Curso Avanzado de PAM en Linux

Diseño Corporativo, Seguridad y Automatización

Miguel Alpañez Alcalde

16 de octubre de 2025

Índice general

Prefacio

Figura 1: Flujo alto nivel de PAM y módulos relevantes.

Este curso busca ser una guía completa y práctica para dominar PAM en entornos personales, profesionales y corporativos. Prioriza seguridad, auditabilidad, automatización y recuperación ante errores.

Capítulo 1

Arquitectura y flujo de PAM

1.1. Pilas y flags de control

Nota

Pilas: `auth`, `account`, `password`, `session`.

1.2. Servicios relevantes

- `sshd`, `sudo`, `su`, `login`, `gdm`, `crond`

1.3. Ficheros por defecto

- `/etc/pam.d/*`, `/etc/security/*`, `/etc/login.defs`

1.4. Ejemplo de configuración de SSH

Capítulo 2

Políticas de contraseñas y hashing

2.1. Complejidad con `pam_pwquality`

```
# /etc/security/pwquality.conf example
minlen = 12
dcredit = -1
ucredit = -1
lcredit = -1
ocredit = -1
maxrepeat = 2
minclass = 3
```

2.2. Historial y reutilización con `pam_pwhistory`

Ejemplo de pila (RHEL-like) en `system-auth`:

```
# /etc/security/pwquality.conf example
minlen = 12
dcredit = -1
ucredit = -1
lcredit = -1
ocredit = -1
maxrepeat = 2
minclass = 3
```

2.3. Buenas prácticas

- Evitar `nullok`
- Asegurar `sha512` y `rounds` adecuados en `pam_unix.so`

Capítulo 3

Bloqueos y antifuerza bruta con `pam_faillock`

3.1. Parámetros en `/etc/security/faillock.conf`

```
# /etc/security/faillock.conf example
deny = 5
fail_interval = 900
unlock_time = 600
audit
```

3.2. Persistencia del tally y exclusiones

Discute cuándo usar un directorio persistente y sus implicaciones operativas.

3.3. Visibilidad de eventos

Usar `faillock -user` y `ausearch`/`aureport`.

Capítulo 4

Control de acceso y sesión

4.1. Módulos útiles

`pam_access`, `pam_time`, `pam_limits`, `pam_tty_audit`, `pam_mkhomedir`, `pam_env`.

4.2. Plantillas típicas

```
# /etc/security/faillock.conf example
deny = 5
fail_interval = 900
unlock_time = 600
audit
```

Capítulo 5

Integración corporativa: SSSD, IdM/AD y Smartcards

5.1. Conceptos clave

Perfiles `authselect`, SSSD, *certificate mapping* para PIV/Smartcards.

Capítulo 6

MFA con llaves físicas: `pam_u2f` y YubiKey

6.1. MFA en `sudo`

```
# /etc/security/faillock.conf example
deny = 5
fail_interval = 900
unlock_time = 600
audit
```

Peligro

Proveer *break-glass* y segundo factor de respaldo.

Capítulo 7

Observabilidad y auditoría

7.1. Reglas de auditd

```
# /etc/security/faillock.conf example
deny = 5
fail_interval = 900
unlock_time = 600
audit
```

Capítulo 8

Automatización y testing

8.1. Pruebas de pilas PAM

`pamtester` / `pam-tester` y CI antes de desplegar en producción.

Capítulo 9

Desarrollo seguro de módulos PAM

9.1. Buenas prácticas

Evitar abuso de `pam_exec`, validar entradas y firmar artefactos.

Capítulo 10

Blueprint corporativo y operaciones

10.1. Perfiles por rol

Workstations, servidores y bastiones, con políticas diferenciadas.

10.2. Recuperación

Procedimientos de *fallback* y *rescue*.

Capítulo 11

Guía Debian/Ubuntu: equivalencias y buenas prácticas

Este capítulo resume las diferencias clave respecto a la familia RHEL y cómo aplicar las mismas políticas con las herramientas nativas de Debian/Ubuntu.

11.1. Mapa de equivalencias

RHEL/derivados	Debian/Ubuntu
`system-auth`, `password-auth`	`common-auth`, `common-password`, `common-account`, `co`
`authselect` (perfiles y symlinks)	`pam-auth-update` (perfiles en `/usr/share/pam-confi`
`pam_pwquality` + `/etc/security/pwquality.conf`	Igual (misma ruta `pwquality.conf`)
`pam_faillock` + `/etc/security/faillock.conf`	Igual (confirmar directorio de *tally*, por defecto `/var/r`

11.2. Gestionar PAM "a la Debian/Ubuntu"

En lugar de editar `common-*` directamente, crea perfiles en `/usr/share/pam-configs/` y actívalos con `pam-auth-update`. Esto evita conflictos en actualizaciones y asegura orden correcto de módulos.

