Autenticación de acceso I: PAM.

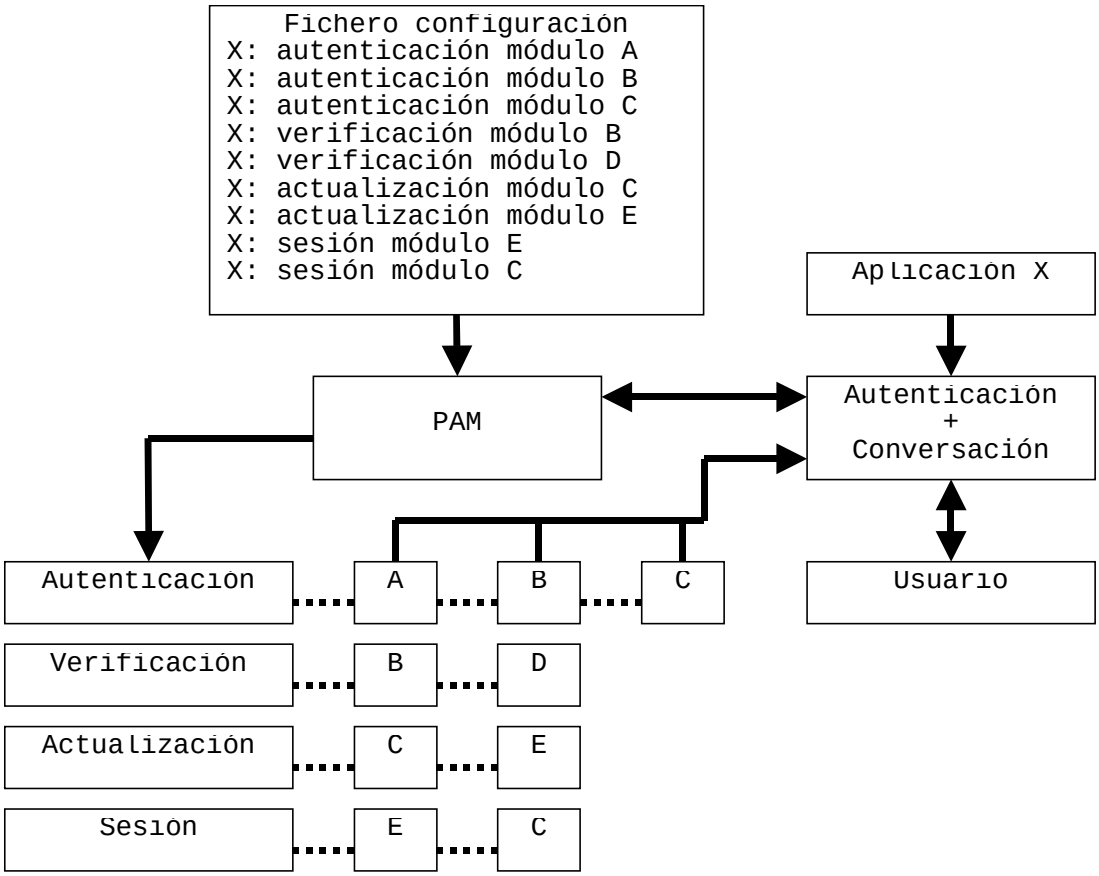
Autor: Enrique V. Bonet Esteban

Introducción.

Pluggable Authentication Modules (PAM) son un conjunto de librerías compartidas que permiten seleccionar como deben identificarse los usuarios ante una determinada aplicación.

El uso de PAM permite, tanto cambiar el funcionamiento de un método de autenticación sin necesidad de cambiar todas las aplicaciones que utilizan ese método, como cambiar el método de autenticación de una aplicación tan solo cambiando las librerías compartidas que utiliza en el proceso de autenticación.

Para poder utilizar PAM una aplicación tan solo necesita conocer la existencia de un conjunto de funciones (Application Programming Interface) que le permiten intercambiar información con PAM, encargándose esté de cargar los módulos necesarios para la autenticación de esa aplicación, etc., y enviar a la aplicación el resultado final de la autenticación. Un breve esquema que explica este funcionamiento se encuentra en la siguiente figura:



En la figura podemos ver como la aplicación X intenta autenticar a un usuario. Para ello, utiliza los módulos que posee de autenticación y conversación para

comunicarse con el gestor de librerías de PAM y con el usuario respectivamente. El gestor de librerías lee el fichero de configuración de la autenticación de la aplicación X y carga los módulos indicados, ejecutando los mismos en función de su pertenencia a cada uno de los grupos de administración (autenticación, verificación, actualización y sesión)[1].

El orden de llamada de cada módulo dentro de un grupo de administración corresponde con el orden en que aparecen en el fichero de configuración. Por ello, en autenticación puede verse que son llamados los módulos A, B y C en ese orden, mientras que en actualización es llamado primero C y luego E y en sesión primero E y luego C, pues C está delante de E en el fichero de configuración para el grupo de actualización y detrás de E en el fichero de configuración para el grupo de sesión.
Configuración de PAM.

La configuración de PAM se puede encontrar en dos lugares diferentes, el fichero */etc/pam.conf* o en los ficheros existentes dentro del directorio */etc/pam.d*. Ambos modos de configuración pueden ser utilizados, de forma que si la configuración de un servicio se encuentra en un fichero del directorio /*etc/pam.d*, el fichero */etc/pam.conf* es ignorado, mientras que si no se encuentra en un fichero del directorio */etc/pam.d*, la configuración existente para ese servicio en el fichero */etc/pam.conf* es utilizada[2].

Dado que la sintaxis de configuración del fichero */etc/pam.conf* y de los ficheros de configuración que pueden existir en el directorio */etc/pam.d* es similar, veremos la sintaxis del fichero */etc/pam.conf* y explicaremos las breves diferencias existentes.

