Francisco José Ribadas Pena
ribadas@uvigo.es
Departamento de Informática
Universidade de Vigo

Gestión de usuarios y autenticación en GNU/Linux.
Módulos de autenticación PAM

Ferramentas de seguridade en GNU/Linux
Curso de Extensión Universitaria

21 de xuño de 2010

Indice

- Autenticación
- Autenticación de usuarios en GNU/Linux
- *Pluggable Authentication Modules* (PAM)
- Funcionamiento de PAM
- Configuración de PAM
- Módulos PAM
- Ejemplos de uso

Autenticación

- Definición:
  - *Mecanismo/proceso que permita identificar a las entidades (elementos activos del sistema [usuarios]) que intentan acceder a los objetos (elementos pasivos [ficheros, equipos, datos].*
  - **Objetivo:**
    - *Autenticar* = asegurar que una entidad[usuario] es quien dice ser
- Mecanismos generales de autenticación:
  - en base a *algo que sabe* el usuario
    - Ej.: claves, passwords y similares
  - en base a *algo que posee* el usuario
    - Ej.: DNIe, tarjetas acceso, tarjetas inteligentes, ...
  - en base a alguna *característica física* del usuario
    - Ej.: medidas biométricas (huellas, iris, voz, etc)

Autenticación de usuarios en GNU/Linux (I)

Esquema clásico

- Cada usuario se identifica por un UID (*user id*) al que se asocia un nombre (*login*) y una clave secreta (*password*)
- Asociaciones almacenadas en el fichero ***/etc/passwd***
- Una línea por usuario con campos separados por ":"

  login:password:uid:gid:nombre:home:shell
- Claves almacenadas están cifradas con CRYPT
  - A partir de: 8 bytes password claro + **salt** aletorio
  - Varios cifrados DES en cascada [cifrado en único sentido]
  - /etc/passwd almacena **salt** en claro + resultado CRYPT
- Proceso autenticación (login)
  - Password introducido + **salt** se procesa con CRYPT
  - Resultado obtenido es comparado con almacenado
    - si coinciden -> acceso permitido
- Fichero */etc/passwd* debe ser accesible en lectura para todas las aplicaciones que lo usen en la autenticación

Autenticación de usuarios en GNU/Linux (II)

Problemas esquema clásico

- Info. de usuarios en /etc/passwd (incluidos passwords cifrados) debe ser accesibles a todo el mundo

  => son posibles ataques de diccionario (fuerza bruta)

  - Solución: **shadow password**
    - Almacena la versión cifrada del password en el archivo */etc/shadow* que sólo puede leer el administrador (*root*)
    - Añade info. adicional: envejecimiento, caducidad, etc
- Passwords con CRYPT limitados a 8 caracteres
  - Solución: usar *hash MD5*
    - Hash criptográfica, genera “resúmenes únicos” de 128 bits para tamaños de entrada variables (ilimitado en teoría).
- **Problema:** estos cambios requerirían “modificar” y recompilar las aplicaciones que hacían uso del esquema de autenticación clásico.

Pluggable Authentication Modules (I)

- Propuesto por SUN en 1995 (RFC 86.9)
- Proporciona un interfaz modular entre las aplicaciones de usuario y los métodos de autenticación
  - conjunto de librerías de *carga dinámica*
  - independiza aplicaciones del método de autenticación sin necesidad de recompilar
  - permite integrar aplicaciones Unix clásicas con nuevos esquemas de autenticación (biometría, tarjetas inteligentes,...)
- Objetivos de diseño
  - Dar mayor libertad al administrador para elegir el mecanismo de autenticación
  - Configuración de la autenticación a nivel de aplicación
  - Múltiples mecanismos de autentic. para cada aplicación
- **Premisa:** las aplicaciones/servicios no deben cambiar cuando se cambia el mecanismo de autenticación

Pluggable Authentication Modules (II)

