pam.d/login` to lock the account after 4 failed logins. Root account will be locked as well. The accounts will be automatically unlocked after 20 minutes. The module does not have to be called in the account phase because the **login** calls pam_setcred(3) correctly.

```
auth     required       pam_securetty.so
auth     required       pam_tally2.so deny=4 even_deny_root unlock_time=1200
auth     required       pam_env.so
auth     required       pam_unix.so
auth     required       pam_nologin.so
account  required       pam_unix.so
password required       pam_unix.so
session  required       pam_limits.so
session  required       pam_unix.so
session  required       pam_lastlog.so nowtmp
session  optional       pam_mail.so standard
```

6.33.7. FILES

`/var/log/tallylog`	failure count logging file

6.33.8. AUTHOR

pam_tally2 was written by Tim Baverstock and Tomas Mraz.

6.34. pam_time - time controled access

`pam_time.so` [debug] [noaudit]

6.34.1. DESCRIPTION

The pam_time PAM module does not authenticate the user, but instead it restricts access to a system and or specific applications at various times of the day and on specific days or over various terminal lines. This module can be configured to deny access to (individual) users based on their name, the time of day, the day of week, the service they are applying for and their terminal from which they are making their request.

By default rules for time/port access are taken from config file `/etc/security/time.conf`.

If Linux PAM is compiled with audit support the module will report when it denies access.

6.34.2. DESCRIPTION

The pam_time PAM module does not authenticate the user, but instead it restricts access to a system and or specific applications at various times of the day and on specific days or over various terminal lines. This module can be configured to deny access to (individual) users based on their name, the time of day, the day of week, the service they are applying for and their terminal from which they are making their request.

For this module to function correctly there must be a correctly formatted `/etc/security/time.conf` file present. White spaces are ignored and lines maybe extended with '\' (escaped newlines). Text following a '#' is ignored to the end of the line.

The syntax of the lines is as follows:

services;*ttys*;*users*;*times*

In words, each rule occupies a line, terminated with a newline or the beginning of a comment; a '#'. It contains four fields separated with semicolons, ';'.

The first field, the *services* field, is a logic list of PAM service names that the rule applies to.

The second field, the *tty* field, is a logic list of terminal names that this rule applies to.

The third field, the *users* field, is a logic list of users or a netgroup of users to whom this rule applies.

For these items the simple wildcard '*' may be used only once. With netgroups no wildcards or logic operators are allowed.

The *times* field is used to indicate the times at which this rule applies. The format here is a logic list of day/time-range entries. The days are specified by a sequence of two character entries, MoTuSa for example is Monday Tuesday and Saturday. Note that repeated days are unset MoMo = no day, and MoWk = all weekdays bar Monday. The two character combinations accepted are Mo Tu We Th Fr Sa Su Wk Wd Al, the last two being week-end days and all 7 days of the week respectively. As a final example, AlFr means all days except Friday.

Each day/time-range can be prefixed with a '!' to indicate "anything but". The time-range part is two 24-hour times HHMM, separated by a hyphen, indicating the start and finish time (if the finish time is smaller than the start time it is deemed to apply on the following day).

For a rule to be active, ALL of service+ttys+users must be satisfied by the applying process.

Note, currently there is no daemon enforcing the end of a session. This needs to be remedied.

Poorly formatted rules are logged as errors using syslog(3).

6.34.3. OPTIONS

`debug`	Some debug information is printed with syslog(3).
`noaudit`	Do not report logins at disallowed time to the audit subsystem.

6.34.4. MODULE TYPES PROVIDED

Only the `account` type is provided.

6.34.5. RETURN VALUES

PAM_SUCCESS	Access was granted.
PAM_ABORT	Not all relevant data could be gotten.
PAM_BUF_ERR	Memory buffer error.
PAM_PERM_DENIED	Access was not granted.
PAM_USER_UNKNOWN	The user is not known to the system.