11.2.1. Perfil para `pam_faillock`

```
# /etc/security/faillock.conf example
deny = 5
fail_interval = 900
unlock_time = 600
audit
```

Ajusta los parámetros globales en `/etc/security/faillock.conf`:

```
deny = 5
fail_interval = 900
unlock_time = 600
# dir = /var/run/faillock
audit
```

11.2.2. Política de contraseñas (`pam_pwquality`)

Edita `/etc/security/pwquality.conf` (idéntico a RHEL):

```
minlen = 12
dcredit = -1
ucredit = -1
lcredit = -1
ocredit = -1
maxrepeat = 2
minclass = 3
```

11.3. Gotchas frecuentes

- Evita tocar `common-*` si puedes usar `pam-auth-update`.
- Verifica la persistencia del *tally* de `faillock`: por defecto en `/var/run/faillock` (volátil).
- Prueba *en seco* con `pamtester` antes de desplegar en servidores críticos.

Apéndice A

Laboratorios guiados

Laboratorio

Lab 1 — Fail-lock end-to-end

1. Copia de seguridad de `/etc/pam.d` y `/etc/security`
2. Activa perfil `with-faillock` con `authselect`
3. Configura `faillock.conf`; genera intentos fallidos via `ssh`
4. Revisa `faillock -user` y `ausearch`/`aureport`

Laboratorio

Lab 2 — MFA con YubiKey en sudo

1. Instala `libpam-u2f` y genera `/etc/u2f`$_k$*eysAñade*`pam_u2f.so`*en*`/etc/pam.d/sudo`
2. Prueba con/ sin llave; añade llave de respaldo

Laboratorio

Lab 3 — Debian/Ubuntu: faillock con `pam-auth-update`

1. Crea el perfil en `/usr/share/pam-configs/faillock` (ver Apéndice E) y ejecuta `pam-auth-update`.
2. Ajusta `/etc/security/faillock.conf` y verifica el directorio del *tally*.
3. Genera intentos fallidos con `ssh`; revisa `faillock -user` y eventos con `ausearch`/`aureport`.

Laboratorio

Lab 4 — Smartcard/PIV con SSSD (conceptual)

1. Instalar `pcscd` y habilitar certificado (IdM/AD).
2. Probar mapeo de certificados y `pam_cert_auth`.

Laboratorio

Lab 5 — `pam_access` y `pam_time`

1. Definir políticas por host/usuario/TTY y franjas temporales.
2. Validar excepciones para cuentas de servicio.

Laboratorio

Lab 6 — `pam_tty_audit` en `sudo`

1. Activar `tty_audit` y confirmar trazabilidad en auditd.

Apéndice B

Snippets de Ansible

Ejemplo de tareas idempotentes para RHEL con `authselect`, `pwquality` y `faillock`.

```yaml
---
- name: Plantilla de /etc/security/faillock.conf
  copy:
    dest: /etc/security/faillock.conf
    content: |
      deny = 5
      fail_interval = 900
      unlock_time = 600
      audit
  notify: Reiniciar servicios relacionados
```

Apéndice C

Reglas de auditd sugeridas

```
---
- name: Plantilla de /etc/security/faillock.conf
  copy:
    dest: /etc/security/faillock.conf
    content: |
      deny = 5
      fail_interval = 900
      unlock_time = 600
      audit
  notify: Reiniciar servicios relacionados
```

Apéndice D

Checklist de riesgos y buenas prácticas

- Evitar `nullok` y `pam_exec` sin controles.
- Proveer *break-glass* documentado.
- No editar `system-auth`/`password-auth` sin `authselect`.

Perfiles para `pam-auth-update` (Debian/Ubuntu)

Los perfiles se colocan en `/usr/share/pam-configs/` y se activan con `pam-auth-update`.