- Idea base:
  - Separar las acciones típicas de autenticación
  - Delegarlas en el susbsistema PAM (*libpam*)
  - Descomponer tareas de autenticación en “5 pasos”
    - *pam_autenticate()*
    - *pam_acct_mgmt()*
    - *pam_secrted()*
    - *pam_open_session()*
    - *pam_close_session()*
    - *pam_chauthtok()*
  - Componentes software (librerías) intercambiables se encargan de implementar esas tareas
  - Controlado por ficheros de configuración

Pluggable Authentication Modules (III)

Beneficios del uso de PAM:

- Políticas de configuración flexibles
  - Autenticación para cada aplicación/servicio
  - Posibilidad de definir un mecanismo de autenticación por defecto
  - Posibilidad de requerir múltiples autorizaciones
- Facilidad de uso para el usuario final
  - Evitar repetición de passwords en diferentes servicios
  - Solicitar múltiples passwords sin lanzar múltiples comandos
  - Posibilidad de parametrizar (pasar argumentos) a los componentes des sistema de autenticación
- Posibilidad de implementar políticas específicas de una máquina/sistema sin tener que cambiar los servicios

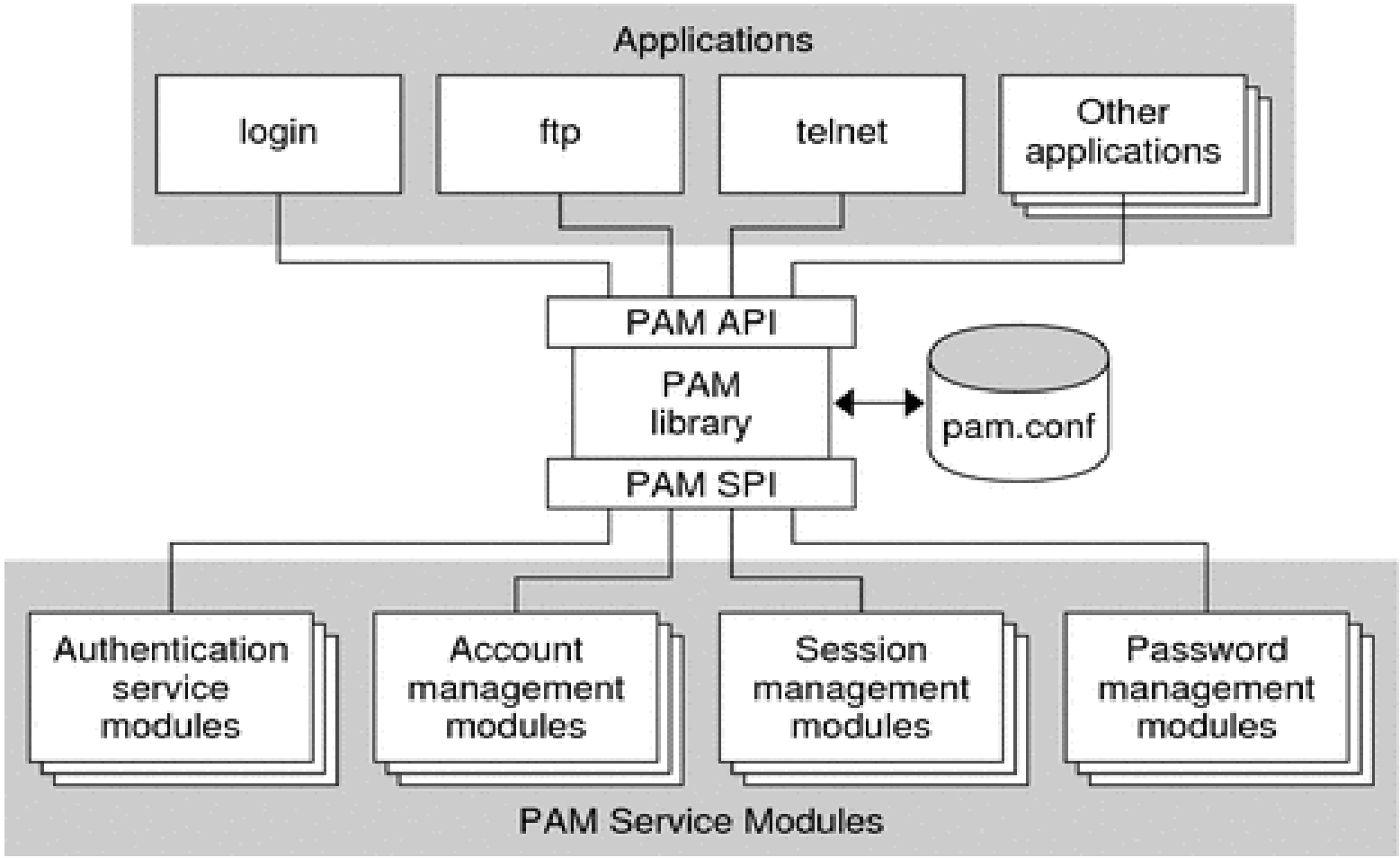
Funcionamiento de PAM (I)