6.34.6. FILES

`/etc/security/time.conf`	Default configuration file

6.34.7. EXAMPLES

These are some example lines which might be specified in `/etc/security/time.conf`.

All users except for *root* are denied access to console-login at all times:

```
login ; tty* & !ttyp* ; !root ; !Al0000-2400
```

Games (configured to use PAM) are only to be accessed out of working hours. This rule does not apply to the user *waster*:

```
games ; * ; !waster ; Wd0000-2400 | Wk1800-0800
```

6.34.8. AUTHOR

pam_time was written by Andrew G. Morgan <morgan@kernel.org>.

6.35. pam_timestamp - authenticate using cached successful authentication attempts

`pam_timestamp.so` [timestampdir=*`directory`*] [timestamp_timeout=*`number`*] [verbose] [debug]

6.35.1. DESCRIPTION

In a nutshell, *pam_timestamp* caches successful authentication attempts, and allows you to use a recent successful attempt as the basis for authentication. This is similar mechanism which is used in **sudo**.

When an application opens a session using *pam_timestamp*, a timestamp file is created in the *timestampdir* directory for the user. When an application attempts to authenticate the user, a *pam_timestamp* will treat a sufficiently recent timestamp file as grounds for succeeding.

6.35.2. OPTIONS

`timestampdir=`*`directory`*	Specify an alternate directory where *pam_timestamp* creates timestamp files.
`timestamp_timeout=`*`number`*	How long should *pam_timestamp* treat timestamp as valid after their last modification date (in seconds). Default is 300 seconds.
`verbose`	Attempt to inform the user when access is granted.
`debug`	Turns on debugging messages sent to syslog(3).

6.35.3. MODULE TYPES PROVIDED

The `auth` and `session` module types are provided.

6.35.4. RETURN VALUES

PAM_AUTH_ERR	The module was not able to retrieve the user name or no valid timestamp file was found.
PAM_SUCCESS	Everything was successful.
PAM_SESSION_ERR	Timestamp file could not be created or updated.

6.35.5. NOTES

Users can get confused when they are not always asked for passwords when running a given program. Some users reflexively begin typing information before noticing that it is not being asked for.

6.35.6. EXAMPLES

```
auth sufficient pam_timestamp.so verbose
auth required   pam_unix.so

session required pam_unix.so
session optional pam_timestamp.so
```

6.35.7. FILES

`/var/run/pam_timestamp/...`	timestamp files and directories

6.35.8. AUTHOR

pam_timestamp was written by Nalin Dahyabhai.

6.36. pam_umask - set the file mode creation mask

`pam_umask.so` [debug] [silent] [usergroups] [nousergroups] [umask=*mask*]

6.36.1. DESCRIPTION

pam_umask is a PAM module to set the file mode creation mask of the current environment. The umask affects the default permissions assigned to newly created files.

The PAM module tries to get the umask value from the following places in the following order:

- umask= entry in the user's GECOS field
- umask= argument
- UMASK entry from /etc/login.defs
- UMASK= entry from /etc/default/login

The GECOS field is split on comma ',' characters. The module also in addition to the umask= entry recognizes pri= entry, which sets the nice priority value for the session, and ulimit= entry, which sets the maximum size of files the processes in the session can create.

6.36.2. OPTIONS

`debug`	Print debug information.
`silent`	Don't print informative messages.
`usergroups`	If the user is not root and the username is the same as primary group name, the umask group bits are set to be the same as owner bits (examples: 022 -> 002, 077 -> 007).
`nousergroups`	This is the direct opposite of the usergroups option described above, which can be useful in case pam_umask has been compiled with usergroups enabled by default and you want to disable it at runtime.
`umask=`*`mask`*	Sets the calling process's file mode creation mask (umask) to `mask` & 0777. The value is interpreted as Octal.

6.36.3. MODULE TYPES PROVIDED

Only the `session` type is provided.

6.36.4. RETURN VALUES

PAM_SUCCESS	The new umask was set successfully.

PAM_SERVICE_ERR No username was given.

PAM_USER_UNKNOWN User not known.