El fichero de configuración */etc/pam.conf* esta formado por líneas, donde toda línea que comienza por el carácter # es considerada un comentario, mientras que toda línea que termina con el carácter \ se considerá que continúa en la siguiente línea del fichero. La sintaxis de cada línea del fichero es la siguiente:

```
servicio   tipo   control   camino al módulo   argumentos del módulo
```

La sintaxis de los ficheros que se encuentran dentro del directorio */etc/pam.d* es similar, excepto que el campo servicio no aparece, pues el nombre que tiene el fichero debe corresponderse con el nombre del servicio, debiendo además estar escrito en minúsculas, pues es sensible al contexto.

Veamos a continuación una descripción de los campos de la sintaxis.

El campo Servicio.

El campos servicio contiene el nombre del servicio asociado a esta línea del fichero[3]. El nombre del servicio es, de forma general, el nombre del servicio,

[1] Como puede verse, un módulo puede pertenecer a más de un grupo y por tanto, puede ser llamado en más de una ocasión dentro del proceso de autenticación.
[2] El uso del fichero */etc/pam.conf* es obsoleto y, por defecto, ya no se crea en la configuración de los sistemas, por lo que si se busca en el directorio */etc* se podrá comprobar que no existe.
[3] Recordar que este campo no existe en el caso de un fichero dentro del directorio /etc/pam.d, pues en este caso el campo servicio debe ser el nombre del fichero.

coincidiendo en la mayoría de ocasiones dicho nombre con el nombre del servidor, como sucede con *vsftpd*, *sshd*, etc.

Existe un nombre especial, *OTHER*[4] que se utiliza para implementar un mecanismo de autenticación por defecto, mecanismo que será utilizado si no existe ninguna entrada de autenticación para el servicio solicitado.

El campo Tipo.

Especifica uno de los cuatro tipos de módulos existentes. Los tipos de módulos existentes son autenticación (*auth*), verificación (*account*), actualización (*password*) y sesión (*session*).

- El tipo *auth* indica que el módulo se encarga de autenticar al usuario mediante la solicitud de una contraseña de acceso o sistema equivalente (tarjeta de acceso, control biométrico, etc.) y después se encarga de proporcionar al usuario los privilegios que la respuesta correcta a la solicitud de autenticación le proporciona, como pueden ser el grupo al que pertenece el usuario, etc .

- El tipo *account* indica que el módulo debe comprobar si el usuario posee permisos para utilizar el servicio solicitado o si el servicio debe facilitarse en ciertas condiciones, como pueden ser su limitación a determinadas franjas horarias, el número de usuarios existentes, la terminal desde la que se intenta el acceso, etc.

- El tipo *password* indica que el módulo es requerido para actualizar el mecanismo de autenticación del usuario, generalmente una contraseña caducada.

- El tipo *session* indica que el módulo esta asociado a ciertas acciones que deben ser realizadas antes y/o después de acceder al servicio. Estas acciones pueden ser la apertura/cierre de ficheros, el montaje de directorios, etc.

El campo Control.

El campo control es usado para indicar a la librería de PAM como debe actuar en caso de que suceda o falle el módulo asociado al servicio y tipo, pudiendo actuar desde ignorar el fallo de ese módulo hasta terminar la ejecución de la librería PAM y devolver el resultado a la aplicación que la ha llamado[5].

Existen dos tipos de sintaxis para el campo control, una es la sintaxis simple o histórica y otra es la sintaxis nueva[6].

[4] La entrada OTHER es independiente del contexto, por lo que puede escribirse en mayúsculas, minúsculas o mediante cualquier combinación de las mismas.
[5] Un aspecto a resaltar es que el programa que llama a la librería de PAM recibe una respuesta sumaria del tipo correcto o incorrecto, sin que sepa exactamente que ha sucedido y porque se ha le ha dado esa respuesta
[6] De forma general, continúa usándose la sintaxis simple o histórica.

La sintaxis simple, se define mediante una palabra que indica la severidad asociada con el que suceda o no la acción de un módulo. La sintaxis simple posee cuatro palabras clave, *required*, *requisite*, *sufficient* y *optional*.

- *requisite* indica que un fallo en este módulo es inmediatamente enviado a la aplicación que llamo a PAM sin ejecutar ninguno de los siguientes módulos de este tipo que existan. Esto puede servir, por ejemplo, para evitar que un usuario introduzca su contraseña en un entorno inseguro, pues no continua la ejecución de ningún otro modulo.

- *required* es similar a la anterior, solo que un fallo en el mismo no implica un envío inmediato de la respuesta a la aplicación que llamó a PAM, sino que previamente se ejecutarán el resto de módulos de este tipo existentes, aunque al final la respuesta devuelta será, en cualquier caso, de fallo.

- *sufficient* indica que la ejecución correcta de este módulo es bastante para satisfacer todos los requisitos de autorización de este tipo de módulos y su respuesta es enviada inmediatamente a la aplicación que llamo a PAM. Es necesario resaltar que si un módulo anterior de tipo *required* ha fallado, dicho fallo no es modificado por este módulo, por lo que el fallo persistirá como respuesta. Si un módulo *sufficient* falla no sucede nada y se continúan ejecutando los módulos siguientes.

- *optional* indica que la respuesta de este módulo solo se tendrá en cuenta si no existe ningún otro módulo de este servicio y tipo.

- *include* indica que se incluyan los módulos del tipo indicado que se encuentran en el fichero cuyo nombre se especifica.

- *substack* indica que se incluyan los módulos del tipo indicado que se encuentran en el fichero cuyo nombre se especifica. La diferencia entre *include* y *substack* es sencilla si se expresa en términos de programación. La inclusión mediante *include* es equivalente a copiar el código en ese lugar, mientras que la inclusión mediante *substack* es equivalente a llamar desde ese punto a una función cuyo código fuera el incluido mediante *substack*, y que devolverá la respuesta al lugar desde donde se llamó, lo cual permite que pueda ser ignorado si así se desea.