Idea de funcionamiento

- El API PAM descompone las tareas de autenticación en cuatro grupos independientes
  - autenticación/obtención de credenciales
  - gestión/verificación de cuentas de usuario
  - gestión de sesiones
  - actualización info. de autenticación (passwords)
- Para cada servicio/aplicación que haga uso de PAM se deberá configurar el modo en que funcionará cada etapa del proceso de autenticación
  - especificando el módulo encargado de gestionarlo
- El servicio/aplicación concreta es ajeno a esos "*pasos intermedios*"
  - simplemente recibe una respuesta final afirmativa o positiva (autoriza/desautoriza al usuario)

Arquitectura de PAM

- Componentes: aplicaciones + módulos autenticación



Funcionamiento de PAM (III)

Tipos de módulos

- de autenticación (*auth*): autentican al usuario mediante un determinado método (solicitud de clave, tarjeta aceso, control biométrico,...) y le asignan una serie de credenciales/privilegios
  - invocados cuando la aplicación llama a la función *pam_authenticate()* ó *pam_secretd()*

- de verificación de cuentas (*account*): realizan comprobaciones relativas al servicio concreto
  - una vez establecido que el usuario es quien dice ser ...
    - ¿usuario tiene permiso para usar el servicio?
    - ¿ha expirado su clave?
    - ¿el servicio está permitido?
  - invocados cuando la aplicación llama a la función *pam_acct_mgmt()*

Funcionamiento de PAM (III)

Tipos de módulos

- de gestión de sesiones (*session*): determinan las acciones que se ejecutarán antes y/o después del acceso del usuario
  - invocados cuando la aplicación llama a la función *pam_open_session()* ó *pam_close_session()*

- de actualización (*password*): empleandos para actualizar la información que usa el mecanismo de autenticación
  - invocados cuando la aplicación llama a la función *pam_chauthtok()*

Funcionamiento de PAM (IV)

Configuración

- Controlada por fichero/s: para cada servicio/aplicación se especifican los módulos encargados de cada tarea
- Esquema original: fichero */etc/pam.conf*
  - Fichero único: asocia servicios con módulos
  - Sintaxis: (1 o más líneas por servicio)

    servicio tipo control módulo argumentos

- Esquema actual: directorio */etc/pam.d/**
  - Un fichero para cada servicio
  - Sintaxis:

    tipo control módulo argumentos

Configuración PAM (I)

- **Idea básica:** pila de módulos
  - lista de acciones a realizar antes de que usuario llegue a acceder al servicio/aplicación
  - el **orden** es muy importante
- Elementos de los ficheros */etc/pam.d*
  - **tipo:** especifica el tipo de módulo (*auth, account, password, session*)
  - **control:** especifica la acción que se toma en base al resultado (éxito/fracaso) del módulo correspondiente
  - **modulo:** path del módulo
    - path absoluto (empieza por /)
    - path relativo (módulos contenidos en /lib/security)
  - **argumentos:** parámetros concretos que se pasarán al módulo (específicos de cada módulo)

Configuración PAM (II)

Acciones (campo “control”)

