Pluggable Authentication Modules (PAM)

Imobach González Sosa
imobachgs@softhome.net

Manolo Padrón Martínez
manolopm@softhome.net

Versión 1.0

Curso 2003/04

Licencia

Índice

1. Acerca de este documento

Los *Pluggable Authentication Modules* se han convertido en el estándar *de facto* para la autenticación de usuarios en los sistemas UNIX. Su gran flexibilidad ofrece a administradores y desarrolladores un control muy valioso.

Gracias a PAM, los administradores de sistemas pueden modelar e implementar diferentes políticas de autenticación para los distintos usuarios de forma individualizada para cada servicio. Pero hay que manejar estas facilidades con sumo cuidado, ya que una mala decisión o, simplemente, un despiste pueden comprometer gravemente la seguridad del sistema. Por tanto, el administrador tiene que conocer muy bien cómo funciona PAM si, realmente, quiere afinar al máximo el proceso de autenticación de su sistema.

Este documento pretende servir como introducción a PAM, centrándose en la implementación desarrollada por *Red Hat* para *GNU/Linux*, *Linux-PAM*, si bien los conceptos básicos son los mismos que presentara *Sun* en su momento allá por 1995.

Como siempre, para obtener información más precisa, lo mejor es consultar directamente la documentación oficial del proyecto, en http://www.kernel.org/pub/linux/libs/pam/.

2. Fundamentos básicos

2.1. ¿Qué es PAM?

La idea original de los *Pluggable Authentication Modules*, en adelante PAM, fue de Sun y sus especificaciones se encuentran recogidas en [5]. Sin embargo, muchos otros sistemas adoptaron esta solución y cuentan desde hace tiempo con sus propias implementaciones. En este sentido, GNU/Linux no es una excepción y, gracias a *Red Hat*, disfruta ya desde hace años de la funcionalidad que ofrece Linux-PAM[1].

Pero, ¿qué es PAM exactamente? Tal y como puede leerse en la FAQ[2] oficial del proyecto, PAM es, básicamente, un mecanismo flexible para la autenticación de usuarios. Y quizás esta característica, la flexibilidad, sea su aportación más importante.

A lo largo de los años, desde los primeros sistemas UNIX, los mecanismos de autenticación han ido evolucionando y han aparecido nuevas opciones: desde mejoras del clásico `/etc/passwd` —como la *shadow*— hasta dispositivos hardware orientados a la autenticación. Y, claro está, cada vez que aparecía y se popularizaba un nuevo método, los desarrolladores debían modificar sus programas para darles soporte.

PAM permite el desarrollo de programas independientes del mecanismo de autenticación a utilizar. Así es posible que un programa que aproveche las facilidades ofrecidas por PAM sea capaz de utilizar desde el sencillo `/etc/passwd` hasta dispositivos hardware —como lectores de huella digital—, pasando por servidores *LDAP*[3] o sistemas de gestión de bases de datos. Y, por supuesto, todo esto sin cambiar ni una sola línea de código.

Pero PAM va más allá todavía, permitiendo al administrador del sistema construir políticas diferentes de autenticación para cada servicio.

En resumen, podrían sintetizarse las ventajas más importantes de PAM en los siguientes puntos:

- Ofrece un esquema de autenticación común y centralizado.
- Permite a los desarrolladores abstraerse de las labores de autenticación.
- Facilita el mantenimiento de las aplicaciones.

[1] A no ser que se indique lo contrario, a partir de ahora se hará referencia "PAM" y "Linux-PAM" indistintamente.

[2] Frequently Asked Questions, http://www.kernel.org/pub/linux/libs/pam/FAQ

[3] Lightweight Directory Access Protocol

- Ofrece flexibilidad y control tanto para el desarrollador como para el administrador de sistema.

2.2. Grupos de gestión

A pesar de lo que se ha explicado anteriormente, la misión de PAM no es, únicamente, comprobar que un usuario es quien dice ser —autenticación—. Su alcance es mucho mayor y pueden dividirse sus tareas en cuatro grupos independientes de gestión, cada uno de los cuáles se encarga de un aspecto diferente de los servicios restringidos.

account (cuenta) En este grupo se engloban tareas que no están relacionadas directamente con la autenticación. Algunos ejemplos son permitir/denegar el acceso en función de la hora, los recursos disponibles o, incluso, la localización. Ofrece verificación de cuentas de usuario. Por ejemplo, se encarga de determinar si el usuario tiene o no acceso al servicio, si su contraseña ha caducado, etc.

authentication (autenticación) Tareas encaminadas a comprobar que, efectivamente, el usuario es realmente quien dice ser. A menudo, cuando se habla de PAM, sólo se tiene en cuenta esta tarea, ignorando las demás. Estas tareas ofrecen incluso un sistema de credenciales que permiten al usuario ganar ciertos privilegios —fijados por el administrador—.

password (contraseña) Se encarga de mantener actualizado el elemento de autenticación asociado a cada usuario —por ejemplo, su contraseña—. Acciones como comprobar la fortaleza de una clave son típicas de este grupo.

session (sesión) En este grupo se engloban tareas que se deben llevar a cabo antes de iniciarse el servicio y después de que este finalice. Es especialmente útil para mantener registros de acceso o hacer accesible el directorio *home* del usuario.