La sintaxis nueva es mucho más elaborada y proporciona un mayor control sobre como es identificado un usuario, siendo definida como un conjunto de acciones a ejecutar según el valor devuelto por el módulo llamado. Su sintaxis es:

```
[valor1=accion1 valor2=accion2 ...]
```

Donde *valor1*, *valor2*, etc., son los valores de retorno que pueden devolver los módulos y cuyos valores que pueden verse en la tabla siguiente:

`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`
`default`		

Utilizándose la acción *default* para indicar la acción para aquellos valores de retorno que no están explícitamente definidos.

Por su parte *accion1*, *accion2*, etc., debe ser un entero positivo o uno de los valores: *ignore*, *ok*, *done*, *bad*, *die* o *reset*. Un entero positivo N indica que los siguientes N módulos del mismo tipo sean ignorados y por tanto no ejecutados, mientras que los valores tienen los efectos indicados en la siguiente tabla:

Valor	Descripción
ignore	El valor de retorno de este módulo es ignorado en el calculo del valor de retorno de PAM
ok	Si este módulo es correcto, PAM devolverá correcto, pero antes ejecutará el resto de módulos de este tipo.
done	Si el módulo es correcto PAM devolverá correcto inmediatamente.
bad	Si este módulo falla, PAM devolverá fallo, pero antes ejecutará el resto de módulos de este tipo.
die	Si el módulo falla PAM devolverá fallo inmediatamente.
reset	Elimina el estado de la respuesta actual y comienza su calculo a partir del siguiente módulo.

Las palabras clave de la sintaxis simple pueden expresarse en función de la sintaxis nueva tal y como aparece en la siguiente tabla:

Sintaxis simple	Sintaxis nueva
`requisite`	`[success=ok new_authtok_reqd=ok ignore=ignore default=die]`
`required`	`[success=ok new_authtok_reqd=ok ignore=ignore default=bad]`
`sufficient`	`[success=done new_authtok_reqd=done default=ignore]`
`optional`	`[success=ok new_authtok_reqd=ok default=ignore]`

El campo Camino al módulo.

Este campo indica sencillamente la localización del módulo PAM que debe ser ejecutado. Si el primer carácter que aparece es / se supone que se está indicando el camino completo al módulo. En caso contrario, el camino por defecto a las librerías de PAM (*/lib/security* o */lib64/security* según la arquitectura del sistema sea de 32 o 64 bits) es antepuesto a la localización especificada.

El campo Argumentos del módulo.

Este campo es una lista de palabras que son pasadas al módulo al ser llamado. De forma general, los argumentos de los módulos son específicos de cada módulo, sucediendo que si se pasa a un módulo un argumento invalido, este es ignorado[7]. Si un argumento requiere espacios entre sus componentes, el argumento debe ser encerrado entre los caracteres *[* y *]*.

Independientemente de los argumentos que cada módulo pueda poseer por si mismo, existen una serie de argumentos generales que son:

Argumento	Descripción
debug	Escribe en el log del sistema información para la depuración.
no_warn	No enviar mensajes de aviso.
use_first_pass	Utilizar el password obtenido previamente en lugar de solicitarlo otra vez. Si no se ha solicitado el password previamente el usuario no es autenticado.
try_first_pass	Utilizar el password obtenido previamente si este existe, o solicitarlo en caso contrario.
use_mapped_pass	No usado actualmente por ningún modulo.
expose_account	Minimizar la información enviada para solicitar la autenticación.

Módulos de la librería de PAM.

La librería de PAM esta formada por un gran número de módulos que permiten verificar y ejecutar las acciones necesarias para autenticar y permitir el acceso a un usuario en un gran número de servicios. Los módulos se encuentran situados, como hemos comentado, en el directorio */lib/security* o */lib64/security* y en los subdirectorios que puedan existir dentro de estos directorios.

Veamos sucintamente los principales módulos existentes.

Módulo pam_access.

Es un módulo de tipo *account* que controla si un usuario o grupos de usuarios pueden acceder desde un determinado terminal, conexión remota, etc. Por defecto utiliza el fichero de configuración situado en */etc/security/access.conf*, en el cual se le indica los accesos permitidos y/o restringidos. Sus principales argumentos se encuentran en la siguiente tabla:

Argumento	Descripción
access_file=	Permite indicar un fichero alternativo a */etc/security/access.conf* como fichero de configuración a utilizar.
fieldsep=	Modifica el separador por defecto (:) de los campos del fichero de configuración por el carácter indicado.

[7] Generalmente, además de ignorar el argumento inválido, suele escribirse información sobre el argumento invalido pasado en el log del sistema.

listsep=	Modifica el separador por defecto de los elementos de una lista (carácter de espacio y tabulador) por el carácter indicado.

Un ejemplo de su uso es el siguiente:

```
account  required       /lib/security/pam_access.so
```

Módulo pam_chroot.

Es un módulo de tipo *auth*, *account* y *session* que permite convertir el directorio de acceso del usuario en el directorio raíz de su árbol de directorios, de forma que el usuario queda restringido en ese subárbol.

Módulo pam_cracklib.

Es un módulo de tipo *password* que permite modificar la contraseña de un usuario y comprobar que esta es fuerte, esto es, no es fácil de adivinar. Para ello utiliza la librería del sistema *libcrack* y el diccionario que se encuentra en */usr/lib/cracklib_dict.** (o */usr/lib64/cracklib_dict.**)para encontrar contraseñas débiles. Sus argumentos pueden verse en la siguiente tabla.

Argumento	Descripción
debug	Escribe información de depuración en el log del sistema. Esta información nunca incluye la contraseña introducida.
type=XXX	Reemplaza la palabra UNIX que se muestra cuando se solicita la nueva contraseña por la palabra especificada.
retry=N	Número de reintentos para cambiar la contraseña. Por defecto el valor es 1.
difork=N	Número de caracteres que debe diferir la nueva contraseña de la anterior. Por defecto el valor es 10.
minlen=N	Número mínimo de caracteres que son aceptables para la contraseña. Es necesario tener en cuenta que algunos caracteres pueden contar como más de 1.
dcredit=N	Si N es mayor que 0 indica por cuantos caracteres cuenta un digito, siendo el valor por defecto de 1. Si N es menor que 0 indica el número mínimo de dígitos que tiene que tener la contraseña.
ucredit=N	Igual que dcredit pero para las letras mayúsculas.
lcredit=N	Igual que dcredit pero para las letras minúsculas.
ocredit=N	Igual que dcredit pero para los caracteres que no son letras o números.
use_authtok	Fuerza a utilizar la contraseña solicitada por un módulo previo de tipo password.