6.36.5. EXAMPLES

Add the following line to `/etc/pam.d/login` to set the user specific umask at login:

```
session optional pam_umask.so umask=0022
```

6.36.6. AUTHOR

pam_umask was written by Thorsten Kukuk <kukuk@thkukuk.de>.

6.37. pam_unix - traditional password authentication

`pam_unix.so` [...]

6.37.1. DESCRIPTION

This is the standard Unix authentication module. It uses standard calls from the system's libraries to retrieve and set account information as well as authentication. Usually this is obtained from the /etc/passwd and the /etc/shadow file as well if shadow is enabled.

The account component performs the task of establishing the status of the user's account and password based on the following *shadow* elements: expire, last_change, max_change, min_change, warn_change. 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 shadow(5) manual page. Should the user's record not contain one or more of these entries, the corresponding *shadow* check is not performed.

The authentication component performs the task of checking the users credentials (password). The default action of this module is to not permit the user access to a service if their official password is blank.

A helper binary, unix_chkpwd(8), is provided to check the user's password when it is stored in a read protected database. This binary is very simple and will only check the password of the user invoking it. It is called transparently on behalf of the user by the authenticating component of this module. In this way it is possible for applications like xlock(1) to work without being setuid-root. The module, by default, will temporarily turn off SIGCHLD handling for the duration of execution of the helper binary. This is generally the right thing to do, as many applications are not prepared to handle this signal from a child they didn't know was `fork()`d. The `noreap` module argument can be used to suppress this temporary shielding and may be needed for use with certain applications.

The maximum length of a password supported by the pam_unix module via the helper binary is *PAM_MAX_RESP_SIZE* - currently 512 bytes. The rest of the password provided by the conversation function to the module will be ignored.

The password component of this module performs the task of updating the user's password. The default encryption hash is taken from the *ENCRYPT_METHOD* variable from */etc/login.defs*

The session component of this module logs when a user logins or leave the system.

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

6.37.2. OPTIONS

`debug`	Turns on debugging via syslog(3).
`audit`	A little more extreme than debug.
`quiet`	Turns off informational messages namely messages about session open and close via syslog(3).
`nullok`	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.
`try_first_pass`	Before prompting the user for their password, the module first tries the previous stacked module's password in case that satisfies this module as well.
`use_first_pass`	The argument `use_first_pass` forces the module to use a previous stacked modules password and will never prompt the user - if no password is available or the password is not appropriate, the user will be denied access.
`nodelay`	This argument can be used to discourage the authentication component from requesting a delay should the authentication as a whole fail. The default action is for the module to request a delay-on-failure of the order of two second.
`use_authtok`	When password changing enforce the module to set the new password to the one provided by a previously stacked `password` module (this is used in the example of the stacking of the **pam_cracklib** module documented below).
`authtok_type=`*`type`*	This argument can be used to modify the password prompt when changing passwords to include the type of the password. Empty by default.
`nis`	NIS RPC is used for setting new passwords.
`remember=`*`n`*	The last *`n`* passwords for each user are saved in `/etc/security/opasswd` in order to force password change history and keep the user from alternating between the same password too frequently. The MD5 password hash algorithm is used for storing the old passwords. Instead of this option the **pam_pwhistory** module should be used.
`shadow`	Try to maintain a shadow based system.
`md5`	When a user changes their password next, encrypt it with the MD5 algorithm.
`bigcrypt`	When a user changes their password next, encrypt it with the DEC C2 algorithm.

