Capítulo 2

Los módulos PAM

Descripción y orígenes

Características

Archivos de configuración

Ejemplos

Capítulo 2 Los módulos PAM

Los programas que permiten a los usuarios acceder a un sistema deben verificar la identidad del usuario a través del proceso que acabamos de discutir en el capítulo anterior, llamado autenticación.

Históricamente, cada programa tiene su forma particular de realizar la autenticación. Bajo Linux, muchos de esos programas pueden ser configurados para usar un proceso de autenticación centralizado utilizando para ello los módulos PAM (Pluggable Authentication Modules).

A continuación veremos con detalle estos módulos; su arquitectura y su uso. Veremos además algunos de los más importantes deteniéndonos en los más usados y otros que centran nuestro interés.

Prestaremos especial atención a la configuración, de forma que podamos establecer una política de autenticación viable y conforme a nuestros fines.

2.1 Descripción y orígenes

PAM es un método flexible para la autenticación de usuarios. Concretamente PAM se refiere a la librería que lleva su nombre, implementada por una serie de módulos (Pluggable Autentication Modules).

Estos módulos constituyen una API generalizada para servicios relacionados con la autenticación, que permiten al administrador añadir nuevos métodos de autenticación simplemente instalando nuevos módulos, y cambiar las políticas de autenticación editando los ficheros de configuración.

PAM fue definido y desarrollado en 1995 por Vipin Samar and Charlie Lai de Sun Microsystems, y no ha cambiado mucho desde entonces. En 1997 Open Group publicó la especificación preliminar XSSO (X/Open Single Sign-on) que estandarizó la API y añadió algunas extensiones.

En el momento de escribir este documento, esta especificación todavía no era un estándar, pero si se usa ampliamente en casi todas las distribuciones y podríamos decir que se ha convertido en un estándar de facto.

2.2 Características

PAM ofrece las siguientes ventajas:

- Un esquema de autenticación común y centralizado que se puede usar con una gran variedad de aplicaciones.
- PAM puede ser usado con diferentes aplicaciones sin tener que recompilarlas para soportar PAM específicamente. A menudo sólo tendremos que modificar el fichero de configuración de cada aplicación específica.
- Permite gran flexibilidad y control de la autenticación para el administrador del sistema y el desarrollador de la aplicación.
- Los desarrolladores de aplicaciones no necesitan desarrollar su programa para usar un determinado esquema de autenticación. En su lugar, pueden concentrarse puramente en los detalles de su programa.
- El administrador puede elegir un método de autenticación por defecto para las aplicaciones no definidas.
- Posibilidad de múltiples claves en entornos de alta seguridad.
- Posibilidad de paso de parámetros opcionales a los servicios.

Por último decir que PAM utiliza una arquitectura conectable y modular, que otorga al administrador del sistema de una gran flexibilidad en establecer las políticas de autenticación para el sistema.

En la mayoría de los casos, el archivo de configuración por defecto para una aplicación tipo PAM es suficiente. Sin embargo, algunas veces es necesario modificar el archivo de configuración. Debido a que un error en la configuración de PAM puede comprometer la seguridad del sistema, es importante comprender la estructura del estos archivos antes de hacer cualquier modificación.

2.3 Funcionamiento

A continuación mostramos el esquema de funcionamiento de PAM:

