PAM modules are classified into module type. Any given module should implement at least one of the four module type functions:

1. The **authentication module** is used to authenticate users or set/destroy credentials.
2. The **account management modules** perform actions related to access, account and credential expiration, password restrictions/rules, etc.
3. The **session management module** is used for initializing and terminating sessions.
4. The **password management module** performs actions related to password change/updates.

PAM provides different functional capabilities, such as single sign-on authentication, access control, and more. The implementation of each are handled by different modules. Here are some of the major modules:

* pam\_access delivers log-daemon-style login access control using login/domain names depending on pre-defined rules in /etc/security/access.conf.
* pam\_cracklib checks the passwords against the password rules.
* pam\_env sets/unsets environment variables from /etc/security/pam\_env\_conf.
* pam\_debug debugs PAM.
* pam\_deny locks out PAM modules.
* pam\_echo prints messages.
* pam\_exec executes an external command.
* pam\_ftp is the module for anonymous access.
* pam\_localuser requires the user to be listed in /etc/passwd.
* pam\_unix provides traditional password authentication from /etc/passwd.

There are many other modules (pam\_userdb, pam\_warn, pam\_xauth), which take a set of values which they return. (Details of these modules can be found in the PAM administration guide in [Related topics](https://developer.ibm.com/tutorials/l-pam/#artrelatedtopics).)

Configuring PAM

PAM configuration is generally implemented in the configuration file residing in /etc/pam.d or /etc/pam.conf (for old versions).

The configuration file structure

For each service that uses PAM, there is a corresponding file in the directory, which contains the rules or instructions for how authentication and account information should be obtained for that service. There is usually one rule per line.

Fields in the PAM configuration files include:

* Service\_name specifies the name of the service/application. (The default is OTHER.)
* Module\_type specifies module type (auth/account/session/passwd) for the corresponding service in Service\_name field.
* Control\_flag specifies the stacking behavior of the module. It can take such values as requisite, required, sufficient, and optional.
* Module\_path specifies the path name to the library object which implements the module. It is set to /lib/security by default.
* Module\_options/module\_args (optional fields) specify the options or arguments that can be passed to the services modules.

The modules are invoked in the order in which they are listed in the configuration file, depending on what the Control\_flag for each entry allows. Control\_flag values include:

* Required: All required modules in a stack must pass for a successful result. If one or more of the required module fails, all of the required modules in the stack are implemented, but the first error is returned.
* Sufficient: If a module flagged as sufficient succeeds and no previous required or sufficient modules have failed, then all remaining modules in the stack are ignored and success is returned.
* Optional: If none of the modules in the stack are required and no sufficient modules have succeeded, then at least one optional module of the service/application must succeed.

Examples of PAM config files

Table 1 shows some examples of PAM configuration files on various operating systems.

Table 1. A world of PAM configuration files

| **System** | **Found in …** | **Type** | **Control\_flag** | **Module** |
| --- | --- | --- | --- | --- |
| Red Hat | /etc/pam.d | auth | required | /lib/security/pam\_unix.so |
| Red Hat | /etc/pam.d | account | sufficient | /lib/security/pam\_unix.so |
| Red Hat | /etc/pam.d | session | required | /lib/security/pam\_limit.so |
| AIX | /etc/pam.conf | auth | required | /usr/lib/security/pam\_aix |
| AIX | /etc/pam.conf | account | required | /usr/lib/security/pam\_aix |
| AIX | /etc/pam.conf | password | required | /usr/lib/security/pam\_aix |
| zSUSE 64-bit | 32-bit | /etc/pam.conf | auth | required | /lib64/security/pam\_unix.so | /lib/security/pam\_unix.so |
| zSUSE 64-bit | 32-bit | /etc/pam.conf | account | required | /lib64/security/pam\_unix.so | /lib/security/pam\_unix.so |
| zSUSE 64-bit | 32-bit | /etc/pam.conf | session | required | /lib64/security/pam\_unix.so | /lib/security/pam\_unix.so |
| Solaris | /etc/pam.conf | auth | required | /usr/lib/security/pam\_unix.so.1 |
| Solaris | /etc/pam.conf | account | required | /usr/lib/security/pam\_unix.so.1 |
| Solaris | /etc/pam.conf | password | required | /usr/lib/security/pam\_unix.so.1 |
| HP-UX | /etc/pam.conf | auth | required | libpam\_unix.so.1 |
| HP-UX | /etc/pam.conf | account | required | libpam\_unix.so.1 |
| HP-UX | /etc/pam.conf | password | required | libpam\_unix.so.1 |

The PAM “other” file

The default PAM configuration file /etc/pam.d is used for all other services that are not explicitly configured and is perhaps the simplest and most robust default file upon which PAM relies. The internals look something like this:

/etc/pam.d/other File

auth required pam\_warn.so

auth required pam\_deny.so

account required pam\_warn.so

account required pam\_deny.so

password required pam\_warn.so

password required pam\_deny.so

session required pam\_warn.so

session required pam\_deny.so

Show more

This file is very simple. For all module types, the Control\_flag is the same: required. Two modules are called:

1. First, pam\_warn.so is called to log information about the attempt in progress.
2. Then pam\_deny.so is called to simply return a failure and prevent any kind of connection or authentication from taking place.

Therefore, any service that uses PAM must be explicitly configured to allow authentication; otherwise, attempts will fail.

Ten steps to designing a simple PAM login app

These 10 steps can help you implement your own PAM application and help you understand the workings of a PAM session:

1. Include header files for the PAM implementation (for example, pam\_appl.h, pam\_misc.h).
2. In the main function, initialize the PAM library libpam.so (which loads the modules specified in the configuration file for application) using a unique handle.
3. Attempt authentication for all modules and handle failure scenarios.
4. Check for user credential and account details.
5. Open a new PAM session.
6. Set the environment for the user using credentials.
7. When the user is done, unset the user environment.
8. Close the PAM session.
9. Exit from the library libpam.so with the handle value.
10. EXIT.