`sha256`	When a user changes their password next, encrypt it with the SHA256 algorithm. The SHA256 algorithm must be supported by the crypt(3) function.
`sha512`	When a user changes their password next, encrypt it with the SHA512 algorithm. The SHA512 algorithm must be supported by the crypt(3) function.
`blowfish`	When a user changes their password next, encrypt it with the blowfish algorithm. The blowfish algorithm must be supported by the crypt(3) function.
`gost_yescrypt`	When a user changes their password next, encrypt it with the gost-yescrypt algorithm. The gost-yescrypt algorithm must be supported by the crypt(3) function.
`yescrypt`	When a user changes their password next, encrypt it with the yescrypt algorithm. The yescrypt algorithm must be supported by the crypt(3) function.
`rounds=`*n*	Set the optional number of rounds of the SHA256, SHA512, blowfish, gost-yescrypt, and yescrypt password hashing algorithms to *n*.
`broken_shadow`	Ignore errors reading shadow information for users in the account management module.
`minlen=`*n*	Set a minimum password length of *n* characters. The max. for DES crypt based passwords are 8 characters.
`no_pass_expiry`	When set ignore password expiration as defined by the *shadow* entry of the user. The option has an effect only in case *pam_unix* was not used for the authentication or it returned authentication failure meaning that other authentication source or method succeeded. The example can be public key authentication in *sshd*. The module will return *PAM_SUCCESS* instead of eventual *PAM_NEW_AUTHTOK_REQD* or *PAM_AUTHTOK_EXPIRED*.

Invalid arguments are logged with syslog(3).

6.37.3. MODULE TYPES PROVIDED

All module types (`account`, `auth`, `password` and `session`) are provided.

6.37.4. RETURN VALUES

PAM_IGNORE Ignore this module.

6.37.5. EXAMPLES

An example usage for `/etc/pam.d/login` would be:

```
# Authenticate the user
auth       required   pam_unix.so
# Ensure users account and password are still active
account    required   pam_unix.so
# Change the user's password, but at first check the strength
# with pam_cracklib(8)
password   required   pam_cracklib.so retry=3 minlen=6 difok=3
password   required   pam_unix.so use_authtok nullok md5
session    required   pam_unix.so
```

6.37.6. AUTHOR

pam_unix was written by various people.

6.38. pam_userdb - authenticate against a db database

`pam_userdb.so` db=*`/path/database`* [debug] [crypt=[crypt|none]] [icase] [dump] [try_first_pass] [use_first_pass] [unknown_ok] [key_only]

6.38.1. DESCRIPTION

The pam_userdb module is used to verify a username/password pair against values stored in a Berkeley DB database. The database is indexed by the username, and the data fields corresponding to the username keys are the passwords.

6.38.2. OPTIONS

`crypt=[crypt|none]`
: Indicates whether encrypted or plaintext passwords are stored in the database. If it is `crypt`, passwords should be stored in the database in crypt(3) form. If `none` is selected, passwords should be stored in the database as plaintext.

`db=`*`/path/database`*
: Use the `/path/database` database for performing lookup. There is no default; the module will return *PAM_IGNORE* if no database is provided. Note that the path to the database file should be specified without the `.db` suffix.

`debug`
: Print debug information.

`dump`
: Dump all the entries in the database to the log. Don't do this by default!

`icase`
: Make the password verification to be case insensitive (ie when working with registration numbers and such). Only works with plaintext password storage.

`try_first_pass`
: Use the authentication token previously obtained by another module that did the conversation with the application. If this token can not be obtained then the module will try to converse. This option

	can be used for stacking different modules that need to deal with the authentication tokens.
`use_first_pass`	Use the authentication token previously obtained by another module that did the conversation with the application. If this token can not be obtained then the module will fail. This option can be used for stacking different modules that need to deal with the authentication tokens.
`unknown_ok`	Do not return error when checking for a user that is not in the database. This can be used to stack more than one pam_userdb module that will check a username/password pair in more than a database.
`key_only`	The username and password are concatenated together in the database hash as 'username-password' with a random value. if the concatenation of the username and password with a dash in the middle returns any result, the user is valid. this is useful in cases where the username may not be unique but the username and password pair are.

6.38.3. MODULE TYPES PROVIDED

The `auth` and `account` module types are provided.

