**Configure And Use Linux-PAM**

**Table of Contents**[[**hide**](https://likegeeks.com/linux-pam-easy-guide/)]

* [**1 Linux-PAM benefit**](https://likegeeks.com/linux-pam-easy-guide/#Linux-PAM-benefit)
* [**2 Linux-PAM configuration**](https://likegeeks.com/linux-pam-easy-guide/#Linux-PAM-configuration)
* [**3 PAM services**](https://likegeeks.com/linux-pam-easy-guide/#PAM-services)
* [**4 Management groups**](https://likegeeks.com/linux-pam-easy-guide/#Management-groups)
* [**5 Control flags**](https://likegeeks.com/linux-pam-easy-guide/#Control-flags)
* [**6 Modules order**](https://likegeeks.com/linux-pam-easy-guide/#Modules-order)
* [**7 PAM modules**](https://likegeeks.com/linux-pam-easy-guide/#PAM-modules)
* [**8 pam\_succeed\_if module**](https://likegeeks.com/linux-pam-easy-guide/#pam-succeed-if-module)
* [**9 pam\_nologin module**](https://likegeeks.com/linux-pam-easy-guide/#pam-nologin-module)
* [**10 pam\_access module**](https://likegeeks.com/linux-pam-easy-guide/#pam-access-module)
* [**11 pam\_deny module**](https://likegeeks.com/linux-pam-easy-guide/#pam-deny-module)
* [**12 pam\_unix module**](https://likegeeks.com/linux-pam-easy-guide/#pam-unix-module)
* [**13 pam\_localuser module**](https://likegeeks.com/linux-pam-easy-guide/#pam-localuser-module)
* [**14 pam\_mysql module**](https://likegeeks.com/linux-pam-easy-guide/#pam-mysql-module)
* [**15 pam\_cracklib module**](https://likegeeks.com/linux-pam-easy-guide/#pam-cracklib-module)
* [**16 pam\_rootok module**](https://likegeeks.com/linux-pam-easy-guide/#pam-rootok-module)
* [**17 pam\_limits module**](https://likegeeks.com/linux-pam-easy-guide/#pam-limits-module)

**Linux-PAM benefit**

There are many programs on your system that use PAM modules like su, passwd, ssh, login, and other services. We will discuss some of them.

PAM’s main focus is to authenticate your [**users**](https://likegeeks.com/listing-users-in-linux/).

Authentication in Linux is done by matching the encrypted password in /etc/shadow file with the entered one.

We have many services on our systems that require authentication like SSH, [**FTP**](https://likegeeks.com/ftp-server-linux/), TELNET, [**IMAP,**](https://likegeeks.com/linux-mail-server/#POP3-and-IMAP-Protocol-Basics)and many other services. So, we will have a lot of authentication files besides /etc/shadow file to maintain, and it could be a serious problem if there is any inconsistent data between these authentication files.

Here comes PAM. Linux-PAM offers a unified login system for your services.

To check if your program uses Linux-PAM or not:

[root@iscsi security]# ldd `which su`| grep libpam

libpam.so.0 => /lib64/libpam.so.0 (0x00007fbbfd137000)

libpam\_misc.so.0 => /lib64/libpam\_misc.so.0 (0x00007fbbfcf33000)

You should see libpam.so library.

**Linux-PAM configuration**

The configuration of Linux-PAM is in the directory /etc/pam.d/.

Some PAM modules require configuration files with the PAM configuration to operate. You can find the configuration files in /etc/security.

If you misconfigure PAM, this could lead to serious problems.

**PAM services**

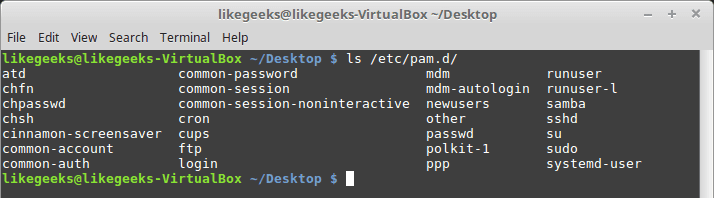
The 四 types of PAM services:

1. Authentication service modules.
2. Account management modules.
3. Session management modules.
4. Password management modules.

Any application requires authentication can register with PAM using a service name.

You can list Linux services that use Linux-PAM.

ls /etc/pam.d/



If you open any service file, you will see that the file is divided into three columns. The first column is the management group, the second column is for control flags, and the third column is the module (so file) used.

$ cat /etc/pam.d/sshd

account    required     pam\_nologin.so

The account is the management group, required is the control flag, and the used module is pam\_nologin.so.

You may find a fourth column, which is for module parameters.

**Management groups**

There are 四Management Groups:

* **auth Group**: it can validate users
* **account Group**: controls the access to the service like how many times you should use this service.
* **session Group**: responsible for the service environment.
* **password Group**: for password updating.