Algunos ejemplos de su uso son los siguientes:

```
password  required       pam_cracklib.so retry=3

password  required       pam_cracklib.so \
          difok=3 minlen=15 dcredit= 2 ocredit=2

password  required       pam_cracklib.so \
```

```
          dcredit=-1 ucredit=-1 ocredit=-1 lcredit=0 minlen=8

password  required       pam_cracklib.so \
          dcredit=0 ucredit=0 ocredit=0 lcredit=0 minlen=8
```

Módulo pam_deny.

Es un módulo de tipo *auth*, *account*, *password* y *session* que devuelve el valor incorrecto en cualquiera de los tipos. Un ejemplo de su uso es:

```
account  required       pam_deny.so
```

Módulo pam_env.

Este módulo es de tipo *auth* y permite asignar o quitar valores a las variables de ambiente del sistema. Su configuración se realiza por defecto en el fichero */etc/security/pam_env.conf* y lee variables directamente del fichero */etc/environment*. Sus argumentos pueden verse en la siguiente tabla.

Argumento	Descripción
debug	Escribe información en el log del sistema.
conffile=	Cambia el fichero de configuración por defecto al indicado.
envfile=	Cambia el fichero desde donde se leen directamente variables.
readenv=0|1	Desactiva (0) o activa (1) la lectura de variables del fichero */etc/environment* o su sustituto. Por defecto esta activo (1).

Módulo pam_fprintd.

Es un módulo de tipo *auth* que comprueba la huella dactilar del usuario y comprueba si es valida o no para acceder al sistema. Un ejemplo de su uso es el siguiente:

```
auth  sufficient       pam_fprintd.so
```

Módulo pam_ftp.

Es un módulo de tipo *auth* que solicita el correo de un usuario para que acceda al sistema. Sus argumentos se encuentran en la tabla siguiente.

Argumento	Descripción
debug	Escribe información en el log del sistema.
users=XXX,YYY,...	En lugar de a los usuarios ftp o anonymous, provee acceso anónimo a los usuarios especificados.
ignore	No prestar atención al correo suministrado.

Un ejemplo de su uso es el siguiente:

```
auth  sufficient       pam_ftp.so
```

Módulo pam_group.

Es un módulo de tipo *auth* que configura el grupo del usuario que esta accediendo al sistema. Su configuración se realiza mediante el fichero /*etc/security/group.conf*.

Módulo pam_issue.

Es un modulo de tipo *auth* que muestra el contenido del fichero */etc/issue* cuando un usuario intenta acceder al sistema mediante *telnet*. Sus argumentos se encuentran en la siguiente tabla:

Argumento	Descripción
issue=XXX	Utilizar el fichero indicado en vez del fichero por defecto.
noesc	Deshabilitar el código de escape.

Módulo pam_keyinit.

Es un módulo de tipo *session* que comprueba que el proceso que invoca la sesión tiene una clave de sesión diferente de la clave de sesión por defecto. El módulo comprueba si la clave de sesión es la clave por defecto y si es así, crea una nueva clave de sesión y reemplaza la anterior. Sus argumentos son:

Argumento	Descripción
debug	Escribe información en el log del sistema.
force	Cambia la clave de sesión del proceso que invoca la sesión de forma incondicional.
revoke	Elimina la clave de sesión del proceso que invoca la sesión al terminar esté si la clave de sesión fue creada con anterioridad.

Un ejemplo de su uso es el siguiente:

```
session  optional       pam_keyinit.so force revoke
```

Módulo pam_lastlog.

Es un módulo de tipo *session* que utiliza el fichero */var/log/lastlog* para almacenar información sobre el acceso de los usuarios al sistema. Además, muestra una línea con la información sobre el último acceso realizado por el usuario al sistema. Sus argumentos se encuentran en la siguiente tabla.

Argumento	Descripción
debug	Escribe información en el log del sistema.
nodate	No mostrar la información sobre la fecha y hora del último acceso.
noterm	No mostrar la información sobre la terminal utilizada en el último acceso.
nohost	No mostrar la información sobre el ordenador desde el que se realizó el último acceso.

silent	No mostrar ninguna información sobre el último acceso, solo se actualizará la información de registro.
never	Si el usuario nunca ha entrado en el sistema indicarlo mostrando, además, un mensaje de bienvenida.

Un ejemplo de su uso es el siguiente:

```
session  optional       pam_lastlog.so
```

Módulo pam_limits.

Es un módulo de tipo *session* que establece el límite de recursos que un usuario puede tomar en una sesión. Se configura mediante el fichero */etc/security/limits.conf*, que debe ser un fichero con permisos de solo lectura para el administrador, y requiere un kernel que soporte la limitación de recursos. Sus argumentos se encuentran en la tabla siguiente.

Argumento	Descripción
debug	Escribe información en el log del sistema.
conf=	Cambia el fichero de configuración por defecto al indicado.
change_uid	Cambia el UID del usuario al que se le aplican los limites.
utmp_early	Elimina el registro de acceso del usuario al sistema si algún módulo anterior lo ha añadido al mismo.

Un ejemplo de uso es:

```
session  required       pam_limits.so
```

Módulo pam_listfile.