6.38.4. RETURN VALUES

PAM_AUTH_ERR	Authentication failure.
PAM_AUTHTOK_RECOVERY_ERR	Authentication information cannot be recovered.
PAM_BUF_ERR	Memory buffer error.
PAM_CONV_ERR	Conversation failure.
PAM_SERVICE_ERR	Error in service module.
PAM_SUCCESS	Success.
PAM_USER_UNKNOWN	User not known to the underlying authentication module.

6.38.5. EXAMPLES

```
auth  sufficient pam_userdb.so icase db=/etc/dbtest
```

6.38.6. AUTHOR

pam_userdb was written by Cristian Gafton >gafton@redhat.com<.

6.39. pam_warn - logs all PAM items

`pam_warn.so`

6.39.1. DESCRIPTION

pam_warn is a PAM module that logs 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. The module always returns *PAM_IGNORE*, indicating that it does not want to affect the authentication process.

6.39.2. OPTIONS

This module does not recognise any options.

6.39.3. MODULE TYPES PROVIDED

The `auth`, `account`, `password` and `session` module types are provided.

6.39.4. RETURN VALUES

PAM_IGNORE This module always returns PAM_IGNORE.

6.39.5. EXAMPLES

```
#%PAM-1.0
#
# If we don't have config entries for a service, the
# OTHER entries are used. To be secure, warn and deny
# access to everything.
other auth     required       pam_warn.so
other auth     required       pam_deny.so
other account  required       pam_warn.so
other account  required       pam_deny.so
other password required       pam_warn.so
other password required       pam_deny.so
other session  required       pam_warn.so
other session  required       pam_deny.so
```

6.39.6. AUTHOR

pam_warn was written by Andrew G. Morgan <morgan@kernel.org>.

6.40. pam_wheel - only permit root access to members of group wheel

`pam_wheel.so` [debug] [deny] [group=*`name`*] [root_only] [trust] [use_uid]

6.40.1. DESCRIPTION

The pam_wheel PAM module is used to enforce the so-called *wheel* group. By default it permits access to the target user if the applicant user is a member of the *wheel* group. If no group with this name exist, the module is using the group with the group-ID *0*.

6.40.2. OPTIONS

`debug`	Print debug information.
`deny`	Reverse the sense of the auth operation: if the user is trying to get UID 0 access and is a member of the wheel group (or the group of the `group` option), deny access. Conversely, if the user is not in the group, return PAM_IGNORE (unless `trust` was also specified, in which case we return PAM_SUCCESS).
`group=`*`name`*	Instead of checking the wheel or GID 0 groups, use the *`name`* group to perform the authentication.
`root_only`	The check for wheel membership is done only when the target user UID is 0.
`trust`	The pam_wheel module will return PAM_SUCCESS instead of PAM_IGNORE if the user is a member of the wheel group (thus with a little play stacking the modules the wheel members may be able to su to root without being prompted for a passwd).
`use_uid`	The check for wheel membership will be done against the current uid instead of the original one (useful when jumping with su from one account to another for example).

6.40.3. MODULE TYPES PROVIDED

The *auth* and *account* module types are provided.

6.40.4. RETURN VALUES

PAM_AUTH_ERR	Authentication failure.
PAM_BUF_ERR	Memory buffer error.
PAM_IGNORE	The return value should be ignored by PAM dispatch.
PAM_PERM_DENY	Permission denied.
PAM_SERVICE_ERR	Cannot determine the user name.
PAM_SUCCESS	Success.
PAM_USER_UNKNOWN	User not known.

6.40.5. EXAMPLES

The root account gains access by default (rootok), only wheel members can become root (wheel) but Unix authenticate non-root applicants.

```
su      auth     sufficient     pam_rootok.so
su      auth     required       pam_wheel.so
su      auth     required       pam_unix.so
```

6.40.6. AUTHOR

pam_wheel was written by Cristian Gafton <gafton@redhat.com>.

6.41. pam_xauth - forward xauth keys between users

`pam_xauth.so` [debug] [xauthpath=*`/path/to/xauth`*] [systemuser=*`UID`*] [targetuser=*`UID`*]

6.41.1. DESCRIPTION

The pam_xauth PAM module is designed to forward xauth keys (sometimes referred to as "cookies") between users.