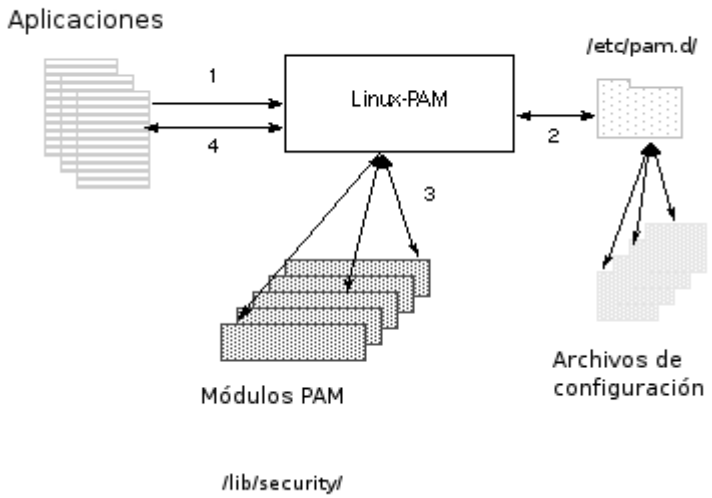

Figura 2.1 Esquema de funcionamiento de PAM

1) La aplicación, por ejemplo login, realiza una llamada inicial a PAM
2) PAM localiza el fichero de configuración adecuado (en /etc/pam.d o dentro del fichero /etc/pam.conf) para obtener la lista de módulos necesarios para el servicio que lo solicita.
3) Entonces PAM carga cada módulo en el orden establecido en el fichero de configuración para el procesamiento. Dependiendo de los parámetros de configuración puede que alguno de los módulos cargados no sean finalmente invocados.
4) Algunos de los módulos o incluso todos, pueden necesitar tener algún tipo de conversación con el usuario a través de la aplicación que llamó a PAM. Esta conversación a menudo incluye una solicitud de algún tipo de información al usuario, como puede ser un password, y la recepción de una respuesta. Si la respuesta del usuario satisface al módulo PAM en particular, o se satisface de otra forma, el control se devuelve a PAM y continúa procesando el siguiente módulo.

   Los pasos 3 y 4 se repiten para cada módulo que aparece en el fichero de configuración de la aplicación en cuestión.

5) Por último, el procesamiento se completa devolviendo un valor representativo de éxito, o de fallo. Generalmente el valor del mensaje que se le presenta al usuario no es indicativo de la causa del fallo, en caso de que ocurra. De esta forma no se puede utilizar esta información y prevenir así diversos tipos de ataques.

   Afortunadamente, la mayoría de módulos disponen de diversos niveles de registro, permitiendo a los administradores de sistema resolver problemas e identificar violaciones de seguridad.

2.4 Archivos de configuración

Cada aplicación que soporta PAM debe disponer de un archivo de configuración. Cada uno de estos archivos guarda la política de acceso de cada servicio o aplicación.

El nombre de cada fichero de configuración suele ser bastante representativo del servicio al que está asociado. Por ejemplo, el servicio login que ya hemos visto dispone de /etc/pam.d/login. Veamos dicho fichero para hacernos una primera idea de su organización.

```
#%PAM-1.0
auth            required     pam_securetty.so
auth            required     pam_stack.so service=system-auth
auth            required     pam_nologin.so

account         required     pam_stack.so service=system-auth

password        required     pam_stack.so service=system-auth

session         required     pam_stack.so service=system-auth
session         optional     pam_console.so
```

Cada archivo de configuración PAM contiene un grupo de directivas formateadas como vemos en el ejemplo. Responden al siguiente esquema:

```
Interfaz de módulo     Indicadores de control     Rutas     Argumentos
```

Pasemos pues a detallar estos campos.

2.4.1 Interfaz de módulo

Existen cuatro tipos de módulos PAM usados para controlar el acceso a los servicios. Todos ellos están relacionados con los diferentes aspectos del proceso de autenticación.

- **Auth**. Estos módulos autentican a los usuarios de forma básica. Para

ello piden contraseñas y las validan. También pueden establecer credenciales, tales como membrecía o tickets Kerberos.

- **Account**. Estos módulos comprueban que la autenticación sea válida. Entre estas comprobaciones se encuentran determinadas acciones como ver si la cuenta no ha caducado, si el usuario tiene permiso para iniciar sesiones a esa hora del día, etc.).
- **Password**. Éstos sirven para establecer y verificar contraseñas.
- **Session**. Estos módulos configuran y administran sesiones de usuarios. Los módulos con esta interfaz también pueden realizar tareas adicionales que sean necesarias para permitir el acceso, como montar el directorio principal del usuario o hacer el correo disponible.

No debemos extrañarnos si alguna vez nos entrontramos un módulo en varias de estas interfaces, o incluso en todas. Por ejemplo, pam_unix.so tiene componentes que direccionan las cuatro interfaces.

En un archivo de configuración PAM la interfaz del módulo es el primer aspecto que tenemos que definir. Por ejemplo, una línea típica sería:

```
auth                required            /lib/securety/pam_unix.so
```

como ya hemos visto en en ejemplo preliminar que mostramos de login.

Esto provoca que PAM se dirija hacia la componente auth del módulo pam_unix.