Es un módulo de tipo *auth* que permite, mediante un fichero arbitrario, admitir o denegar el acceso a determinados servicios. Sus argumentos son los siguientes:

- `onerr=succeed|fail`
- `sense=allow|deny`
- `file=filename`
- `item={user | tty | rhost | ruser | group | shell}`
- `apply=usuario | @grupo.`

El modulo coge el elemento del tipo especificado y busca una instancia del mismo en el fichero especificado por *file*[8]. Si la instancia es encontrada, se devuelve el valor correcto si *sense=allow*, devolviéndose el valor incorrecto si *sense=deny*. Si el fichero no existe se devuelve el valor indicado en *onerr*. Por último, *apply=* permite indicar un usuario o un grupo de usuarios a los que se aplica esta comprobación. Ejemplos de utilización de este módulo son los siguientes:

```
auth  required       pam_listfile.so \
      onerr=succeed item=user sense=deny file=/etc/ftpusers
```

[8] El fichero debe contener una única línea por instancia listada.

```
auth  required       pam_listfile.so \
      onerr=fail item=user sense=allow file=/etc/loginusers
```

Módulo pam_localuser

Es un módulo de tipo *account*, *auth*, *password* y *session* que comprueba que un usuario se encuentra en el fichero que contiene los usuarios locales del sistema, generalmente */etc/passwd*. Sus argumentos son:

Argumento	Descripción
debug	Escribe información en el log del sistema.
file=	Utiliza el fichero indicado en lugar del fichero por defecto.

Un ejemplo de su uso es el siguiente:

```
account    sufficient     pam_localuser.so
```

Módulo pam_loginuid.

Es un módulo de tipo *session* que asigna los atributos del identificador del usuario al proceso que es autenticado. Solo debe usarse en los procesos *login*, *sshd*, *gdm*, *vsftpd*, *crond*, *at* y *remote*. Sus argumentos son:

Argumento	Descripción
require_auditd	Comprueba si se esta ejecutando el demonio de *auditd* y en caso contrario devuelve fallo.

Un ejemplo de su uso es el siguiente:

```
session    required     pam_loginuid.so
```

Módulo pam_mail.

Es un módulo de tipos *auth* y *session* que mira si un usuario tiene correo en su buzón, que por defecto debe estar en el directorio */var/spool/mail*. Los posibles argumentos se encuentran en la siguiente tabla.

Argumento	Descripción
debug	Escribe información en el log del sistema.
dir=	Cambia el directorio por defecto donde mirar si el usuario tiene correo.
nopen	No mostrar información sobre la existencia de nuevo correo. Se utiliza para coger el valor de la variable de ambiente MAIL.
close	Indica si el usuario tiene correo y su permiso de acceso se encuentra revocado.
noenv	No coger el valor de la variable de ambiente MAIL.
empty	Indicar que el correo del usuario esta vacío.
hash=	Profundidad de directorios hasta la que debe mirarse si se encuentra el fichero con el correo del usuario.

standard	Mostrar información en el formato antiguo, que implica de forma implícita el uso de empty.
quiet	Solo informar cuando exista nuevo correo.

Un ejemplo de uso es el siguiente:

```
session  optional       pam_mail.so
```

Módulo pam_mkhomedir.

Es un módulo de tipo *session* que crea el directorio raíz de un usuario si no existe. Sus argumentos se pueden ver en la tabla siguiente.

Argumento	Descripción
skel=	Directorio desde el que se copiaran unos ficheros iniciales al directorio creado.
umask=	Valor octal con la máscara con cuyo valor se crearan los ficheros.

Un ejemplo de uso es el siguiente:

```
session  required pam_mkhomedir.so skel=/etc/skel/ umask=0022
```

Módulo pam_motd.

Es un módulo de tipo *session* que muestra el contenido del fichero */etc/motd*[9] cuando un usuario accede al sistema. Su argumento *motd=<fichero>* permite modificar el fichero que se mostrará.

Módulo pam_nologin.

Es un módulo de tipos *auth* y *account* que si existe el fichero */etc/nologin* solo permite el acceso al usuario root, mostrando a todos los usuarios el contenido del fichero. Sus argumentos se encuentran en la siguiente tabla.

Argumento	Descripción
successok	Devuelve el valor correcto en lugar del valor ignorar si el fichero existe.
file=	Cambia el fichero a buscar y mostrar.

Módulo pam_permit.

Es un módulo de tipo *auth*, *account*, *password* y *session* que devuelve el valor correcto[10]. Un ejemplo de uso es:

```
account  required       pam_permit.so
```

[9] El fichero /etc/motd es el Message Of The Day, mensaje del día.
[10] Este módulo debe utilizarse con mucha precaución, pues devuelve el valor correcto sin realizar ninguna comprobación.

Módulo pam_pwdb.

Es un módulo de tipo *auth*, *account*, *password* y *session* que es un interfaz genérico a la librería de base de datos de contraseñas (*libpwdb*). El módulo tiene distintos argumentos según el tipo para el que se utilice. Así para el tipo *auth* los argumentos validos se encuentran en la siguiente tabla:

Argumento	Descripción
debug	Escribe información en el log del sistema.
user_first_pass	Utiliza la contraseña introducida previamente en otro módulo del mismo tipo. En caso de que no exista devuelve fallo.
try_first_pass	Utiliza la contraseña introducida en otro módulo del mismo tipo si esta existe. Si no existe se solicita la contraseña.
nullok	Permite por defecto que un usuario acceda si su contraseña es nula, pues el comportamiento por defecto es denegar dicho acceso.
nodelay	Elimina el retardo existe entre dos reintentos de introducción de la contraseña.
likeauth	Indica que devuelva los mismos valores que cuando es llamado como módulo de tipo account.
noreap	Modifica el funcionamiento por defecto del módulo en aspectos de su funcionamiento mediante procesos hijos, etc.

Mientras que para el tipo *account* sus argumentos son:

Argumento	Descripción
debug	Escribe información en el log del sistema.