Without pam_xauth, when xauth is enabled and a user uses the su(1) command to assume another user's privileges, that user is no longer able to access the original user's X display because the new user does not have the key needed to access the display. pam_xauth solves the problem by forwarding the key from the user running su (the source user) to the user whose identity the source user is assuming (the target user) when the session is created, and destroying the key when the session is torn down.

This means, for example, that when you run su(1) from an xterm session, you will be able to run X programs without explicitly dealing with the xauth(1) xauth command or ~/.Xauthority files.

pam_xauth will only forward keys if xauth can list a key connected to the $DISPLAY environment variable.

Primitive access control is provided by `~/.xauth/export` in the invoking user's home directory and `~/.xauth/import` in the target user's home directory.

If a user has a `~/.xauth/import` file, the user will only receive cookies from users listed in the file. If there is no `~/.xauth/import` file, the user will accept cookies from any other user.

If a user has a `.xauth/export` file, the user will only forward cookies to users listed in the file. If there is no `~/.xauth/export` file, and the invoking user is not *root*, the user will forward cookies to any other user. If there is no `~/.xauth/export` file, and the invoking user is *root*, the user will *not* forward cookies to other users.

Both the import and export files support wildcards (such as *). Both the import and export files can be empty, signifying that no users are allowed.

6.41.2. OPTIONS

`debug`	Print debug information.
`xauthpath=`*`/path/to/xauth`*	Specify the path the xauth program (it is expected in `/usr/X11R6/bin/xauth`, `/usr/bin/xauth`, or `/usr/bin/X11/xauth` by default).
`systemuser=`*`UID`*	Specify the highest UID which will be assumed to belong to a "system" user. pam_xauth will refuse to forward credentials to users

with UID less than or equal to this number, except for root and the "targetuser", if specified.

`targetuser=`*`UID`* Specify a single target UID which is exempt from the systemuser check.

6.41.3. MODULE TYPES PROVIDED

Only the *session* type is provided.

6.41.4. RETURN VALUES

PAM_BUF_ERR	Memory buffer error.
PAM_PERM_DENIED	Permission denied by import/export file.
PAM_SESSION_ERR	Cannot determine user name, UID or access users home directory.
PAM_SUCCESS	Success.
PAM_USER_UNKNOWN	User not known.

6.41.5. EXAMPLES

Add the following line to `/etc/pam.d/su` to forward xauth keys between users when calling su:

```
session  optional  pam_xauth.so
```

6.41.6. AUTHOR

pam_xauth was written by Nalin Dahyabhai <nalin@redhat.com>, based on original version by Michael K. Johnson <johnsonm@redhat.com>.

Chapter 7. 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.

Chapter 8. Author/acknowledgments

This document was written by Andrew G. Morgan (morgan@kernel.org) with many contributions from Chris Adams, Peter Allgeyer, Tim Baverstock, Tim Berger, Craig S. Bell, Derrick J. Brashear, Ben Buxton, Seth Chaiklin, Oliver Crow, Chris Dent, Marc Ewing, Cristian Gafton, Emmanuel Galanos, Brad M. Garcia, Eric Hester, Michel D'Hooge, Roger Hu, Eric Jacksch, Michael K. Johnson, David Kinchlea, Olaf Kirch, Marcin Korzonek, Thorsten Kukuk, Stephen Langasek, Nicolai Langfeldt, Elliot Lee, Luke Kenneth Casson Leighton, Al Longyear, Ingo Luetkebohle, Marek Michalkiewicz, Robert Milkowski, Aleph One, Martin Pool, Sean Reifschneider, Jan Rekorajski, Erik Troan, Theodore Ts'o, Jeff Uphoff, Myles Uyema, Savochkin Andrey Vladimirovich, Ronald Wahl, David Wood, John Wilmes, 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*.

Chapter 9. Copyright information for this document