2.4.2 Indicadores de control

Todos los módulos PAM generan un código de éxito o fracaso cuando son invocados. Los indicadores de control muestran a PAM lo que hacer con el resultado. Además debido a que los módulos son apilables, esto es, pueden colocarse en cascada, los indicadores de control nos dan la posibilidad de

fijar la importancia de cada uno de ellos respecto al objetivo final del proceso de autenticación para el servicio.

Existen cuatro indicadores de control definidos:

- Required. El resultado del módulo debe ser exitoso para que la autenticación continúe. Si un módulo required falla, el usuario no es notificado hasta que los resultados en todos los módulos referenciando esa interfaz sean completados.
- Requisite. Al igual que el módulo anterior el resultado del módulo debe ser exitoso par que al autenticación continue. Sin embargo, si el resultado de un módulo requisite falla, el usuario es notificado de inmediato, con un mensaje del primer módulo required o requisite fracasado.
- Sufficient. El resultado del módulo es ignorado si falla. Pero si el resultado del módulo con indicador sufficient es exitoso y ningún módulo con indicador required ha fallado, entonces no se requiere otro resultado y el usuario es autenticado para el servicio.
- Optional. Se ignora el resultado si el módulo falla. Si el resultado es exitoso, no juega ningún papel en el resultado global del módulo. Un módulo con un modificador optional es necesario para la autenticación exitosa cuando no hay otros módulos referenciando la interfaz.

Un aspecto que debemos remarcar es lo que hemos llamado apilación de módulos. El orden en el que estos módulos son apilados es importante en el caso de los indicadores requisite y sufficient, pero no con required, como ya hemos visto.

2.4.3 Rutas de módulos

Las rutas de los módulos le indican a PAM donde encontrar dicho módulo con la interfaz especificada. Generalmente, se proporciona como una ruta completa, como /lib/security/pam_unix.so. Sin embargo, si no se

proporciona la ruta completa, se asume que el módulo se encuentra en el directorio /lib/security, que es la dirección por defecto para los módulos PAM.

2.4.4 Argumentos de módulo

PAM utiliza los argumentos para pasar información a algunos módulos durante la autenticación.

Por ejemplo, el módulo pam_userdb.so usa claves almacenadas en una base de datos Berkeley DB para autenticar a los usuarios. El módulo toma como argumento db para que la base de datos conozca el fichero base donde buscar, para el servicio solicitado.

Una línea típica que use este módulo sería de este modo:

```
auth     required       /lib/security/pam_userdb.so     db=/etc/claves.db
```

donde el fichero claves.db sería la base de datos que estaríamos usando para este servicio.

Los argumentos inválidos se ignoran y no afectan en ningún modo al éxito o fracaso del módulo. Sin embargo, la mayoría de módulos reportarán un error al archivo /var/log/messages.

Un aspecto importante que hemos comentado poco es el sistema operativo usado. En este caso es Red Hat 9 por lo que el archivo anterior está referido a esta distribución, y por tanto puede variar respecto en otras. En la distribución Debian, por ejemplo, el archivo que se suele usar para ello es /var/log/auth.log.

2.5 Ejemplo de archivos de configuración PAM

Veamos un ejemplo.

```
1. #%PAM 1.0
2. auth           required    /lib/securety/pam_securetty.so
3. auth           required    /lib/securety/pam_unix.so
4. account        required    /lib/securety/pam_unix.so
5. password       required    /lib/securety/pam_cracklib.so retry=3
6. password       required    /lib/securety/pam_unix.so shadow use_authtok
7. session        required    /lib/securety/pam_unix.so
```

La primera línea es un comentario, al igual que cualquier línea que comience con un signo de almohadilla (#).

Las dos siguientes son un ejemplo de módulos apilados. La línea 2 ejecuta una comprobación. Si el usuario accede como root, el tty en el cual el usuario se está conectando queda registrado en el archivo /etc/securetty.

La siguiente línea corresponde a un módulo que solicita la contraseña al usuario y la compara con la encuentra en el fichero /etc/passwd, y si existe con /etc/shadow. Pam_unix.so detecta automáticamente y utiliza las contraseñas shadow para autenticar a los usuarios. El argumento nullok modifica el comportamiento del módulo, permitiendo que acepte contraseñas en blanco.