3. Arquitectura

Hasta ahora se ha estudiado cuál es la misión de PAM y qué grupos de tareas lleva a cabo. Así que, la siguiente pregunta es: ¿cómo se organizan todas estas ideas?

La figura 1, extraída de [**?**], ilustra de forma clara su arquitectura.

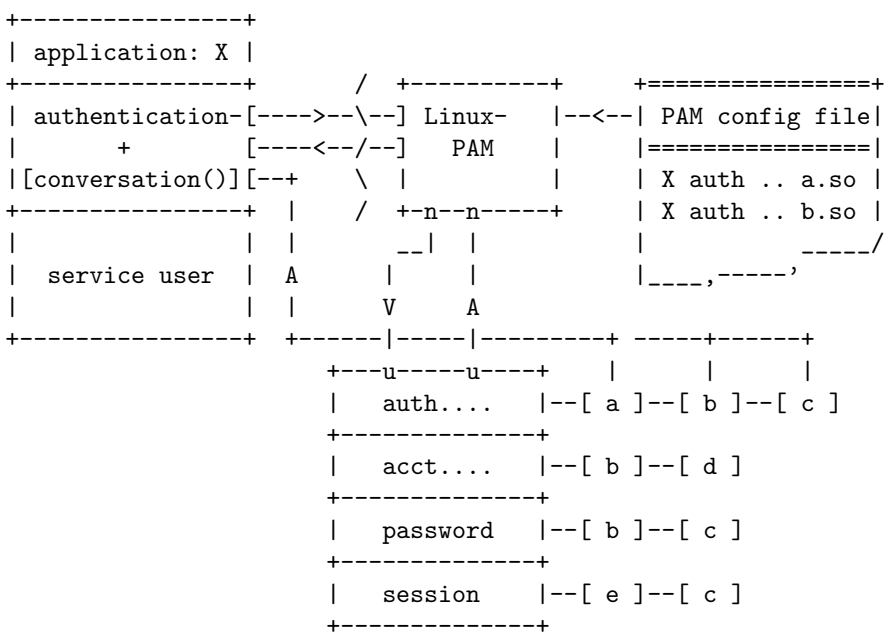

Figura 1: Arquitectura de Linux-PAM

Supóngase que una aplicación `X` quiere hacer uso de las facilidades ofrecidas por PAM. Para ello, interactúa con la biblioteca de Linux-PAM, sin tener que conocer ningún detalle acerca de como está configurado el sistema para la aplicación `X`. Será precisamente esta biblioteca quien se encargue de leer la configuración de PAM para conocer qué política de autenticación ha de aplicarse —combinando de forma conveniente una serie de módulos—.

Los módulos se colocan en una *pila* según el grupo de gestión y el orden en el que aparecen en la configuración —un módulo puede pertenecer a varios grupos—, para ser utilizados por PAM cuando corresponda. Este aspecto es tremendamente importante, ya que como se verá más adelante, el orden de los módulos en la pila va a determinar, en gran medida, el comportamiento de PAM para un servicio dado. En la figura, para la tarea de autenticación, se invocará primero al módulo `a`, luego a `b` y, finalmente, a `c`[4].

Finalmente, PAM ofrece a la aplicación una serie de funciones para llevar a cabo las diferentes tareas de cada grupo (autenticar, abrir sesión, etc.), mientras que la aplicación

[4]Pueden existir variaciones en el flujo, pero por el momento no se van a tener en cuenta

brinda a PAM una función de *conversación* destinada a intercambiar información textual. Gracias a esta función, PAM se libera de tener que preocuparse de cómo enviar/recibir información del cliente —cuadros de diálogo, intercambio en un terminal, protocolos de red, etc.—.

4. Configuración

4.1. Enfoques de la organización de la configuración

La configuración de Linux-PAM puede organizarse siguiendo dos esquemas diferentes:

- Poner toda la configuración en el fichero `/etc/pam.conf` —la más antigua—.
- Colocar la configuración de cada servicio en ficheros separados bajo el directorio `/etc/pam.d/`. Este tipo de organizaciones ha ido ganando adeptos, y proyectos como *Apache* también hacen uso de un esquema de este tipo.
- Una combinación de las dos anteriores: por ejemplo, leer primero `/etc/pam.d/` y luego `/etc/pam.conf` —como en el caso de *Red Hat Linux*.

Afortunadamente, en todos los casos se utiliza la misma sintaxis, con la salvedad de que en el caso de `/etc/pam.conf` hay que indicar el servicio al que pertenece cada directiva —con `/etc/pam.d/` esta información está implícita en el nombre del fichero de configuración—.