Y para el tipo *password* sus argumentos son:

Argumento	Descripción
debug	Escribe información en el log del sistema.
nullok	Permite la modificación de una contraseña que se encuentra vacia.
not_set_pass	No hacer disponible la vieja y nueva contraseña al resto de módulos.
use_authtok	Obliga a solicitar una nueva contraseña aunque ya se haya introducido previamente.
try_first_pass	Utiliza la contraseña introducida en otro módulo del mismo tipo si esta existe. Si no existe se solicita la contraseña.
use_first_pass	Utiliza la contraseña introducida previamente en otro módulo del mismo tipo. En caso de que no exista devuelve fallo.
md5	Utilizar encriptación md5 en lugar de la encriptación de crypt.
bigcrypt	Utilizar encriptación bigcrypt en lugar de la encriptación de crypt.
shadow	La contraseña se almacena en modo oculto (shadow).
radius	La contraseña se almacenará para un servidor de contraseñas radius.
unix	La contraseña se almacena del modo estándar de UNIX.

Por último, para el tipo *session* el módulo no admite ningún argumento.

Un ejemplo de su uso es el siguiente:

```
account  required       pam_pwdb.so
```

Módulo pam_rhosts_auth.

Es un modulo de tipo *auth* que habilita la autenticación por red de servicios remotos como *rlogin* o *rsh*[11]. Como indicación, la autenticación remota se basa en los ficheros */etc/hosts.equiv* y *~/.rhosts* que indican los ordenadores que son equivalentes y por tanto, a los que se puede permitir el acceso sin necesidad de que el usuario facilite su contraseña. Sus argumentos se encuentran en la siguiente tabla.

Argumento	Descripción
debug	Escribe información en el log del sistema.
no_warn	No enviar avisos al usuario sobre fallos, etc.
no_hosts_equiv	Ignorar el contenido del fichero /etc/hosts.equiv.
hosts_equiv_rootok	Permitir el uso del fichero /etc/hosts.equiv para el administrador. Esta opción no funciona si se utiliza la opción no_hosts_equiv.
no_rhosts	Ignorar el contenido del fichero ~/.rhosts.
privategroup	Utilizar el fichero ~/.rhosts, aunque tenga permisos de escritura para los miembros del grupo del usuario, siempre que el usuario sea el único miembro del grupo.
promiscuous	Permitir el uso de la opción '+' en los ficheros de equivalencia.
supress	Suprimir la escritura de cualquier información de log en el sistema.

Un ejemplo de su uso es el siguiente:

```
auth  sufficient      pam_rhosts_auth.so no_rhosts
```

Módulo pam_rootok.

Es un módulo de tipo *auth* que se utiliza cuando el administrador desea obtener acceso a un servicio sin tener que introducir su contraseña. El ejemplo más típico es cuando el administrador desea convertirse en otro usuario del ordenador utilizando el comando su. Sus argumentos se encuentran en la siguiente tabla.

Argumento	Descripción
debug	Escribe información en el log del sistema.

Un ejemplo de su utilización es el siguiente:

```
auth  sufficient      pam_rootok.so
```

[11] Estos servicios son desaconsejados en cualquier sistema seguro por la vulnerabilidad que introducen en el sistema.

Módulo pam_securetty.

Es un módulo de tipo *auth* que utiliza el fichero */etc/securetty* para comprobar si una terminal es segura, denegando el acceso al administrador si la terminal no es segura.

Módulo pam_selinux.

Es un módulo de tipo *session* que asigna el contexto de seguridad por defecto para la shell que se ejecutará. Sus parámetros son:

Argumento	Descripción
close	Ejecutar solo la porción del módulo de cierre de sesión.
debug	Escribe información de depuración en el log del sistema.
open	Ejecutar solo la porción del módulo de apertura de sesión.
nottys	No asignar las ttys del contexto de seguridad.
verbose	Informar al usuario cuando el contexto de seguridad sea asignado.
select_context	Preguntar al usuario por el contexto de seguridad deseado.
env_params	Obtener un contexto de seguridad a media de las variables de ambiente de PAM. Esta opción es incompatible con la anterior.
use_current_range	Usar el nivel del proceso actual en vez del nivel por defecto para el usuario. Esta opción suprime cualquiera de las dos opciones anteriores.

Un ejemplo de su utilización es el siguiente:

```
session    required     pam_selinux.so close
session    required     pam_selinux.so open env_params
```

Módulo pam_sepermit.

Es un módulo de tipos *auth* y *account* que permite o deniega el acceso de un usuario dependiendo de si se encuentra listado en un fichero y del valor de SELinux[12]. Si el usuario se encuentra en el fichero y SELinux esta en modo *enforcing* el acceso es permitido, siendo denegado en caso contrario. Si el usuario no se encuentra listado en el fichero se devuelve un valor de ignorado (PAM_IGNORE). Sus parámetros son:

Argumento	Descripción
debug	Escribe información de depuración en el log del sistema.
conf=	Fichero de configuración a utilizar en lugar del fichero por defecto.

Un ejemplo de su utilización es el siguiente:

[12] NSA Security-Enhanced Linux es una implementación de una arquitectura de control de acceso en Linux que permite especificar políticas de control de acceso y de uso del sistema, lo que permite no solo comprobar que un usuario o proceso tiene permisos para utilizar el sistema, sino también que lo esta realizando de forma correcta y autorizada.

```
auth       required     pam_sepermit.so
```

Módulo pam_shells.

Es un módulo de tipo *auth* que utiliza el fichero */etc/shells* para comprobar si la shell del usuario se encuentra en ese fichero y por tanto es autorizada.

Módulo pam_succeed_if.

Es un módulo de tipo *auth* que permite comprobar características de un usuario antes de permitir su acceso. Sus argumentos son los siguientes:

Argumento	Descripción
debug	Escribe información en el log del sistema.
use_id	Evaluar las condiciones utilizando el identificador del usuario como el que se ejecuta la aplicación en lugar de utilizar el identificador del usuario que esta siendo autenticado.
quiet	No escribir el fallo o acierto en el log del sistema.
quiet_fail	No escribir el fallo en el log del sistema.
quiet_success	No escribir el acierto en el log del sistema.
campo < número	Campo tiene un valor numérico menor que número.
campo <= número	Campo tiene un valor numérico menor o igual que número.
campo eq número	Campo tiene un valor numérico igual a número.
campo >= número	Campo tiene un valor numérico mayor o igual que número.
campo > número	Campo tiene un valor numérico mayor que número.
campo neq número	Campo tiene un valor numérico diferente de número.
campo = cadena	Campo es exacto a la cadena de texto.
campo != cadena	Campo es distinto de la cadena de texto.