- Sintaxis simplificada (4 acciones posibles)
  - **requisite:** debe verificarse
    - si falla, se envía **fallo inmediatamente** a la aplicación
    - no se ejecuta resto de módulos del mismo tipo en la lista
  - **required:** debe verificarse
    - si falla, se enviará fallo una vez completada la pila
    - se ejecutan el resto de módulos
  - **sufficient:** debe verificarse
    - si se verifica, se enviará **correcto inmediatamente**, salvo que exista un fallo previo (*required* fallido)
      - la ejecución correcta del módulo satisface los requisitos de autenticación
    - si falla, se ignora el fallo
  - **optional:** puede verificarse o no
    - su resultado sólo es tenido en consideración si es el único módulo en la lista

Configuración PAM (II)

Acciones (campo “control”)

- Sintaxis extendida (mayor control)
  - Cada “acción” es un conjunto de pares valor+acción

  [valor=accion, valor=accion, ..., valor=accion]

    - acciones a realizar en función del valor devuelto por el módulo llamado
  - Valores de retorno:

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

Configuración PAM (III)

- Aciones:
  - *nº entero K*: ignorar siguientes *K* módulos del mismo tipo
  - *ignore:* valor de retorno se omite
  - *ok:* si es correcto, devolver correcto después de ejecutar el resto de módulos de ese tipo
  - *done*: si es correcto, devolver correcto inmediatamente
  - *bad:* si es fallo, devolver fallo después de ejecutar el resto de módulos de ese tipo
  - *die:* si es fallo, devolver fallo inmediatamente
  - *reset:* desecha el estado actual y comienza de nuevo en el siguiente módulo
- Equivalencias

Simple	**Extendida**
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]

Módulos PAM (I)

Módulo *pam_unix.so*

- Tipo: *auth, account, password, session*
- Implementa el mecanismo de autenticación por passwords clásico de UNIX
  - Consulta */etc/password* + */etc/shadow*
- Argumentos propios:
  - *auth*: debug, audit, user_first_pass, try_first_pass, nullok, nodelay, noreap
  - *account*: debug, audit,
  - *password*: debug, audit, nullook, not_set_pass, use_authtok, try_fisrt_pass, use_first_pass, md5, bigcrypt, shadow, nis, remember=*n*

Módulo *pam_cracklib.so*

- Tipo: *password*
- Comprueba la fortaleza de las nuevas contraseñas introducidas.
- Usa diccionario para evitar contraseñas débiles
  - */etc/lib/cracklib_dict*
- Argumentos propios:
  - *password*: debug, type=XXX, retry=*n*, difok=*n*, minlen=*n*, dcredit=*n*, ucredit=*n*, lcredit=*n*, ocredit=*n*, use_authtok
- Módulo similar: *pam-passwdqc.so*

Módulos PAM (III)

Módulo *pam_env.so*

- Tipo: *auth*
- Permite asignar valores a variables de entorno

Módulo *pam_group.so*

- Tipo: *auth*
- Establece el grupo del usuario que accede al sistema

Módulo *pam_lastlog.so*

- Tipo: *session*
- Anota en el fichero */var/log/lastlog* el acceso del usuario y le informa de su último acceso (siempre se satisface)

Módulos PAM (IV)

Módulo *pam_limits.so*

- Tipo: *session*
- Permite establecer límites a los recursos del sistema disponibles para un usuario
- Configurado en */etc/security/limits.conf*

Módulo *pam_mkhomedir.so*

- Tipo: *session*
- Crea el directorio *home* del usuario si no existía
- Argumentos propios:
  - *skel*: directorio home base a copiar (*/etc/skel*)
  - *umask*: máscara de permisos

Módulo *pam_access.so*

- Tipo: *account*
- Comprueba si el usuario puede acceder desde una posición remota (*/etc/security/access*)

Módulos PAM (IV)

Módulo *pam_chroot.so*

- Tipo: *auth, account, session*
- Convierte el directorio del usuario en el directorio raíz (/) [entorno *chroot*]
  - Útil para limitar las acciones de los usuarios (no “ven” los directorios del sistema)

Módulo *pam_nologin.so*

- Tipo: *auth, account*
- Si existe el fichero */etc/nologin* sólo permite acceso al administrador (root)