En aras de la simplicidad, en adelante se supondrá que se usa el enfoque combinado, si bien este es un detalle que, conceptualmente, carece de importancia.

```
$ ls /etc/pam.d/
chage  chsh  groupadd  kde-np  other   shadow  su    system-auth  xdm
chfn   cups  kde       login   passwd  sshd    sudo  useradd      xserver
```

Figura 2: Organización de la configuración en ficheros separados

4.2. Reglas

Los ficheros de configuración de PAM están compuestos por una serie de reglas a aplicar, una por línea —aunque pueden dividirse en varias usando el caracter '\'—. Se ignoran todas las líneas que comiencen por el caracter '#'.

Todas estas reglas tienen la siguiente forma:

```
servicio*  tipo  control  ruta  [argumentos]
```

A continuación, se va a profundizar en el significado y la sintaxis de cada uno de estos campos. Hay que mencionar que, exceptuando los casos de `ruta` y `argumentos` —depende del módulo—, en el resto de los campos no se hacen distinciones entre mayúsculas y minúsculas.

Pero antes, y para que lector se haga una idea de la estructura de este fichero, un pequeño ejemplo. Esta configuración pertenece al servicio `login` de *Red Hat Linux* —`/etc/pam.d/login`—.

```
#%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
```

Servicio

Este primer campo indica el nombre de la aplicación/servicio correspondiente, como por ejemplo `login`, `su` o `smtp`. Tal y como se explicó anteriormente, se utiliza sólo en `/etc/pam.conf`, ya que en el caso de la configuración mediante ficheros individuales el nombre servicio se encuentra implícito en el del fichero.

Existe un nombre de servicio reservado, `other`, que se utiliza para especificar reglas por defecto. Así, en el caso de que, por ejemplo, no hubiese reglas para `smtp`, se le aplicarían las asociadas al "servicio" `other`.

Tipo

Como se vio anteriormente, PAM puede dividir su actividad en cuatro tareas de gestión. Y `tipo` expresa, precisamente, el área al que se destina esta regla. Puede adoptar uno de estos valores: `auth`, `account`, `session` o `password`. Como ya se había mencionado con anterioridad, los módulos correspondientes a un mismo área forman una *pila*.

Control

Este campo indica a PAM qué hacer en caso de éxito/fallo del módulo de la regla en cuestión. ejecutándose en serie todos los del mismo `tipo`. El orden en la pila hay que

tenerlo muy en cuenta a la hora de determinar el valor adecuado para este campo.

Actualmente, pueden usarse dos sintaxis: la más simple consiste en una sola palabra clave, mientras que la otra —más precisa y compleja— implica el uso de corchetes y parejas `valor-acción`. Para facilitar la compresión, en primer lugar se va a estudiar la sintaxis más simple para, a continuación, explorar su alternativa compleja.

Usando la sintaxis sencilla, este campo puede tomar únicamente cuatro valores diferentes:

required Indica que es necesario que el módulo tenga éxito para que la pila también lo tenga. Si se produce un fallo, no se notifica hasta que se procesa el resto de la pila.

requisite En esencia, es igual que el anterior, con la diferencia de que en caso de fallo, el control se devuelve inmediatamente a la aplicación.

sufficient El éxito en este módulo, si no se ha producido un fallo en los procesados anteriormente en la pila, es "suficiente". Llegados a este punto, el procesamiento se detiene —ignorando incluso posibles *required* posteriores—. Un fallo no siempre resulta definitivo para la pila.

optional Por lo general, PAM ignora los módulos marcados con este indicador. Su valor será tenido en cuenta sólo en caso de que no se haya llegado a ningún valor concreto de éxito o fracaso —por ejemplo, `PAM_IGNORE`—.

La sintaxis alternativa de configuración ofrece mayor flexibilidad y funcionalidad, pero introduce cierta complejidad que en muchos casos no es deseable. Se basa en asociar parejas `valor-acción`. Así, pueden asociarse acciones con los valores devueltos por los módulos.

```
[valor1=acción1 valor2=acción2 valor3=acción3 ...]
```

Los posibles valores de $valor_i$ se encuentran documentados en el apéndice A (pág. 22). En lo que a las partes derechas se refiere, $accion_i$ puede ser un número entero positivo n o cualquiera de estos valores:

ignore El valor que devuelve este módulo no se tiene en cuenta.

bad Si se trata del primer módulo en fallar, el valor que devuelva la pila será el que devuelva este módulo.

die Equivalente al anterior, pero en este caso se devuelve el control a la aplicación de inmediato.

ok Si hasta el momento, el estado de la pila conduce a un éxito —`PAM_SUCCESS`—, el código que devuelve será sobreescrito por el de este módulo. En caso contrario, deja el estado tal y como se lo encuentra.

done Funciona del mismo modo que `ok`, con la salvedad de que, llegado a este punto, devolverá el control a la aplicación.

reset Se olvida el estado de la pila hasta el momento y se sigue con el siguiente módulo de la pila.

