PAM, or Pluggable Authentication Modules, is an abstraction layer that exists on Linux and Unix-like operating systems used to enable authentication between a variety of services.

Built as an intermediary between authentication services and the applications that require user authentication, this system allows these two layers to integrate gracefully and change authentication models without the need to rewrite code. This is accomplished through the use of modules.

Linux’s version of PAM divides module functionality into different categories depending on which part of the process they are involved in. Here is a brief explanation of the categories:

Authentication Functions: The authentication modules validate the user’s authentication credentials. This means it checks if the user can supply valid credentials.

Account Functions: These modules are responsible for deciding if the account that is trying to sign in has access to the resources that it is requesting at this time. PAM allows you to specify controls that can deny or allow users based on predetermined criteria.

Session Functions: These modules establish the environment that will be built up and torn down after the user logs in or logs off. Session files can determine which commands need to be run to prepare the environment.

Password Functions: These modules are responsible for updating various services’ authentication details. If a password needs to be changed for a service, this module can assist in communicating with the service and modifying the correct values.

The first two of these module categories will be referenced every time a program successfully uses PAM for authentication. The session modules will be run if necessary after the first two. The password modules are accessed on-demand.

In terms of directory structure, PAM configuration files are stored at /etc/pam.d

This directory generally has a configuration file for each application that will request PAM authentication. If an application calls PAM but there is no associated configuration file, the “other” configuration file is applied.

Inside the configuration files, there are usually calls to include the configuration files that begin with “common-”. These are general configuration files whose rules should be applied in most situations.

When an application queries the PAM system for authentication, PAM reads the relevant PAM configuration file.

The configuration files contain a list of PAM modules and how they should be handled. Each module is called in turn and each call to a module generates a success or failure result.

Based on these values, the configuration file then decides if it should return an “authentication okay” message to the caller, or send it an authentication failure message.

**[service] type control module-path [module-arguments]**

**[service]** is the application name

**Type** is the kind of service that is provided. It is one of the four categories listed above (authentication, account, password, session).

**Control** specifies the action taken when it receives the return status of the module call. It can be any of these values (and can additionally take a more complex syntax, which won’t be covered here): Here's one example:

required: This will lead to an authentication failure if the module call results in a failure. The remaining specified modules are still called, however.

**Module-path** is the name of the pam module to call.

**Module-arguments** are the optional parameters passed to the module. Sometimes these are necessary for the module to know what action to take if successful.

**sysctl - configure kernel parameters at runtime**

**The -p option allows you to load the settings from a configuration file**

IP spoofing is the creation of Internet Protocol (IP) packets which have a modified source address in order to either hide the identity of the sender, to impersonate another computer system, or both. It is a technique often used by bad actors to invoke DDoS attacks against a target device or the surrounding infrastructure.

Sending and receiving IP packets is a primary way in which networked computers and other devices communicate, and constitutes the basis of the modern internet. All IP packets contain a header which precedes the body of the packet and contains important routing information, including the source address. In a normal packet, the source IP address is the address of the sender of the packet. If the packet has been spoofed, the source address will be forged.

The central file that controls your resolver setup is host.conf. It resides in /etc and tells the resolver which services to use, and in what order.

Options in host.conf must occur on separate lines. Fields may be separated by white space (spaces or tabs). A hash sign (#) introduces a comment that extends to the next newline.

The following options are available:

**order**

This determines the order in which the resolving services are tried. Valid options are :

bind for querying the name server

hosts for lookups in /etc/hosts

The order in which they appear on the line determines the order in which the respective services are tried.

**multi**

Takes on or off as options. This determines if a host in /etc/hosts is allowed to have several IP addresses, which is usually referred to as being ``multi-homed''.

**nospoof**

As explained in the previous chapter, DNS allows you to find the hostname belonging to an IP address. Attempts by name servers to supply a false hostname are called ``spoofing''. To guard against this, the resolver may be configured to check if the original IP address is in fact associated with the hostname obtained. If not, the name is rejected and an error returned. This behavior is turned on by setting nospoof on.

**The ls -l command** to list the contents of the directory in a table format with columns including:

content permissions

number of links to the content

owner of the content

group owner of the content

size of the content in bytes

last modified date / time of the content

file or directory name

u: User, meaning the owner of the file.

g: Group, meaning members of the group the file belongs to.

o: Others, meaning people not governed by the u and g permissions.

a: All, meaning all of the above.

–: Minus sign. Removes the permission.

+: Plus sign. Grants the permission. The permission is added to the existing permissions. If you want to have this permission and only this permission set, use the = option, described below.

=: Equals sign. Set a permission and remove others.

USER is the user name or the user ID (UID) of the new owner. GROUP is the name of the new group or the group ID (GID). FILE(s) is the name of one or more files, directories or links.

USER - If only the user is specified, the specified user will become the owner of the given files, the group ownership is not changed.

USER: - When the username is followed by a colon :, and the group name is not given, the user will become the owner of the files, and the files group ownership is changed to user’s login group.

USER:GROUP - If both the user and the group are specified (with no space betwen them), the user ownership of the files is changed to the given user and the group ownership is changed to the given group.

:GROUP - If the User is omitted and the group is prefixed with a colon :, only the group ownership of the files is changed to the given group.

: If only a colon : is given, without specifying the user and the group, no change is made.

By default, on success, chown doesn’t produce any output and returns zero.