D.1. Perfil: `faillock`

```
---
- name: Plantilla de /etc/security/faillock.conf
  copy:
    dest: /etc/security/faillock.conf
    content: |
      deny = 5
      fail_interval = 900
      unlock_time = 600
      audit
  notify: Reiniciar servicios relacionados
```

D.2. Opcional: `pam_u2f` para `sudo`

```
---
- name: Plantilla de /etc/security/faillock.conf
  copy:
    dest: /etc/security/faillock.conf
    content: |
      deny = 5
      fail_interval = 900
      unlock_time = 600
      audit
  notify: Reiniciar servicios relacionados
```

Infraestructura como Código (IaC) para PAM

Este apéndice reúne recetas listas para automatizar y validar PAM como código: Ansible (aplicación), Molecule + Testinfra (tests), CI, y un perfil `InSpec` para comprobación de cumplimiento.

D.3. Ansible: rol `pam_hardening` (multi-distro)

D.3.1. Variables por defecto

```yaml
pam_policy:
  minlen: 12
  dcredit: -1
  ucredit: -1
  lcredit: -1
  ocredit: -1
  maxrepeat: 2
  minclass: 3
faillock:
  deny: 5
  fail_interval: 900
  unlock_time: 600
  # dir: /var/log/faillock   # si quieres persistencia
use_u2f: false
```

D.3.2. Tareas para RHEL/derivados

```yaml
---
- name: Ensure pwquality.conf
  ansible.builtin.copy:
    dest: /etc/security/pwquality.conf
    mode: '0644'
    content: |
      minlen = {{ pam_policy.minlen }}
      dcredit = {{ pam_policy.dcredit }}
      ucredit = {{ pam_policy.ucredit }}
      lcredit = {{ pam_policy.lcredit }}
      ocredit = {{ pam_policy.ocredit }}
      maxrepeat = {{ pam_policy.maxrepeat }}
      minclass = {{ pam_policy.minclass }}

- name: Ensure faillock.conf
  ansible.builtin.copy:
    dest: /etc/security/faillock.conf
```

```
    mode: '0644'
    content: |
      deny = {{ faillock.deny }}
      fail_interval = {{ faillock.fail_interval }}
      unlock_time = {{ faillock.unlock_time }}
      audit

- name: Check current authselect profile
  ansible.builtin.command: authselect current
  register: authselect_current
  changed_when: false
  failed_when: false

- name: Set authselect profile (sssd with-faillock) if not active
  ansible.builtin.command: authselect select sssd with-faillock --force
  when: authselect_current.rc != 0 or ('with-faillock' not in
  ↪  authselect_current.stdout)
  notify: restart auditd

- name: Deploy audit rules for PAM
  ansible.builtin.copy:
    dest: /etc/audit/rules.d/40-pam.rules
    mode: '0644'
    content: |
      -w /etc/pam.d/ -p wa -k pam_changes
      -w /etc/security/ -p wa -k pam_security
      -w /etc/ssh/sshd_config -p wa -k ssh_config
      -w /etc/sudoers -p wa -k sudo_config
      -w /etc/sudoers.d/ -p wa -k sudo_config
  notify: restart auditd
```

D.3.3. Tareas para Debian/Ubuntu

```
---
- name: Ensure pwquality.conf
  ansible.builtin.copy:
    dest: /etc/security/pwquality.conf
    mode: '0644'
    content: |
      minlen = {{ pam_policy.minlen }}
      dcredit = {{ pam_policy.dcredit }}
      ucredit = {{ pam_policy.ucredit }}
      lcredit = {{ pam_policy.lcredit }}
      ocredit = {{ pam_policy.ocredit }}
      maxrepeat = {{ pam_policy.maxrepeat }}
      minclass = {{ pam_policy.minclass }}

- name: Install faillock profile for pam-auth-update
  ansible.builtin.copy:
    dest: /usr/share/pam-configs/faillock
    mode: '0644'
    content: |
      Name: Faillock (lock after failed logins)
      Default: yes
      Priority: 900
      Auth-Type: Primary
      Auth:
```

```
          requisite pam_faillock.so preauth
          [default=die] pam_faillock.so authfail
          sufficient pam_faillock.so authsucc
      Account-Type: Primary
      Account:
          required pam_faillock.so

- name: Check if faillock is enabled
  ansible.builtin.command: pam-auth-update --list --package
  changed_when: false
  register: pam_list

- name: Enable faillock via pam-auth-update
  ansible.builtin.command: pam-auth-update --enable faillock --package --force
  when: "'[+] faillock' not in pam_list.stdout"

- name: Ensure faillock.conf
  ansible.builtin.copy:
    dest: /etc/security/faillock.conf
    mode: '0644'
    content: |
      deny = {{ faillock.deny }}
      fail_interval = {{ faillock.fail_interval }}
      unlock_time = {{ faillock.unlock_time }}
      audit

- name: Deploy audit rules for PAM
  ansible.builtin.copy:
    dest: /etc/audit/rules.d/40-pam.rules
    mode: '0644'
    content: |
      -w /etc/pam.d/ -p wa -k pam_changes
      -w /etc/security/ -p wa -k pam_security
      -w /etc/ssh/sshd_config -p wa -k ssh_config
      -w /etc/sudoers -p wa -k sudo_config
      -w /etc/sudoers.d/ -p wa -k sudo_config
  notify: restart auditd
```