Donde campo puede tomar los valores *user*, *gid*, *shell*, *home* o *service*.

Módulo pam_tally.

Es un módulo de tipos *auth* y *account* que mantiene una cuenta de los accesos fallidos al sistema en el fichero /*var/log/faillog*, pudiendo borrar la cuenta si un acceso tiene éxito y denegar el acceso si los intentos fallidos exceden de un determinado valor. Sus argumentos en tipo *auth* son los siguientes:

Argumento	Descripción
onerr= succeed | fail	Indica el valor a devolver si el fichero de accesos no existe.
file=	Fichero utilizado en lugar del fichero por defecto.
audit	Escribe el nombre del usuario introducido si el usuario no existe.
deny=N	Denegar el acceso después de N fallos registrados.
lock_time=N	Bloquear el acceso N segundos después del último intento fallido.

unlock_time=N	Desbloquear el acceso al sistema bloqueado después de exceder el número de fallos permitidos al cabo de N segundos.
magic_root	No incrementar la cuenta si el usuario es el administrador.
even_deny_root_account	Permitir bloquear la cuenta del administrador.
per_user	Si el fichero de configuración posee una línea especifica para este usuario, utilizar esos valores en lugar de los valores de la línea de comandos.
no_lock_time	No utilizar el campo del fichero de configuración que indica el tiempo de bloqueo para este usuario.
no_reset	No inicializar la cuenta de fallos aún en caso de un acceso correcto.

Y en tipo *account* son

Argumento	Descripción
onerr= succeed | fail	Indica el valor a devolver si el fichero de accesos no existe.
file=	Fichero utilizado en lugar del fichero por defecto.
magic_root	No decrementar o inicializar la cuenta si el usuario es el administrador.
no_reset	No inicializar la cuenta de fallos aún en caso de un acceso correcto.

Módulo pam_time.

Es un módulo de tipo *account* que permite restringir el acceso a los servicios en función de franjas horarias. Utiliza el fichero de configuración */etc/security/time.conf*. Un ejemplo de uso es el siguiente:

```
account  required       pam_time.so
```

Módulo pam_unix.

Es un módulo de tipo *auth*, *account*, *password* y *session* que proporciona el mecanismo clásico de UNIX para acceder al sistema mediante la comparación de la contraseña con la existente en */etc/passwd* o en */etc/shadow* si están habilitadas las contraseñas ocultas (*shadow*). El módulo tiene distintos argumentos según el tipo para el que se utilice. Así para el tipo *auth* los argumentos validos se encuentran en la siguiente tabla:

Argumento	Descripción
debug	Escribe información en el log del sistema.
audit	Escribe una información mucho más detallada en el log del sistema.
user_first_pass	Utiliza la contraseña introducida previamente en otro módulo del mismo tipo. En caso de que no exista devuelve fallo.

try_first_pass	Utiliza la contraseña introducida en otro módulo del mismo tipo si esta existe. Si no existe se solicita la contraseña.
nullok	Permite por defecto que un usuario acceda si su contraseña es nula, pues el comportamiento por defecto es denegar dicho acceso.
nodelay	Elimina el retardo existe entre dos reintentos de introducción de la contraseña.
noreap	Modifica el funcionamiento por defecto del módulo en aspectos de su funcionamiento mediante procesos hijos, etc.

Mientras que para el tipo *account* sus argumentos son:

Argumento	**Descripción**
debug	Escribe información en el log del sistema.
audit	Escribe una información mucho más detallada en el log del sistema.

Y para el tipo *password* sus argumentos son:

Argumento	**Descripción**
debug	Escribe información en el log del sistema.
audit	Escribe una información mucho más detallada en el log del sistema.
nullok	Permite la modificación de una contraseña que se encuentra vacia.
not_set_pass	No hacer disponible la vieja y nueva contraseña al resto de módulos.
use_authtok	Obliga a solicitar una nueva contraseña aunque ya se haya introducido previamente.
try_first_pass	Utiliza la contraseña introducida en otro módulo del mismo tipo si esta existe. Si no existe se solicita la contraseña.
use_first_pass	Utiliza la contraseña introducida previamente en otro módulo del mismo tipo. En caso de que no exista devuelve fallo.
md5	Utilizar encriptación md5 en lugar de la encriptación de crypt.
bigcrypt	Utilizar encriptación bigcrypt en lugar de la encriptación de crypt.
shadow	La contraseña se almacena en modo oculto (shadow).
nis	Utiliza la llamada a procedimiento remoto de NIS para almacenar la contraseña.
remember=N	Recuerda las últimas N contraseñas del usuario en el fichero */etc/security/opasswd* para evitar su reutilización.

Por último, para el tipo *session* el módulo no admite ningún argumento.

Ejemplos de uso son lo siguientes:

```
auth       required   pam_unix.so
account    required   pam_unix.so
password   required   pam_unix.so
password   required   pam_unix.so use_authtok nullok md5
session    required   pam_unix.so
```

Módulo pam_warn.

Es un módulo de tipo *auth* y *password* que escribe información en el log del sistema sobre el usuario y sistema remoto, siempre que dicha información este disponible. Un ejemplo de su uso es el siguiente:

```
auth  required       pam_warn.so
```

Módulo pam_wheel.

Es un módulo de tipo *auth* y *account* que solo permite el acceso como administrador a miembros del grupo de administración (gid=0). Sus argumentos se encuentran en la tabla siguiente:

Argumento	Descripción
debug	Escribe información en el log del sistema.
use_uid	Utilizar el grupo uid actual en lugar del facilitado por la llamada a getlogin() con el nombre del usuario.
trust	Devolver correcto si el usuario actual es miembro del grupo de administración.
deny	Denegar el acceso si el usuario actual es miembro del grupo de administración.
group=XXX	Utilizar el nombre del grupo XXX en lugar del grupo de administración.

