📁 LPIC-2 / 💻 Examen 210 - Gestión de Clientes de Red

210.2 Autenticación PAM

Teoría

PAM (Pluggable Authentication Modules), o Módulos de Autenticación Conectables, es una infraestructura en Linux que permite a los desarrolladores de aplicaciones escribir software que requiere autenticación sin necesidad de conocer los detalles del mecanismo de autenticación subyacente. La configuración de cómo se realiza la autenticación se define externamente en archivos de configuración.

Concepto Principal: Las aplicaciones (ej: `login`, `sshd`, `su`, `passwd`) hablan con la librería PAM (`libpam`). La librería PAM lee archivos de configuración para saber qué **módulos** ejecutar y en qué orden para realizar las tareas de autenticación, autorización, gestión de cuentas y sesiones.

Grupos de Gestión (Tipos) en PAM:

Cada tarea que PAM puede realizar se divide en uno de los cuatro grupos de gestión. En los archivos de configuración, se especifica el `type` para cada regla/línea.

1. **`auth` (Authentication):** Verifica la identidad de un usuario. Esto puede implicar pedir una contraseña, verificar una clave, usar un token, etc. Los módulos `auth` pueden establecer credenciales de usuario y otorgar/denegar acceso inicial.
2. **`account` (Account Management):** Verifica si las credenciales proporcionadas por el usuario son válidas *en este momento*. Esto incluye verificar expiración de cuentas, restricciones de acceso por hora o terminal, límites de logins simultáneos, etc.
3. **`password` (Password Management):** Se utiliza para tareas relacionadas con el cambio de contraseñas. Los módulos `password` pueden verificar la calidad de la nueva contraseña, actualizar la base de datos de contraseñas, mantener historial de contraseñas, etc.
4. **`session` (Session Management):** Realiza tareas antes y después de que se haya otorgado acceso al usuario y se haya iniciado o cerrado una sesión. Esto puede incluir montar directorios personales, configurar variables de entorno, auditar el inicio/cierre de sesión, etc.

Archivos de Configuración PAM:

PAM lee su configuración de archivos. Hay dos estilos principales, aunque el segundo es el más común y preferido hoy en día:

1. **Estilo de Archivo Único:** Tradicionalmente, toda la configuración de PAM estaba en un solo archivo, `/etc/pam.conf`.
2. **Estilo Basado en Directorios:** El método moderno y más modular. La configuración se divide en archivos separados para cada "servicio" (aplicación que usa PAM).
    - **Ubicación:** `/etc/pam.d/`.

- **Nomenclatura:** Cada archivo en este directorio tiene el nombre del servicio que configura (ej: `/etc/pam.d/login` para el programa `login`, `/etc/pam.d/sshd` para OpenSSH, `/etc/pam.d/su` para el comando `su`, `/etc/pam.d/common-auth`, `/etc/pam.d/system-auth`).
- **Diferencias Debian vs. Red Hat:** Ambas ramas utilizan el estilo basado en directorios (`/etc/pam.d/`). Sin embargo, la organización interna de los archivos y cómo se incluyen unos a otros (ej: usando `include` en Debian, o archivos `system-auth`, `password-auth` en Red Hat) puede variar. Red Hat a menudo centraliza la configuración en `/etc/pam.d/system-auth` y `/etc/pam.d/password-auth` que son incluidos por otros archivos específicos del servicio.

Sintaxis de las Reglas (Líneas) en los Archivos de Configuración:

Cada línea significativa en un archivo de configuración PAM sigue el formato:

`<type> <control> <module-path> <module-options>`

- `<type>`: El grupo de gestión al que pertenece esta regla (`auth`, `account`, `password`, `session`).
- `<control>`: (Fundamental para entender PAM) Define cómo el éxito o fracaso de este módulo específico afecta el resultado final de la "pila" de reglas para este `type`.
    - `required`: El módulo DEBE tener éxito. Si falla, el usuario eventualmente FALLARÁ la autenticación/gestión, pero el procesamiento de los módulos restantes en esta pila para este `type` CONTINUARÁ. Esto evita que un atacante sepa exactamente qué módulo falló.
    - `requisite`: El módulo DEBE tener éxito. Si falla, el usuario FALLA INMEDIATAMENTE la autenticación/gestión, y el procesamiento de los módulos restantes para este `type` SE DETIENE. Más rápido que `required`, pero revela el punto exacto del fallo.
    - `sufficient`: El módulo puede tener éxito o fallar. Si tiene éxito, y ningún módulo `required` o `requisite` anterior ha fallado, el usuario PASA INMEDIATAMENTE la autenticación/gestión para este `type`, y el procesamiento de los módulos restantes para este `type` SE DETIENE. Un módulo exitoso y suficiente puede ser suficiente para permitir el acceso. Si falla, no se considera un fallo (a menos que no haya otros módulos exitosos/requeridos).
    - `optional`: El éxito o fracaso del módulo NO ES CRÍTICO. Generalmente, solo se considera el resultado de los módulos `required` o `requisite`. Los módulos `optional` son útiles para tareas secundarias (ej: logging, funciones experimentales) donde su fallo no debe impedir el login.
    - `include <ruta>`: Incluye reglas de otro archivo. Comúnmente utilizado para compartir configuraciones comunes (ej: `include common-auth`).

    - `substack <ruta>`: Similar a `include`, pero si un módulo en el substack falla con `required`, se comporta como si fuera `required` en la pila principal.
- `<module-path>`: La ruta al archivo de la librería del módulo PAM (ej: `/lib/security/pam_unix.so` o simplemente `pam_unix.so` si está en un directorio de módulos estándar).
- `<module-options>`: Argumentos que se pasan al módulo para configurar su comportamiento (ej: `debug`, `nodelay`, `nullok`, `try_first_pass`).

Procesamiento de Pila (Stack):

PAM procesa las reglas secuencialmente dentro de cada `type`. La evaluación final para cada `type` se basa en la combinación de los resultados de los módulos y sus flags de control.

Módulos PAM Comunes:

- `pam_unix.so`: El módulo estándar para autenticación, gestión de cuentas y contraseñas utilizando la base de datos de usuarios/grupos estándar de Unix (`/etc/passwd`, `/etc/shadow`).
- `pam_deny.so`: Siempre falla. Útil para denegar acceso.
- `pam_permit.so`: Siempre tiene éxito. Útil para permitir acceso.
- `pam_limits.so`: Aplica límites de recursos (`ulimit`) definidos en `/etc/security/limits.conf` (type `session`).
- `pam_securetty.so`: Restringe el login del usuario `root` a terminales seguras listadas en `/etc/securetty` (type `auth`).
- `pam_cracklib.so` / `pam_pwquality.so`: Comprueba la calidad de las nuevas contraseñas durante los cambios (type `password`).
- `pam_tally2.so` / `pam_faillock.so`: Bloquea cuentas después de N intentos fallidos de autenticación (type `auth`, `account`).
- `pam_env.so`: Configura variables de entorno (type `session`).
- `pam_ldap.so`: Autenticación y gestión de cuentas a través de LDAP (type `auth`, `account`, `password`).

Resolución de Problemas de PAM:

- **Archivos de Configuración:** Los errores de sintaxis o configuraciones lógicas incorrectas son comunes. Usa un editor con resaltado de sintaxis si es posible.
- **Logs:** Los mensajes de PAM (éxitos, fallos, errores de módulo) se registran en los logs del sistema, típicamente en `/var/log/auth.log` (Debian/Ubuntu) o `/var/log/secure` (Red Hat/CentOS/Fedora), o en el journal (`journalctl`).
- **Bloqueo del Sistema:** Una configuración incorrecta de PAM, especialmente en los módulos `auth` y `account` del servicio `login` o `sshd`, puede impedir que los usuarios (incluido root) inicien sesión. **SIEMPRE prueba los cambios de PAM en una VM y mantén una sesión de consola de root activa o un plan de recuperación antes de aplicar cambios que puedan bloquearte.**

- **Orden de Procesamiento:** Entender cómo funcionan los flags de control y el orden de las reglas es fundamental para depurar por qué PAM se comporta de una manera u otra.