D.4. Pruebas con Molecule + Testinfra

D.4.1. `molecule.yml` (Docker)

```
---
dependency:
  name: galaxy
driver:
  name: docker
platforms:
  - name: rockylinux9
    image: docker.io/rockylinux:9
    privileged: true
    pre_build_image: true
  - name: ubuntu2404
    image: docker.io/library/ubuntu:24.04
    privileged: true
    pre_build_image: true
```

```
  - name: debian12
    image: docker.io/library/debian:12
    privileged: true
    pre_build_image: true
provisioner:
  name: ansible
  playbooks:
    converge: converge.yml
verifier:
  name: testinfra
```

D.4.2. Playbook de convergencia

```
---
- name: Converge
  hosts: all
  become: true
  gather_facts: true
  roles:
    - role: ../../ansible/roles/pam_hardening
```

D.4.3. Tests

```
import os
import pytest

def test_pwquality_conf(host):
    f = host.file('/etc/security/pwquality.conf')
    assert f.exists
    assert f.contains('minlen = 12')

def test_faillock_conf(host):
    f = host.file('/etc/security/faillock.conf')
    assert f.exists
    assert f.contains('deny = 5')

def test_pam_stack_has_faillock(host):
    # RHEL-like include via system-auth; Debian/Ubuntu via common-auth
    pamd = host.file('/etc/pam.d')
    assert pamd.is_directory
```

D.5. CI con GitHub Actions

```
name: CI
on:
  push:
  pull_request:
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Set up Python
```

```
        uses: actions/setup-python@v5
        with:
          python-version: '3.11'
      - name: Install deps
        run: |
          python -m pip install --upgrade pip
          pip install ansible ansible-lint yamllint molecule molecule-plugins[docker]
          ↪ testinfra
      - name: Molecule test
        run: |
          cd iac/molecule/default
          molecule test
```

D.6. Cumplimiento con InSpec

D.6.1. `inspec.yml`

```
name: pam-profile
title: PAM Policy Profile
version: 0.1.0
supports:
  - os-family: unix
```

D.6.2. Controles

```
control 'pwquality-1' do
  impact 1.0
  title 'Password min length is 12'
  desc 'Enforce password complexity via pwquality'
  describe file('/etc/security/pwquality.conf') do
    it { should exist }
    its('content') { should match /minlen\s*=\s*12/ }
  end
end

control 'faillock-1' do
  impact 1.0
  title 'Faillock deny is 5'
  desc 'Lock accounts after failed attempts'
  describe file('/etc/security/faillock.conf') do
    it { should exist }
    its('content') { should match /deny\s*=\s*5/ }
  end
end
```

D.7. Buenas prácticas de despliegue seguro

- Cambios por **oleadas** (p. ej., `serial: 1`) y con **rollback** preparado.
- Pruebas en **CI** y **ambiente staging** antes de producción.
- Evitar `shell`/`command` salvo cuando no haya módulo; documentar y hacerlos idempotentes.
- Versionar políticas: `pwquality.conf`, `faillock.conf`, perfiles de `pam-auth-update`.

Infraestructura como Código (IaC) para PAM: Ansible + Molecule

Este apéndice aporta un *role* reutilizable que aplica políticas de `pwquality`, `faillock`, reglas de `auditd` y (opcionalmente) `pam_u2f` en `sudo`, con soporte cruzado RHEL y Debian/Ubuntu.