**Control flags**

We have 四control flags:

* **Requisite**: the strongest flag. If the requisite not found or failed to load, it will stop loading other modules and return failure.
* **Required**: The same as requisite, but if the module failed to load for any reason, it continues loading other modules and returns failure at the end of execution.
* **Sufficient**: if the module return success, the processing of other modules no longer needed.
* **Optional**: In the case of failure, the stack of modules continues execution, and the return code is ignored.

**Modules order**

The order is important because each module depends on the previous module on the stack.

If you try a configuration like the following to log in:

auth required pam\_unix.so

auth optional pam\_deny.so

That will work correctly, but what will happen if we change the order like this:

auth optional pam\_deny.so

auth required pam\_unix.so

No one can log in, so the order matters.

**PAM modules**

There are PAM built-in modules on your system that you should know about, so you can use them perfectly.

**pam\_succeed\_if module**

This module allows access for the specified groups. You can validate user accounts like this:

auth required pam\_succeed\_if.so gid=1000,2000

The above line states that only users in the group whose ID 1000 or 2000 are allowed to log in.

You can use uid as user id instead.

auth requisite pam\_succeed\_if.so uid >= 1000

In this example, any user id greater than or equal 1000 can log in.

You can also use it with ingroup parameter like this:

auth required pam\_succeed\_if.so user ingroup mygroup

Only people in the group named mygroup can log in.

**pam\_nologin module**

This module allows root only to log in if the file is available.

/etc/nologin

auth required pam\_nologin.so

You can modify the *login* service file, /etc/pam.d/login, with this line and create /etc/nologin file, so root only can log in.

And you can use it with **auth, account** management groups.

**pam\_access module**

This module works like the pam\_succeed\_if module except the pam\_access module checks logging from networked hosts, while the pam\_succeed\_if module doesn’t care.

account required pam\_access.so accessfile=/etc/security/access.conf

You can type your rules in the file like this:

/etc/security/access.conf

+:mygroup

-:ALL:ALL

The above rules state that only mygroup users are allowed to log in while others can’t.

Where plus sign means allow and minus sign means deny.

This module is used with **auth, account, session, password** management groups.

**pam\_deny module**

You can use this module for restricting access. It will always return a non-OK.

You can use it at the end of your module stack to protect yourself from any misconfiguration.

If you use it at the beginning of the module stack, your service will be disabled:

auth required pam\_deny.so

auth required pam\_unix.so

And you can use it with **auth, account, session, password** management groups.

**pam\_unix module**

You can use this module to check the user’s credentials against /etc/shadow file.

auth required pam\_unix.so

You will see this module used in many services in your system.

And you can use it with **auth, session, password** management groups.

**pam\_localuser module**

You can use this module to check if the user is in /etc/passwd.

account sufficient pam\_localuser.so

And you can use it with **auth, session, password, account** management groups.

**pam\_mysql module**

Instead of checking the user’s credentials against/etc/shadow, you can use a [**MySQL database**](https://likegeeks.com/mysql-on-linux-beginners-tutorial/) as a backend using the pam\_mysql module.

You can use it like this:

auth sufficient pam\_mysql.so user=myuser passwd=mypassword host=localhost db=mydb table=users usercolumn=username passwdcolumn=password

Here we validate the user with the parameters for pam\_mysql.

You can install if it is not on your system like this:

$ yum install libpam-mysql

We use this module with **auth, session, password, account** management groups.

**pam\_cracklib module**

Strong passwords are a must these days. This module ensures that you will use strong passwords.

password required pam\_cracklib.so retry=4 minlen=12 difok=6

This example ensures that:

Password minimum length = 12

Four times to pick a strong password; otherwise, it will exit.

Your new password must have six new characters from the old password. You can use this module with the **password** management group.

**pam\_rootok module**

This module checks if the user ID is 0, which means only root users can run this service.

auth sufficient   pam\_rootok.so

We use this module to ensure that a specific service is allowed for root users only, and you can use it with the **auth** management group.

**pam\_limits module**

You can use this module to set limits on the system resources. It affects even root users.

The limits configuration is in the /etc/security/limits.d/ directory.

session  required  pam\_limits.so

You can use this module to protect your system resources, and you can use it with the **session** management group.

The limits in /etc/security/limits.conf file could be hard or soft.

**Hard**: The user cannot change its value, but root can.

**Soft**: normal user can change it.

The limits could be fsize, cpu, nproc, nproc, data, and many other limits.

@mygroup        hard    nproc           50

myuser          hard    cpu           5000

The first limit for mygroup members, which sets the number of processes for each one of them to be 50.

The second limit for the user named myuser, which limits the CPU time to 5000 minutes.

You can edit any PAM service file in /etc/pam.d/ and use the module you want to protect your services the way you want.