El valor entero positivo n que antes se mencionó, indica que deben saltarse los siguiente n módulos. Ésto permite al administrador tener, en cierto modo, control sobre el flujo de ejecución del proceso.

Para ilustrar la relación entre una y otra sintaxis de configuración, a continuación se expresan los posibles valores de la sintaxis sencilla en función de expresiones de la sintaxis compleja.

required equivale a `[success=ok newauthok_reqd=ok ignore=ignore default=bad]`.

requisite equivale a `[success=ok newauthok_reqd=ok ignore=ignore default=die]`.

sufficient equivale a `[success=done new_authok_reqd=done default=ignore]`.

optional equivale a `[success=ok new_authtok_reqd=ok default=ignore]`.

Ruta

Este campo contiene la ruta del módulo que se va a utilizar: si empieza por el caracter '/' se indica una ruta absoluta. En otro caso, será relativa a `/lib/security/`.

Argumentos

Se trata de argumentos que pueden ser pasados al módulo para su operación. Generalmente, los argumentos son específicos para cada módulo y deberían estar documentados. Si se pasara un argumento no válido, el módulo lo ignoraría, aunque debería usar *syslog* para informar del error.

En cuanto a la sintaxis, hay que señalar que si se quieren introducir espacios en blanco en un argumento, éste deberá ir encerrado entre corchetes. Si lo que se pretende hacer es pasar un corchete como parte del parámetro, habrá hacer uso del carácter de escape '\'.

```
squid auth required pam_mysql.so  user=passwd_query passwd=mada \
  db=eminence [query=select user_name from internet_service where \
    username_='%u' and password=PASSWORD('%p') and service='web_proxy']
```

En el ejemplo anterior puede observarse como se combinan distintos elementos para formar una regla bastante compleja. Nótese que los valores %u y %p son sustituidos por el *nombre de usuario* y la *contraseña* respectivamente.

5. Algunos módulos disponibles

Hasta el momento se han estudiado los fundamentos, la arquitectura y la configuración de PAM. Sin embargo, poco se ha dicho acerca de los módulos que ofrece. Esta sección está dedicada a describir, sin entrar en demasiado detalle, algunos de los módulos que vienen con la distribución oficial de PAM en su versión `0.77`.

Esto no pretende ser una guía de referencia exhaustiva, por lo que para detalles más concretos sobre estos módulos se remite al lector a la documentación del proyecto. Simplemente, se expondrá una breve descripción de cada módulo y se mostrarán cuáles son los principales argumentos que admiten.

5.1. pam_console.so

Grupo *session* y *auth*.

Este módulo permite el cambio de los permisos y de los propietarios de los ficheros indicados en `/etc/security/console.perms` cuando un usuario accede al sistema mediante una consola física y no hay ningún otro usuario registrado en el sistema. Cuando el usuario abandona la sesión, los permisos y propietarios originales se restauran.

En el caso de que varios usuarios trabajasen al mismo tiempo en la consola física, los ficheros sólo se le asignarían al primero en registrarse en el sistema, aunque esto no suele ocurrir.

Los parámetros admitidos por este módulo son:

allow_nonroot_tty Bloquea la consola y cambia los permisos incluso si el propietario del terminal no es el superusuario.

permsfile=fichero Permite especificar un fichero alternativo al `/etc/security/console.perms` del cual leer la base de datos de permisos.

fstab=fichero Permite indicar un fichero alternativo al `fstab`. El fichero `fstab` almacena información relativa a los dispositivos que se pueden montar en el sistema, indicando donde deben de montarse y el sistema de ficheros en el que se encuentra.

5.2. pam_cracklib.so

Grupo *password*

Se trata de un módulo preventivo que se encarga de notificar al usuario de la debilidad de su clave cuando ve que puede "romperse" usando un diccionario. Este módulo entra en funcionamiento cuando se establece o modifica la clave de un usuario.

Básicamente este módulo hace pasar la clave por la biblioteca `cracklib`. En caso de que pase como clave segura, intentará las siguientes comprobaciones, comparando la nueva clave con la antigua:

Palíndromo Comprueba que la nueva contraseña no sea la vieja al revés.

Cambio de mayúsculas Realiza varias combinaciones cambiando mayúsculas por minúsculas y viceversa.

Similar Comprueba que las claves se diferencian en más de `difok` caracteres.

Simple Verifica el tamaño de la clave.

Rotada Comprueba que la clave nueva no es una rotación de la antigua.

Ya usada Se asegura de que la clave no se ha usado con anterioridad. Las contraseñas ya usadas se almacenan en `/etc/security/opasswd`.

Así mismo este módulo permite un gran número de parámetros, ahora se mostraran los mas importantes:

debug Muestra información mas detallada a través de `syslog`. Esta opción no imprimirá las claves en el fichero de log.

retry=N Permite especificar el número de veces que se volverá a pedir la clave en caso de que no sea segura. Por defecto es 1.

difok=N Indica el número de caracteres que deben ser diferentes entre la clave anterior y la nueva.

minlen=N Indica el número mínimo de caracteres que debe tener una clave.

use_authok Evita que el usuario introduzca la nueva contraseña tomándola del módulo que se encuentra por delante en la pila.