D.8. Estructura del role

```
ansible/roles/pam_hardening/
 defaults/main.yml
 handlers/main.yml
 meta/main.yml
 tasks/{main.yml, rhel.yml, debian.yml}
 templates/{faillock.conf.j2, pwquality.conf.j2}
 molecule/default/{molecule.yml, prepare.yml, converge.yml, verify.yml,
↪  tests/test_pam.py}
```

D.9. Variables por defecto

```yaml
---
# Defaults for PAM hardening
pam_pwquality:
  minlen: 12
  dcredit: -1
  ucredit: -1
  lcredit: -1
  ocredit: -1
  maxrepeat: 2
  minclass: 3

pam_faillock:
  deny: 5
  fail_interval: 900
  unlock_time: 600
  # dir: /var/log/faillock   # uncomment to make tally persistent
  audit: true

pam_limits:
  - { domain: '*', type: 'soft', item: 'nofile', value: 65536 }

enable_u2f_for_sudo: false
u2f_authfile: /etc/u2f_keys
```

D.10. Tareas RHEL (authselect) y Debian/Ubuntu (pam-auth-update)

RHEL/derivados

```yaml
---
# RHEL/Alma/Rocky - use authselect and managed conf files
- name: Ensure authselect profile is selected (sssd with faillock)
  ansible.builtin.command: authselect select sssd with-faillock --force
  register: authselect_result
  changed_when: "'Profile was selected' in authselect_result.stdout or 'with-faillock' in authselect_result.stdout"
  failed_when: authselect_result.rc not in [0]
  check_mode: no

- name: Deploy /etc/security/faillock.conf
  ansible.builtin.template:
    src: faillock.conf.j2
    dest: /etc/security/faillock.conf
    owner: root
    group: root
    mode: '0644'
  notify: restart auditd

- name: Deploy /etc/security/pwquality.conf
  ansible.builtin.template:
    src: pwquality.conf.j2
    dest: /etc/security/pwquality.conf
    owner: root
    group: root
    mode: '0644'

- name: Ensure useful audit rules for PAM
  ansible.builtin.copy:
    dest: /etc/audit/rules.d/40-pam.rules
    content: |
      -w /etc/pam.d/ -p wa -k pam_changes
      -w /etc/security/ -p wa -k pam_security
      -w /etc/ssh/sshd_config -p wa -k ssh_config
      -w /etc/sudoers -p wa -k sudo_config
      -w /etc/sudoers.d/ -p wa -k sudo_config
  notify: restart auditd

- name: Optionally enable pam_u2f in sudo
  ansible.builtin.lineinfile:
    path: /etc/pam.d/sudo
    regexp: '^\s*auth\s+required\s+pam_u2f\.so'
    line: 'auth required pam_u2f.so authfile={{ u2f_authfile }} cue'
    insertafter: BOF
    state: "{{ 'present' if enable_u2f_for_sudo else 'absent' }}"
  when: enable_u2f_for_sudo | bool

- name: Ensure pam limits
  community.general.pam_limits:
    domain: "{{ item.domain }}"
    limit_type: "{{ item.type }}"
    limit_item: "{{ item.item }}"
```

```
    value: "{{ item.value }}"
  loop: "{{ pam_limits }}"
```

Debian/Ubuntu

```
---
# Debian/Ubuntu - use pam-auth-update profiles
- name: Install libpam-modules if missing
  ansible.builtin.package:
    name: [ 'libpam-modules', 'libpam-modules-bin' ]
    state: present

- name: Deploy faillock pam-auth-update profile
  ansible.builtin.copy:
    dest: /usr/share/pam-configs/faillock
    owner: root
    group: root
    mode: '0644'
    content: |
      Name: Faillock (lock after failed logins)
      Default: yes
      Priority: 900
      Auth-Type: Primary
      Auth:
          requisite pam_faillock.so preauth
          [default=die] pam_faillock.so authfail
          sufficient pam_faillock.so authsucc
      Account-Type: Primary
      Account:
          required pam_faillock.so

- name: Check if faillock profile is enabled
  ansible.builtin.command: pam-auth-update --list
  register: pam_list
  changed_when: false

- name: Enable faillock profile (idempotent)
  ansible.builtin.command: pam-auth-update --enable faillock --package
  when: "'faillock [enabled]    Faillock (lock after failed logins)' not in
  ↪  pam_list.stdout"

- name: Deploy /etc/security/faillock.conf
  ansible.builtin.template:
    src: faillock.conf.j2
    dest: /etc/security/faillock.conf
    owner: root
    group: root
    mode: '0644'
  notify: restart auditd

- name: Deploy /etc/security/pwquality.conf
  ansible.builtin.template:
    src: pwquality.conf.j2
    dest: /etc/security/pwquality.conf
    owner: root
    group: root
    mode: '0644'
```

```yaml

- name: Optionally enable pam_u2f in sudo (Debian/Ubuntu)
  ansible.builtin.lineinfile:
    path: /etc/pam.d/sudo
    regexp: '^\s*auth\s+required\s+pam_u2f\.so'
    line: 'auth required pam_u2f.so authfile={{ u2f_authfile }} cue'
    insertafter: BOF
    state: "{{ 'present' if enable_u2f_for_sudo else 'absent' }}"
  when: enable_u2f_for_sudo | bool

- name: Ensure pam limits
  community.general.pam_limits:
    domain: "{{ item.domain }}"
    limit_type: "{{ item.type }}"
    limit_item: "{{ item.item }}"
    value: "{{ item.value }}"
  loop: "{{ pam_limits }}"
```