En la línea 4 tenemos la verificación de la cuenta necesaria. Por ejemplo, para las contraseñas Shadow, el componente de la cuenta del módulo pam_unix.so comprobará si la cuenta ha expirado o si el usuario ha cambiado la contraseña durante el periodo concedido. En el caso de que tengamos que cambiar el password porque haya expirado tenemos la siguiente línea. El módulo pam_cracklib.so pide una nueva contraseña. Además de este uso tiene sus cometidos típicos, como son la evaluación de la contraseña introducida y la confirmación de la nueva.

En cuanto a la evaluación de las contraseñas, pam_cracklib efectúa un ataque de diccionario para ver la robustez de dichas claves. Además comprueba que tienen la longitud mínima y que no son fáciles de averiguar (que no tengan alguna relación sintáctica con el nombre de usuario, por ejemplo).

El argumento retry=3 nos indica las veces que este módulo pedirá una nueva contraseña. Cuando introducimos una clave que no considera válida, nos pida que introduzcamos otra. Esto lo repite durante 3 veces, en este caso.

La línea 6 es la que realmente cambia la contraseña. En ella se especifica que se use la componente password del módulo pam_unix.so para ello. El argumento shadow indica al módulo que cree la contraseña de este tipo y el último, use_authtok, indica que no se solicite de nuevo la contraseña y que use la que registraron los módulos de contraseña anteriores.

2.6 Módulos más comunes

Veamos algunos de los módulos más utilizados. Debemos comentar que los vamos a tratar de forma general. La mayoría de los módulos difieren su comportamiento según cada uno de los cuatro interfaces. Incluso otros no implementan algún interfaz. Además cada uno cuenta con unos argumentos específicos con lo cual este análisis se extendería muchísimo.

Con el afán de no romper el hilo del proyecto, incluimos el análisis detallado de todos los módulos que se utilizarán en el apéndice. No obstante vamos a describir los más usados, para poder así entender los ejemplos propuestos.

pam_deny. Este módulo se utiliza para denegar el acceso. Devuelve un código de error cuando la autenticación falla.

pam_filter. Este módulo no está muy desarrollado todavía. Por el momento el único filtro implementado es la conversión de minúsculas a mayúsculas y viceversa, de un flujo de datos.

pam_env. Este módulo permite establecer variables de entorno usando cadenas fijas de texto.

pam_group. Este módulo provee una configuración de grupo basada en el nombre de usuario y en el terminal desde el que se está pidiendo el servicio en cuestión. Además toma nota de la hora del día en la que se realiza la petición.

pam_lastlog. Este módulo de tipo “session” se encarga de mantener el archivo /var/log/lastlog. Añade una entrada a dicho archivo cuando se abre una sesión, de forma que puede proveer información sobre la última sesión que abrió un usuario en particular.

pam_limits. Este módulo permite establecer límites en los recursos del sistema que puede obtener un usuario en su sesión. Estos límites se fijan en el fichero de configuración /etc/security/limits.conf.

pam_nologin. Este módulo provee autenticación sin introducir el login. Si el archivo /etc/nologin existe, sólo el root puede acceder; los otros usuarios son rechazados mediante un código de error.

pam_permit. Este módulo permite accesos. Debido a esto debe ser usado con mucha precaución.

pam_pwdb. Este módulo puede reemplazar a los módulos pam_unix_... Usa una interfaz genérica a la librería libpwdb (base de datos de claves).

pam_warn. Este módulo se usa principalmente para registrar información sobre la autenticación y actualización de claves.

pam_ldap. Este módulo consulta diversos datos a un directorio usando el protocolo LDAP. Entre los datos consultados se encuentran en nombre de usuario, clave, correo y otro tipo de información.

2.7 Creación de aplicaciones PAM

A continuación vamos a ver lo fácil que resulta crear una aplicación que use autenticación vía PAM.