5.3. pam_deny.so

Grupo *account,authentication*, *password* y *session*.

El objetivo de este módulo es producir un fallo siempre. Si este módulo es el único que se encuentra en la pila, se considerará que ésta ha fallado.

5.4. pam_env.so

Grupo *authentication*

Permite establecer las variables de entorno por defecto o sustituir los valores de las variables ya establecidas cuando un usuario se registra en el sistema. El fichero donde se definen dichas variables se encuentra en `/etc/security/pam_env.conf`.

Las cláusulas de este fichero son de la siguiente manera:

```
VARIABLE [DEFAULT=[valor]] [OVERRIDE=[valor]]
```

donde la cláusula `default` especificará el valor a tomar por defecto y `override` el valor por el que será sustituido el contenido de la variable.

Este módulo admite cuatro parámetros:

debug Muestra información mas detallada sobre el módulo en syslog.

conffile=fichero Especifica el fichero de configuración alternativo.

envfile=fichero Especifica el fichero alternativo a `/etc/enviroment` que contiene las variables de entorno establecidas para el sistema.

readenv=0/1 Especifica si se lee o no el fichero con las variables de entorno. Por defecto sí se lee.

5.5. pam_limits.so

Grupo *session*

Controla los límites impuestos en el fichero `/etc/security/limits.conf` a los recursos disponibles. Las limitaciones se pueden aplicar a un usuario, a un grupo o a todos los usuarios del sistema. Los recursos que se pueden limitar desde este módulo van desde el máximo tiempo de CPU asignable, hasta el número máximo de ficheros que puede tener bloqueados simultáneamente.

Este módulo permite los siguiente parámetros:

debug Muestra información añadida a través de syslog.

conf=fichero Permite especificar un fichero de configuración alternativo.

change_uid Permite cambiar el `uid` real para los usuarios que se han establecido límites. Esta opción se usa principalmente en sistemas en los que al poner un límite de 0 procesos el usuario no puede ni abrir la sesión.

utmp_early Corrige el problema generado por algunas aplicaciones que intentan crear `utmp` para el usuario antes de que este entre en el sistema.

La sintaxis del fichero de configuración es la siguiente:

```
<dominio> <tipo> <elemento> <valor>
```

donde `<dominio>` es el nombre de usuario, nombre de grupo, el comodín '*' o el comodín '%'. `<tipo>` indicara si se trata de un limite duro (`hard`) débil (`soft`) o los dos tipos simultáneamente (`-`). Finalmente, el parámetro `<elemento>` se puede sustituirse por varios valores que indican el recurso que se esta limitando.

5.6. pam_nologin.so

Grupo *account* y *authentication*

Este módulo sólo deja entrar a los usuarios del sistema si el fichero `/etc/nologin` no existe. En caso contrario, sólo el superusuario puede ingresar en el sistema. Al resto de los usuarios se les mostrará el contenido de dicho archivo.

`Pam_nologin` permite dos parámetros:

successok Devuelve `PAM_SUCCESS` en vez de `PAM_IGNORE` en caso de no encontrar el fichero `/etc/nologin`.

file=fichero Permite indicar un fichero alternativo a `/etc/nologin`.

5.7. pam_permit.so

Grupo *account*,*authentication*, *password* y *session*.

Pam_permit funciona justamente al contrario del módulo pam_deny.so. Para cada llamada al módulo este devuelve un acierto , PAM_SUCCESS. Por esta razón este módulo es muy inseguro y debe de ser usado con precaución extrema.

5.8. pam_rootok.so

Grupo *authentication*

Pam_rootok.so devuelve un acierto siempre que el identificador real del usuario sea 0 —el usuario root—. Con ello se consigue que el usuario root no necesite introducir la clave para acceder a los servicios asociados a este módulo. Al usar pam_rootok hay que tener mucho cuidado ya que plantea un grave problema de seguridad. Si asociaramos pam_rootok a un proceso que se arranca con el sistema, como por ejemplo login, el uid real sera 0 y por lo tanto acertara siempre, independientemente de que intentemos registrarnos con un usuario con uid distinto de 0.

5.9. pam_securetty.so

Grupo *authentication*

Sirve para limitar las consolas en las que se puede autenticar el usuario root. El fichero /etc/securetty contiene una lista de consolas seguras.

5.10. pam_stack.so

Grupo *account*,*authentication*, *password* y *session*.

Este módulo permite la asociación en pila de varios módulos PAM devolviendo un acierto, en caso de que toda la pila devuelva un acierto. En caso de que uno de los módulos de la pila devuelva un error, pam_stack devolverá el código de error devuelto por el módulo que ha fallado.

5.11. pam_wheel.so

Grupo *authentication* Limita la autenticación como root a los usuarios del grupo wheel.

