PAM
https://dzone.com/articles/linux-pam-easy-guide

Linux-PAM. PAM (Pluggable Authentication Modules) is the management layer that sits between Linux applications and the Linux native authentication system.

There are many programs in your system that use PAM modules, such as SU, password, SSH, logging in, and other services. We will discuss some of them.

PAM can do many things for you, but the primary focus is to authenticate your users.

Authentication in Linux is done by comparing the encrypted password in /etc/shadow file, but each program that requires authentication implements its own authentication mechanisms.

We have many services on our systems that require authentication, like SSH, FTP, TELNET, IMAP, and many other services, so we will have a lot of authentication files besides /etc/shadow to maintain — and it could be a serious problem if there is any inconsistent data between these authentication files.

Here comes PAM! With Linux-PAM, the system administrator can use the same user database for login to all services if he or she wants to.

You can check if a program uses Linux-PAM or not.

```
$ ldd /bin/su:
```

Linux PAM check pam usability

You should see the libpam.so library.
Linux-PAM Configuration

The configuration of Linux-PAM can be done in two ways. You can either put everything in one single file as /etc/pam.conf or split the configuration by service in the directory with /etc/pam.d/.

The last way is better because it makes it easy to work with each service individually.

Keep in mind that Linux-PAM will ignore /etc/pam.conf if the /etc/pam.d directory exists.

Some PAM modules require configuration files beside the PAM configuration to operate. These module-specific configuration files are stored in /etc/security.

If PAM is wrongly configured, your environment can easily be compromised.
PAM Services

Linux-PAM relies on dynamically loaded modules (SO files). A module can provide mechanisms to authenticate users from any backend like a file /etc/passwd or database.

One of the Linux PAM modules is the PAM service module. The PAM service module is a library that provides authentication and other security services to applications such as login or FTP.

There are four types of PAM services:

- Authentication service modules.
- Account management modules.
- Session management modules
- Password management modules.

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

You can check Linux services that use Linux-PAM.

```
$ ls /etc/pam.d/:
```

Linux PAM services

If you open any service file, you will see that the file is divided into three columns. The first column is 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 module used is pam_nologin.so.

In some lines, you will find a fourth column, which is for module parameters.
Management Groups

There are four Management Groups you will see in PAM services files

- Auth group: Provides two functions. First, the user can be validated (that is, the user provides proof of authenticity). Second, credentials are granted by the auth management group.
- Account group: Controls the access to the service, like using a service for a number of times per week.
- Session group: The environment for a given service is built up by the session management group, and when you stop using a service, the session group puts down the environment.
- Password group: Only used when a user wishes to update the password.

Control Flags

We have four control flags in services files

- Requisite: The strongest flag. If a module is flagged as requisite and it fails, PAM will return to the calling application and report the failure.
- Required: In the case of failure, execution is not stopped but continues to the next module. If, after all of the modules have been executed, one or more has failed, PAM will return failure to the calling application.
- Sufficient: If a sufficient module returns OK, the processing of modules will be stopped.
- Optional: In the case of failure, the stack of modules continues execution and the return code is ignored.

Modules Order

The Linux-PAM modules in the stack are tried one by one.

The order matters because the effect of one module is required for the next module to work correctly.

A configuration like the following for login will work properly:

```
auth required pam_unix.so
auth optional pam_deny.so
```

But if you change the order like this:

```
auth optional pam_deny.so
auth required pam_unix.so
```

Then no one can log in, so the order matters.

Keep in mind that the pam_deny module can be included as the last module in any stack for every service as a failsafe solution.
PAM Modules

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

`Pam_succeed_if Module`

This module can be used to restrict access so that only listed groups can log in. You can validate user accounts like this: **`auth required pam_succeed_if.so gid=1000,2000.`**

If the user is not a member of either of the groups with ID 1000 or 2000, the user will not be 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 to 1000 can log in.

You can also use it with in group 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 only the root to log in if the file /etc/nologin exists.

If /etc/nologin does not exist, valid users can log in: auth required pam_nologin.so.

You can change login service file with this line and create a /etc/nologin file so that only the root can login.

This module used with auth and account management groups.

Pam_access Module

The pam_access module can be used to obtain the same functionality as the pam_succeed_if module, but the pam_access module is focused on logging in from networked hosts, while the pam_succeed_if module has no hint as to where the user is coming from: account required pam_access.so accessfile=/etc/security/access.conf.

The restriction is configured in the /etc/security/access.conf file

You can write the rules in access.conf file like this:

```
+:mygroup
-:ALL:ALL
```

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

Pam_deny Module

The module is able to restrict users from obtaining access to the system. It will always return non-OK.

It always used at the end of the auth stack in order to prevent weaknesses due to misconfigurations.

You can disable a service by adding the pam_deny module at the top of a modules stack like this:

```
auth required pam_deny.so
auth required pam_unix.so
```

We use the module here with auth management group. However, if the module is used in the password management group, it will prevent the user from changing his password.

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

Pam_unix Module

This module is used to validate users against the /etc/shadow file as a backend: auth required pam_unix.so.

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

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

Pam_localuser Module

This module is used to check if the user is listed in /etc/passwd: account sufficient pam_localuser.so.

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

Pam_mysql Module

The pam_mysql module can be used to authenticate users with credentials stored in a database

You can use this module like this

```
auth sufficient pam_mysql.so
USER=myuser passwd=mypassword host=localhost db=mydb
TABLE=users usercolumn=username passwdcolumn=password
```

The parameters for pam_mysql are used to validate the user with the data in the MySQL database.

You can install if it is not on your system: $ yum install libpam-mysql.

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

Pam_cracklib Module

Weak passwords can lead to system compromise. This module ensures that you will use strong **passwords: password required pam_cracklib.so retry=3 minlen=12 difok=3.**

This example ensures that the password should be at least six characters and at least three characters. You have three chances to pick a good password before the password program aborts.

This module is used with the password management group.

Pam_rootok Module

This module checks if the user ID is 0. That means only root users can run this service: auth sufficient pam_rootok.so.

You can use this module to ensure that a specific service is allowed for root users only.

This module is used with auth management group.

Pam_limits Module

This module is used to set limits on the system resources, even root users are affected by these limits.

Limits are taken from the /etc/security/limits.conf file then /etc/security/limits.d/ directory: session required pam_limits.so.

You can use this module to protect your system resources. This module is used with session management group. The limits in /etc/security/limits.conf file could be hard or soft.

Hard limits are set by the superuser and enforced by the kernel. The user cannot change the value.

Soft limits are ones that the user can move up or down within the permitted range by any pre-existing hard limits.

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

```
@mygroup        hard    nproc           50
myuser          hard    cpu           5000
```

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

The second limit for the user named myuser limits the cCPUtime to 5,000 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.