Módulo *pam_rootok.so*

- Tipo: *auth*
- Permite acceso al administrador (*root*) sin tener que introducir su contraseña (comando *su*)

Módulos PAM (V)

Módulo *pam_wheel.so*

- Tipo: *auth, account*
- Solo permite acceso como administrador (*root*) a los miembros de un grupo concreto

Módulo *pam_deny.so*

- Tipo: *auth, account, session, password*
- Devuelve siempre fallo.
- Util para organizar pila de módulos (acción por defecto)

Módulo *pam_permit.so*

- Tipo: *auth, account, session, password*
- Devuelve siempre correcto
- Util para organizar pila de módulos (acción por defecto)

Módulos PAM (VI)

Módulo *pam_mysql.so*

- Tipo: *auth, account, session, password*
- Soporta la autenticación contra los datos almacenados en una BD *mysql*
- Configuración en */etc/pam_mysql.conf*

Módulo *pam_ldap.so*

- Tipo: *auth, account, session, password*
- Soporta la autenticación contra los datos almacenados en un directorio *LDAP*
- Configuración en */etc/pam_ldap.conf*

Ejemplos de uso (I)

Nota: Estructura */etc/pam.d* en Debian/Ubuntu

- Se definen pilas por defecto para *auth*, *account*, *password*, *session* que son incluidas en los ficheros PAM de cada uno de los servicios.
  - *common-account, common-auth,common-password, common-session*
- Ejemplo: */etc/pam.d/common-auth*

```
# /etc/pam.d/common-auth - authentication settings common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of the authentication modules that define
# the central authentication scheme for use on the system
# (e.g., /etc/shadow, LDAP, Kerberos, etc.).  The default is to use the
# traditional Unix authentication mechanisms.
#
auth    required        pam_unix.so nullok_secure
```

Ejemplos de uso (II)

- Ejemplo: */etc/pam.d/login*

```
# The PAM configuration file for the Shadow `login' service
auth       requisite  pam_securetty.so
auth       requisite  pam_nologin.so
session    required   pam_env.so readenv=1
session    required   pam_env.so readenv=1 envfile=/etc/default/locale

# Standard Un*x authentication.
@include common-auth
auth       optional   pam_group.so
session    required   pam_limits.so
session    optional   pam_motd.so
session    optional   pam_mail.so standard

# Standard Un*x account and session
@include common-account
@include common-session
@include common-password
```

Ejemplos de uso (III)

- Ejemplo: */etc/pam.d/ssh*

```
# PAM configuration for the Secure Shell service
auth       required     pam_env.so # [1]
auth       required     pam_env.so envfile=/etc/default/locale

# Standard Un*x authentication.
@include common-auth
account    required     pam_nologin.so

# Standard Un*x authorization.
@include common-account

# Standard Un*x session setup and teardown.
@include common-session
session    optional     pam_motd.so # [1]
session    optional     pam_mail.so standard noenv # [1]
session    required     pam_limits.so

# Standard Un*x password updating.
@include common-password
```

NSS: Name Service Switch

- Servicio complemento a PAM
- Unifica la gestión de la traducción/mapeo entre IDs numéricos (*uid, gid, dir. ip*) y nombres (*login, grupo, nombre dominio*)
- Unifica/reemplaza parte de la información gestionada por los ficheros de configuración típicos de Unix (*/etc/passwd, /etc/group, /etc/hosts, ...*)
- Configuración: */etc/nsswitch.conf*

```
passwd:         files db ldap
group:          files db ldap
shadow:         file db ldap

hosts:          files dns
networks:       files

protocols:      db files
services:       db files
ethers:         db files

netgroup:       nis
```

Enlaces y referencias

- SEGURIDAD EN UNIX Y REDES Versión 2.1.
  - http://www.rediris.es/cert/doc/unixsec/
- Linux PAM
  - http://www.kernel.org/pub/linux/libs/pam/
- The Linux-PAM System Administrators' Guide
  - http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/Linux-PAM_SAG.html

  (Lista de módulos+parametros en Sección 6)
- Sun PAM
  - http://www.sun.com/software/solaris/pam/