Los parámetros admitidos por pam_wheel son los siguientes:

debug Muestra información añadida a través del `syslog`.

use_uid Modifica el comportamiento del módulo usando el el `uid` del proceso y no el nombre de usuario de `getlogin`.

trust Con esta opción se consigue que los usuarios que pertenecen al grupo `wheel` cambiarse por el usuario `root` sin usar la clave. Debe de usarse con mucho cuidado.

deny Este parámetro hace que el módulo se comporte al revés denegando a los usuarios del grupo `wheel` la posibilidad de convertirse en `root`.

group=XXXX Permite especificar los grupos a los que se les permite la autenticación.

5.12. pam_xauth.so

Grupo *session*

Se encarga de redireccionar las "cookies" de autenticación de un usuario a otro en el sistema *X-Window*. Esto se usa para que, cuando un usuario se haga pasar por otro mediante el uso del comando `su` o alguno de sus derivados, pueda seguir lanzando aplicaciones como el nuevo usuario sin necesidad de salir y volver a entrar en el entorno *X-Window*.

6. Ejemplos de configuración

Ahora que ya se han estudiado tanto los fundamentos de configuración como el cometido de los módulos más destacados de Linux-PAM, parece buena idea fijar la atención en una serie de ejemplos reales, con el fin de ilustrar los conceptos vistos con anterioridad.

Para ello se comentará a continuación la configuración que da *Red Hat Linux* a algunos de sus servicios. Concretamente, `login`, `passwd`, `su` y `other`.

Sin embargo, en primer lugar se va a tener en cuenta un servicio "artificial", `system-auth`. Éste engloba políticas comunes a muchos otros servicios que lo incluyen en su configuración haciendo uso del módulo `pam_stack`. La ventaja de este esquema es que, en caso de querer cambiar el comportamiento común de varios servicios, tan sólo es necesario modificar `system-auth`.

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.

### auth ###

# Inicializa las variables de entorno definidas en /etc/security/pam_env.conf.
auth        required      /lib/security/$ISA/pam_env.so

# Autentica al usuario al estilo 'unix' tradicional (/etc/passwd). Se detiene
# aquí si tiene éxito.
# nullok: permite contraseñas en blanco.
# likeauth: pam_sm_setcred() devuelve lo mismo que pam_sm_authenticate()
auth        sufficient    /lib/security/$ISA/pam_unix.so likeauth nullok

# Si se llega hasta aquí sin éxito, se falla.
auth        required      /lib/security/$ISA/pam_deny.so

### account ###

# Basándose en /etc/shadow, determina el estado de la cuenta de usuario (si ha
# expirado, debe cambiar la contraseña, etc.).
account     required      /lib/security/$ISA/pam_unix.so

### password ###

# Comprueba que la clave no es vulnerable a un ataque simple basado en
# diccionario.
# retry: 3 intentos para cambiar.
# type: modifica el mensaje. Para type=XXX, sería 'New XXX password: '.
password    required      /lib/security/$ISA/pam_cracklib.so retry=3 type=

# Actualiza la contraseña.
# nullok: permite cambiar una contraseña en blanco. Sin este parámetro,
#         una clave en blanco se interpreta como señal de que la cuenta se
#         encuentra desactivada.
```

```
# use_authtok: toma como nueva contraseña la obtenida por el módulo anterior
#              (cracklib en este caso).
# md5: usa la función MD5 para cifrar la contraseña.
password    sufficient    /lib/security/$ISA/pam_unix.so nullok use_authtok md5 shadow

# Si se llega hasta aquí sin éxito, se falla.
password    required      /lib/security/$ISA/pam_deny.so

### session ###

# Limita los recursos para la sesión del usuario en base al contenido del
# archivo /etc/security/limits.conf.
session     required      /lib/security/$ISA/pam_limits.so

# Registra el acceso al servicio en syslog, tanto al principio como al final
# de la sesión.
session     required      /lib/security/$ISA/pam_unix.so
```

Sólo un apunte más, los comentarios de los ficheros que se presentan en esta sección han sido añadido por los autores de este documento, por lo que es probable que introduzcan algunas imprecisiones.

6.1. login

El programa `login` ofrece a los usuarios la posibilidad de abrir una sesión en el sistema. Éste es uno de los puntos de acceso más sensibles y que, por lo general, ofrece un mayor riesgo para la integridad del sistema. Así pues, la configuración de Linux-PAM en este —como en todos los servicios, a decir verdad— debe estar especialmente cuidada.

Por lo general, `login` pide un nombre de usuario y una contraseña. Después de llevar a cabo una serie de comprobaciones e inicializaciones, al usuario se le ofrece un intérprete de comandos —`shell`—.

La configuración de este servicio en *Red Hat Linux* hace uso intensivo del servicio "artificial" `system-auth`, siendo idénticas las pilas `account` y `password`. Las diferencias radican, principalmente, en que al superusuario sólo se le permite acceder desde terminales físicas —`tty`—, se deshabilita la entrada si existe el fichero `/etc/nologin` y se realizan cambios de permisos/propietario según lo indicado en `/etc/security/console.perms`.