Un ejemplo de su uso es:

```
auth     required       pam_wheel.so
```

Ejemplos de ficheros PAM.

Veamos a continuación algunos ejemplos de ficheros de configuración de servicios cuyo acceso utiliza PAM[13], previamente veremos el fichero *password-auth*, que es incluido por otros ficheros:

```
#%PAM-1.0
auth        required      pam_env.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 1000
quiet_sucess
auth        required      pam_deny.so

account     required      pam_unix.so
account     sufficient    pam_localuser.so
account     sufficient    pam_succeed_if.so uid < 1000 quiet
account     required      pam_permit.so

password    requisite     pam_cracklib.so try_first_pass retry=3
authtok_type=
password    sufficient    pam_unix.so sha512 shadow nullok
try_first_pass use_authtok
```

[13] Los servicios aquí comentados serán vistos en temas posteriores.

```
password    required      pam_deny.so

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
session     [success=1 default=ignore] pam_succeed_if.so service
in crond quiet use_id
session     required      pam_unix.so
```

Dentro del tipo *auth*, la primera línea indica que se requiere poder asignar valores a las variables de ambiente del sistema, mientras que la segunda indica que es suficiente con comprobar el usuario y la contraseña, permitiendo utilizar contraseñas nulas y reutilizar los datos, si estos ya han sido solicitados, o solicitarlos en caso contrario. La cuarta línea indica que es un requisito que si el usuario tiene un identificador mayor o igual que 1000 no se escriba información en el log (usuario no privilegiado) y la última línea indica que devuelva fallo.

En el tipo *account*, la primera línea indica que se requiere que la cuenta no haya caducado, mientras que la segunda línea indica que es suficiente que el usuario se encuentre en el fichero de contraseñas y la tercera que pertenezca al grupo de usuarios privilegiados, no escribiendo nada en el log, la última línea indica que se devuelva correcto.

En el tipo *password*, la primera línea indica que si la contraseña ha caducado el usuario proporcione una nueva contraseña teniendo 3 intentos para ello, mientras que la segunda indica que se utilizan contraseñas *shadow* cifradas mediante sha512 y permitiendo contraseñas nulas. Por último, se indica que se devuelva fallo.

En el tipo *session* se requiere, en la primera línea, que se establezcan los límites de recursos del usuario, mientras que en la segunda se indica que se ejecute las acciones estándar de establecimiento de sesión de UNIX.

El funcionamiento global de cada tipo es sencillo. Así, por ejemplo, en el tipo *auth*, la primera línea indica que si no es posible asignar valores a las variables de ambiente la autorización fallará (*required*), pero antes se continuará ejecutando el resto de módulos de autorización. Si ha sido posible asignar valores a las variables de ambiente y se proporciona de forma correcta el usuario y la contraseña, la autorización será correcta (*sufficient*) y se enviará la respuesta al servicio que llamo a PAM. En este punto, se ejecutará el siguiente módulo de escritura en el log del sistema y, por último, el módulo *pam_deny* fija que se devuelva fallo.

Servicio de SSH (fichero /etc/pam.d/sshd).

El servicio de SSH permite el acceso remoto de los usuarios a una shell del sistema utilizando un mecanismo de cifrado en la comunicación que imposibilita que otros usuarios de la red puedan interceptar y descifrar la comunicación. Su fichero de configuración es el siguiente[14]:

```
#%PAM-1.0
auth       required     pam_sepermit.so
```

[14] En la configuración se puede ver que se incluye un fichero llamado *postlogin* que por defecto esta vacío.

```
auth       substack     password-auth
auth       include      postlogin
account    required     pam_nologin.so
account    include      password-auth
password   include      password-auth
# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so close
session    required     pam_loginuid.so
# pam_selinux.so open should be followed by sessions to be
executed in the user context
session    required     pam_selinux.so open env_params
session    optional     pam_keyinit.so force revoke
session    include      password-auth
session    include      postlogin
```

En el podemos ver, aparte de la inclusión del fichero *password-auth* en todos los tipos, que en el tipo *auth* es requerido previamente el módulo *pam_sepermit* para comprobar el estado de SELinux, en el tipo *account* es requerido previamente el módulo *pam_nologin* para comprobar si el acceso de usuarios que no sean root esta permitido, mientras que en el tipo *session* es requerido previamente *pam_selinux* para ejecutar el cierre de cualquier sesión previa y *pam_loginuid* para asignar el UID, etc., a la sesión, otra vez *pam_selinux* para abrir la sesión con variables de ambiente adecuadas a SELinux y *pam_keyinit* para forzar la creación de una nueva clave de sesión y su eliminación al terminar la sesión.

Servicio de FTP (fichero /etc/pam.d/vsftpd).

```
#%PAM-1.0
session    optional     pam_keyinit.so force revoke
auth       required     pam_listfile.so item=user sense=deny
file=/etc/vsftpd/ftpusers onerr=succeed
auth       required     pam_shells.so
auth       include      password-auth
account    include      password-auth
session    required     pam_loginuid.so
session    include      password-auth
```

En el podemos ver, aparte de la inclusión del fichero *password-auth* en todos los tipos utilizados (no se utiliza el tipo *password*), que en el tipo *auth* son requeridos previamente los módulos *pam_listfile*, que indica que se compruebe si el usuario se encuentra en el fichero indicado, devolviendo error si el usuario se encuentra en el mismo, y el módulo *pam_shells*, que comprueba si su shell es autorizado y en el tipo session se requiere previamente el modulo *pam_keyinit* para forzar la creación de una nueva clave de sesión y su eliminación al terminar la sesión, siendo utilizado posteriormente el módulo *pam_loginuid* para asignar el identificador del proceso.