D.11. Pruebas automatizadas con Molecule + Testinfra

```yaml
---
dependency:
  name: galaxy
driver:
  name: docker
lint: |
  set -e
  ansible-lint
platforms:
  - name: rockylinux9
    image: rockylinux:9
    privileged: true
    command: /sbin/init
  - name: ubuntu2404
    image: ubuntu:24.04
    privileged: true
    command: /sbin/init
provisioner:
  name: ansible
  playbooks:
    converge: converge.yml
verifier:
  name: testinfra
```

```python
import os
import pytest

def test_pwquality_conf(host):
    f = host.file('/etc/security/pwquality.conf')
    assert f.exists
    assert f.contains('minlen = 12')

def test_faillock_conf(host):
    f = host.file('/etc/security/faillock.conf')
    assert f.exists
```

```
    assert 'deny = ' in f.content_string

def test_pam_stack_contains_faillock(host):
    # Check common-auth or system-auth depending on distro
    if host.file('/etc/pam.d/common-auth').exists:
        cf = host.file('/etc/pam.d/common-auth')
        assert 'pam_faillock.so' in cf.content_string
    elif host.file('/etc/pam.d/system-auth').exists:
        sf = host.file('/etc/pam.d/system-auth')
        assert 'pam_faillock.so' in sf.content_string
```

D.12. Uso

```
pipx install ansible ansible-lint molecule-plugins[docker] pytest-testinfra
cd ansible/roles/pam_hardening
molecule test
```

Nota

Idempotencia: las tareas evitan cambios innecesarios; los comandos se ejecutan sólo cuando la configuración aún no está activa (p. ej., `pam-auth-update`).

Matriz de controles (NIST 800-53, ISO 27001, CIS)

Esta matriz enlaza políticas y configuraciones del curso con marcos de control comunes.

Marco	**Control**	**Implementación en el curso**
NIST 800-53	IA-5 (1), (6)	Complejidad y calidad de contraseñas con `pam_pwquality`; historial con `pam_pwhistory`.
NIST 800-53	AC-7	Bloqueo tras intentos fallidos con `pam_faillock`.
NIST 800-53	IA-2 (1), (2)	MFA con `pam_u2f` (U2F/FIDO2) en `sudo`/`login`.
NIST 800-53	AU-2, AU-6	Auditoría de cambios en PAM/sudo/SSH con `auditd`, análisis con `ausearch`/`aureport`.
ISO 27001 A.5.17	Gestión de identidades	Políticas de autenticación, historial y bloqueo por fallos.
ISO 27001 A.8.2	Gestión de privilegios	Reglas PAM para `sudo`, `su`, y límites de sesión.
CIS L1 Linux	5.x, 6.x	`pwquality.conf`, `faillock.conf`, `auditd` activo y con reglas de archivos sensibles.

Runbooks: *break-glass* y recuperación

D.13. Escenario: bloqueo por configuración PAM

1. Acceso a consola (IPMI/DRAC/ILO) o modo `rescue`.
2. Arrancar con `systemd.unit=rescue.target` o editar `grub` para `single`.
3. Montar la raíz en lectura-escritura y revertir los cambios:

```
cp -a /etc/pam.d /etc/pam.d.backup.$(date +%F-%T)
# RHEL-like (authselect)
authselect check || authselect select sssd --force
restorecon -R /etc/pam.d || true
# Debian/Ubuntu
pam-auth-update --disable faillock --package || true
```

4. Revisar `/var/log/audit/audit.log` y `journalctl -u sshd`.

D.14. Checklist de recuperación

- Validar acceso local y por SSH con cuenta administrativa.
- Restaurar plantillas de `pwquality.conf` y `faillock.conf` conocidas.
- Ejecutar pruebas básicas con `pamtester`.