```
#%PAM-1.0

### auth ###

# Deshabilita el 'login' para el root exceptuando los tty's.
auth       required pam_securetty.so

# Procesa los módulos 'auth' del servicio 'system-auth'.
```

```
auth       required pam_stack.so service=system-auth

# Deshabilita la entrada de cualquier usuario que no sea el 'root' cuando el
# fichero /etc/nologin existe.
auth       required pam_nologin.so

### account ###

# Procesa los módulos 'account' del servicio 'system-auth'.
account    required pam_stack.so service=system-auth

### password ###

# Procesa los módulos 'password' del servicio 'system-auth'.
password   required pam_stack.so service=system-auth

### session ###

# Procesa los módulos 'session' del servicio 'system-auth'.
session    required pam_stack.so service=system-auth

# Cambia permisos y propietarios de ciertos ficheros según se indica en
# /etc/security/console.perms si el acceso se realiza por medio de una
# consola 'física'. Los permisos/propietarios originales se restauran al
# salir.
session    optional pam_console.so
```

6.2. passwd

`passwd` es la utilidad que permite modificar la contraseña de las cuentas asociadas a los usuarios y a los grupos. Actuando como un usuario normal del sistema solamente podrá modificar la propia contraseña, mientras que el superusuario podrá modificar la contraseña de cualquier cuenta. Así mismo, solamente el administrador de un grupo podrá cambiar la contraseña del grupo.

Además de permitir el cambio de la contraseña, `passwd` permite modificar otros parámetros asociados a una cuenta de usuario como son el nombre, el `shell` de inicio o la fecha de caducidad de la clave.

```
#%PAM-1.0

### auth ###

# Procesa los módulos 'authentication' del servicio 'system-auth'
auth       required pam_stack.so service=system-auth

### account ###

# Procesa los módulos 'account' del servicio 'system-auth'
account    required pam_stack.so service=system-auth
```

```
### password ###

# Procesa los módulos 'password' del servicio 'system-auth'
password   required pam_stack.so service=system-auth
```

6.3. su

`Su` es una aplicación que permite convertirse en otro usuario una vez que tenemos una sesión ya abierta. Cuando se invoca sin parámetros el `su` actúa como si se quisiera convertir al superusuario, `root`. En caso contrario `su` intentara convertirse en el usuario pasado por parámetro.

A continuación se mostrara el fichero de configuración del `pam` para la utilidad `su` suministrado en `RedHat`.

```
#%PAM-1.0

### auth ###

# Permite que el usuario root use la utilidad su sin suministrar
# ninguna contraseña.
auth       sufficient   /lib/security/$ISA/pam_rootok.so

# Descomentar la siguiente linea para indicar que los usuarios
# pertenecientes al grupo wheel sean usuarios en los que se confía y
# puedan realizar un 'su' sin suministrar contraseña.
#auth       sufficient   /lib/security/$ISA/pam_wheel.so trust use_uid

# Descomentar la siguiente linea para indicar que los usuarios
# pertenecientes al grupo wheel son los únicos usuarios a los que se
# les permite usar la utilidad 'su'
#auth       required     /lib/security/$ISA/pam_wheel.so use_uid

# Procesa los módulos 'authentication' del servicio 'system-auth'.
auth       required     /lib/security/$ISA/pam_stack.so service=system-auth

### account ###

# Procesa los módulos 'account' del servicio 'system-auth'.
account    required     /lib/security/$ISA/pam_stack.so service=system-auth

### password ###

# Procesa los módulos 'password' del servicio 'system-auth'.
password   required     /lib/security/$ISA/pam_stack.so service=system-auth

### session ###

# Procesa los módulos 'session' del servicio 'system-auth'.
session    required     /lib/security/$ISA/pam_stack.so service=system-auth

# Carga el modulo xauth como opcional, permitiendo que el usuario al
# que se ha cambiado mediante el su pueda seguir lanzando aplicaciones X-Window
```

```
# en la sesión abierta por el usuario original.
session    optional     /lib/security/$ISA/pam_xauth.so
```

6.4. other

Tal y como se explicó con anterioridad, cuando PAM no encuentra un fichero de configuración para un servicio en particular, aplica las reglas del servicio especial `other`. Por tanto, puede decirse que se trata de la política por defecto.

Por motivos de seguridad, es aconsejable que este servicio deniegue cualquier acceso. Así, si Linux-PAM no encuentra configuración para un servicio, simplemente denegará el acceso, minimizando posibles amenazas.

Con esta premisa, la configuración resulta tremendamente simple, tal y como puede apreciarse en las siguientes líneas.