La aplicación se va a llamar TTT, y usará un fichero de configuración PAM, llamado ttt, que se encontrará en /etc/pam.d, tal como ocurre típicamente. Utilizamos estos nombres para que sean más fáciles de localizar en el código que presentamos a continuación.

```
#include <security/pam_appl.h>
#include <security/pam_misc.h>
#include <pwd.h>
#include <sys/types.h>
#include <stdio.h>
#define MY_CONFIG "ttt"
static struct pam_conv conv = { misc_conv, NULL };

main( )
{
  pam_handle_t *pamh;
  int result;
  struct passwd *pw;
  if ((pw = getpwuid(getuid( ))) == NULL)
    perror("getpwuid");
  else if ((result = pam_start(MY_CONFIG, pw->pw_name, &conv, &pamh)) !=
PAM_SUCCESS)
    fprintf(stderr, "inicio fallido: %d\n", result);
  else if ((result = pam_authenticate(pamh, 0)) != PAM_SUCCESS)
    fprintf(stderr, "fallo en la autenticacion: %d\n", result);
  else if ((result = pam_acct_mgmt(pamh, 0)) != PAM_SUCCESS)
    fprintf(stderr, "acct_mgmt fallod: %d\n", result);
  else if ((result = pam_end(pamh, result)) != PAM_SUCCESS)
    fprintf(stderr, "final fallido: %d\n", result);
  else
    TTT( );            /* Y por fin ejecutamos la aplicación */
}
```

Pues bien, compilamos el código enlazando las librerías adecuadas, que en este caso son libpam y libpam_misc.

```
gcc TTT.c -lpam -lpam_misc
```

Esto es todo. Es evidente que se necesita muy poco código para tener una aplicación TTT genérica que hace uso de la autenticación vía PAM. Resulta verdaderamente fácil dar soporte PAM a una aplicación.

2.8 PAM y la propiedad del dispositivo

En este apartado vamos a comentar una funcionalidad muy interesante que permite PAM y que nos permitirá manipular dispositivos que, a menudo, sólo podría hacerlo el usuario root. Y además de forma muy sencilla.

El proceso para permitir a los usuarios este tipo de accesos se controla mediante el módulo pam_console.so. Se realiza de la siguiente forma:

Cuando el usuario se registra por primera vez en la consola física en una máquina Linux, el módulo pam_console es llamado por login (o por gdm o kdm, si el inicio es gráfico). Entonces el módulo le concede la propiedad de una serie de dispositivos que generalmente sólo puede manipular el root.

Entre estos dispositivos se encuentran el audio, cdrom y disquetera, que son los más usuales.

El siguiente usuario que acceda al sistema ya no dispondría de la propiedad del dispositivo y por tanto necesitaría los privilegios de root para el acceso a dichos dispositivos.

Las funcionalidades que nos permite pam_console tienen sentido cuando hablamos de una estación de trabajo. En este caso, el usuario de ese sistema es normalmente el primero (y el único) que accede a él.

En otros sistemas, como por ejemplo Debian, no existe este módulo. Para acceder en ellos a los dispositivos sin ser usuario root disponemos de diversos grupos. La pertenencia a estos grupos determina la accesibilidad a dichos dispositivos.

En estos casos para conceder el acceso sólo debemos añadir el usuario al grupo adecuado.

```
adduser usuario audio
adduser usuario cdrom
adduser usuario floppy
```

2.8.1 Acceso a aplicaciones

De la misma forma que podemos acceder a dispositivos sin ser usuario root, también podemos ejecutar ciertas aplicaciones. Éstas deben ser especificadas dentro del directorio /etc/security/console.apps mediante un fichero. Este fichero no tiene porqué contener ningún dato, simplemente debe tener el mismo nombre del comando al que se corresponde.

Algunas de las aplicaciones típicas a las que tiene acceso el usuario de la consola son aquellas que apagan o encienden el sistema, como por ejemplo halt, poweroff o reboot.

2.9 Notas finales

Por último señalar que existe otra aproximación en cuanto a la configuración de PAM, que es más antigua y ya no se utiliza. Consiste en un solo fichero (/etc/pam.conf) para la configuración de los distintos servicios. Se antepone una columna con el nombre del servicio y el resto queda igual. Podemos ver un ejemplo:

```
login   auth     required       /lib/security/pam_nologin
login   auth     required       /lib/security/pam_unix.so
login   auth     optional       /lib/security/pam_test.so
use_first_pass
other   auth     required       /lib/security/pam_prohibit.so
...
```

El fichero /etc/pam.conf solamente se utiliza si no existe el directorio que hemos comentado en los apartados anteriores, /etc/pam.d. La opción de usar el fichero /etc/pam.conf no se contempla actualmente, ni se aconseja su uso.