```
#%PAM-1.0

# Por defecto, se deniega el acceso a cualquiera
auth     required       /lib/security/$ISA/pam_deny.so
account  required       /lib/security/$ISA/pam_deny.so
password required       /lib/security/$ISA/pam_deny.so
session  required       /lib/security/$ISA/pam_deny.so
```

A. Valores devueltos por los módulos de PAM

Cada módulo de `PAM` devuelve un valor al ser invocado. Éste indica si se ha producido algún fallo o si, por el contrario, todo ha ido bien.

PAM permite especificar comportamientos para cada uno de estos posibles valores en el fichero de configuración (ver 4, pág. 6). Para ello, se utiliza la sintaxis de pares `valor=acción`, colocando en la parte izquierda de la asignación la la etiqueta del valor —cualquiera de los valores descritos a continuación sin el prefijo `PAM_` y escrito en minúscula; por ejemplo, `auth_err`—.

A.1. authentication

PAM_AUTH_ERR El usuario no se autenticó.

PAM_AUTHINFO_UNAVAIL El módulo no fue capaz de acceder a la información de autenticación. Esto se debe generalmente a un fallo hardware o de acceso a la red.

PAM_CRED_ERR Este valor se devuelve cuando el módulo no pudo establecer las credenciales del usuario.

PAM_CRED_EXPIRED Indica que las credenciales del usuario han caducado.

PAM_CRED_INSUFFICIENT Por alguna razón, la aplicación no tiene suficientes credenciales para autenticar al usuario.

PAM_CRED_UNAVAIL El módulo no pudo obtener las credenciales de los usuarios.

PAM_MAXTRIES Los módulos devuelven este valor cuando han alcanzado el número máximo de reintentos de autenticar al usuario.

PAM_USER_UNKNOWN El nombre de usuario es desconocido para el sistema de 'authentication'.

PAM_SUCCESS Este valor se devuelve cuando el módulo ha tenido éxito.

A.2. account

PAM_ACCT_EXPIRED La cuenta del usuario ha caducado y ya no puede volver a acceder con dicha cuenta.

PAM_AUTH_ERR El usuario no se autenticó.

PAM_AUTHTOKEN_REQD El token de autenticación ha caducado. Antes de volver a llamar a esta función se debería pedir un nuevo token.

PAM_SUCCESS Este valor se devuelve cuando el módulo ha tenido éxito.

PAM_USER_UNKNOWN El nombre de usuario es desconocido para el sistema de 'account'.

A.3. password

PAM_AUTHOK_DISABLE_AGING El token de autenticación ha sido desactivado.

PAM_AUTHTOK_ERR El módulo fue incapaz de obtener un nuevo token de autenticación.

PAM_AUTHTOK_RECOVERY_ERR No se consiguió leer el anterior token de autenticación.

PAM_AUTHTOK_LOCK_BUSY El token de autenticación estaba bloqueado cuando se intentó cambiar.

PAM_PERM_DENIED Permiso denegado.

PAM_SUCCESS Este valor se devuelve cuando el módulo ha tenido éxito.

PAM_TRY_AGAIN Las comprobaciones anteriores fallaron. Se pide que se vuelva a hacer la misma llamada.

PAM_USER_UNKNOWN El nombre de usuario es desconocido para el sistema de 'password'.

A.4. session

PAM_SESSION_ERR Se ha producido un fallo al intentar abrir o cerrar la sesión.

PAM_SUCCESS Este valor se devuelve cuando el módulo ha tenido éxito.

B. Referencias

Además de las páginas del manual, la documentación adjunta a la distribución oficial y algo de código fuente, el presente texto ha tomado una importante cantidad de información de los documentos que se listan a continuación. Así que, cualquiera que desee profundizar en el tema, debería referirse a ellos.

[1] *A Linux-PAM page* http://www.kernel.org/pub/linux/libs/pam/

[2] Andrew G. Morgan, *The Linux-PAM Administrators' Guide*
http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam.html
1996

[3] Andrew G. Morgan, *The Linux-PAM Application Developers' Guide*
http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam_appl.html
2001

[4] Andrew G. Morgan, *The Linux-PAM Module Writers' Guide*
http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam_modules.html
2002

[5] V. Samar y R. Schemers, *RFC 86.0: Unified login with pluggable authentication modules (PAM)*
Open Software Foundation, 1995

[6] Dag-Erling Smørgrav, *Pluggable Authentication Modules*
http://www.freebsd.org/doc/en_US.ISO8859-1/articles/pam/index.html Networks Associates Technology, Inc., 2003

```
$ ls /lib/security
apacheconf      printconf                   redhat-config-services
authconfig      printconf-gui               redhat-config-soundcard
authconfig-gtk  printconf-tui               redhat-config-time
bindconf        printtool                   redhat-config-users
chfn            reboot                      redhat-config-xfree86
chsh            redhat-cdinstall-helper     redhat-install-packages
cups            redhat-config-bind          redhat-logviewer
dateconfig      redhat-config-date          rhn_register
ethereal        redhat-config-httpd         samba
gdm             redhat-config-keyboard      screen
...
```

Figura 3: Lista parcial de módulos PAM en `/lib/security/`