

Curso Avanzado de PAM en Linux  
Diseño Corporativo, Seguridad y Automatización

Miguel Alpañez Alcalde

16 de octubre de 2025

# Índice general

<b>1. Arquitectura y flujo de Pluggable Authentication Modules (PAM)</b>	<b>1</b>
1.1. Pilas y flags de control . . . . .	1
1.2. Servicios relevantes . . . . .	1
1.3. Ficheros por defecto . . . . .	1
1.4. Ejemplo de configuración de SSH . . . . .	1
<b>2. Políticas de contraseñas y hashing</b>	<b>2</b>
2.1. Complejidad con <code>pam_pwquality</code> . . . . .	2
2.2. Historial y reutilización con <code>pam_pwhistory</code> . . . . .	2
2.3. Buenas prácticas . . . . .	2
<b>3. Bloqueos y antifuerza bruta con <code>pam_faillock</code></b>	<b>3</b>
3.1. Parámetros en <code>/etc/security/faillock.conf</code> . . . . .	3
3.2. Persistencia del tally y exclusiones . . . . .	3
3.3. Visibilidad de eventos . . . . .	3
<b>4. Control de acceso y sesión</b>	<b>4</b>
4.1. Módulos útiles . . . . .	4
4.2. Plantillas típicas . . . . .	4
<b>5. Integración corporativa: SSSD, IdM/AD y Smartcards</b>	<b>5</b>
5.1. Conceptos clave . . . . .	5
<b>6. MFA con llaves físicas: <code>pam_u2f</code> y YubiKey</b>	<b>6</b>
6.1. MFA en <code>sudo</code> . . . . .	6
<b>7. Observabilidad y auditoría</b>	<b>7</b>
7.1. Reglas de auditd . . . . .	7
<b>8. Automatización y testing</b>	<b>8</b>
8.1. Pruebas de pilas PAM . . . . .	8
<b>9. Desarrollo seguro de módulos PAM</b>	<b>9</b>
9.1. Buenas prácticas . . . . .	9
<b>10.Blueprint corporativo y operaciones</b>	<b>10</b>
10.1. Perfiles por rol . . . . .	10
10.2. Recuperación . . . . .	10
<b>11.Guía Debian/Ubuntu: equivalencias y buenas prácticas</b>	<b>11</b>
11.1. Mapa de equivalencias . . . . .	11
11.2. Gestionar PAM “a la Debian/Ubuntu” . . . . .	11
11.2.1. Perfil para <code>pam_faillock</code> . . . . .	11

11.2.2. Política de contraseñas ( <code>pam_pwquality</code> ) . . . . .	12
11.3. Gotchas frecuentes . . . . .	12
<b>A. Laboratorios guiados</b>	<b>13</b>
<b>B. Snippets de Ansible</b>	<b>15</b>
<b>C. Reglas de auditd sugeridas</b>	<b>16</b>
<b>D. Checklist de riesgos y buenas prácticas</b>	<b>17</b>
<b>Perfiles para <code>pam-auth-update</code> (Debian/Ubuntu)</b>	<b>19</b>
D.1. Perfil: <code>faillock</code> . . . . .	19
D.2. Opcional: <code>pam_u2f</code> para <code>sudo</code> . . . . .	19
<b>Infraestructura como Código (IaC) para PAM</b>	<b>20</b>
D.3. Ansible: rol <code>pam_hardening</code> (multi-distro) . . . . .	20
D.3.1. Variables por defecto . . . . .	20
D.3.2. Tareas para RHEL/derivados . . . . .	20
D.3.3. Tareas para Debian/Ubuntu . . . . .	21
D.4. Pruebas con Molecule + Testinfra . . . . .	22
D.4.1. <code>molecule.yml</code> (Docker) . . . . .	22
D.4.2. Playbook de convergencia . . . . .	23
D.4.3. Tests . . . . .	23
D.5. CI con GitHub Actions . . . . .	23
D.6. Cumplimiento con InSpec . . . . .	24
D.6.1. <code>inspec.yml</code> . . . . .	24
D.6.2. Controles . . . . .	24
D.7. Buenas prácticas de despliegue seguro . . . . .	24
<b>Infraestructura como Código (IaC) para PAM: Ansible + Molecule</b>	<b>25</b>
D.8. Estructura del role . . . . .	25
D.9. Variables por defecto . . . . .	25
D.10. Tareas RHEL ( <code>authselect</code> ) y Debian/Ubuntu ( <code>pam-auth-update</code> ) . . . . .	26
D.11. Pruebas automatizadas con Molecule + Testinfra . . . . .	28
D.12. Uso . . . . .	29
<b>Matriz de controles (NIST 800-53, ISO 27001, CIS)</b>	<b>30</b>
<b>Runbooks: <i>break-glass</i> y recuperación</b>	<b>31</b>
D.13. Escenario: bloqueo por configuración PAM . . . . .	31
D.14. Checklist de recuperación . . . . .	31

# 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

---

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

---

### 2.2. Historial y reutilización con pam\_pwhistory

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

---

```
1 # /etc/security/pwquality.conf example
2 minlen = 12
3 dcredit = -1
4 ucredit = -1
5 lcredit = -1
6 ocredit = -1
7 maxrepeat = 2
8 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

---

```
1 # /etc/security/faillock.conf example
2 deny = 5
3 fail_interval = 900
4 unlock_time = 600
5 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

---

```
1 # /etc/security/faillock.conf example
2 deny = 5
3 fail_interval = 900
4 unlock_time = 600
5 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

---

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

---

#### Peligro

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

# Capítulo 7

## Observabilidad y auditoría

### 7.1. Reglas de auditd

---

```
1 # /etc/security/faillock.conf example
2 deny = 5
3 fail_interval = 900
4 unlock_time = 600
5 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, common-session
authselect (perfiles y symlinks)	pam-auth-update (perfiles en /usr/share/pam-configs/)
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/run/tally)

### 11.2. Gestionar PAM “a la Debian/Ubuntu”

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

#### 11.2.1. Perfil para `pam_faillock`

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

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

```
1 deny = 5
2 fail_interval = 900
3 unlock_time = 600
4 # dir = /var/run/faillock
5 audit
```

### 11.2.2. Política de contraseñas (`pam_pwquality`)

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

---

```
1 minlen = 12
2 dcredit = -1
3 ucredit = -1
4 lcredit = -1
5 ocredit = -1
6 maxrepeat = 2
7 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
<b>Lab 1 — Fail-lock end-to-end</b> <ol style="list-style-type: none"><li>1. Copia de seguridad de <code>/etc/pam.d</code> y <code>/etc/security</code></li><li>2. Activa perfil <code>with-faillock</code> con <code>authselect</code></li><li>3. Configura <code>faillock.conf</code>; genera intentos fallidos via <code>ssh</code></li><li>4. Revisa <code>faillock -user</code> y <code>ausearch/aureport</code></li></ol>
<b>Lab 2 — MFA con YubiKey en sudo</b> <ol style="list-style-type: none"><li>1. Instala <code>libpam-u2f</code> y genera <code>/etc/u2f_keys</code> Añade <code>pam_u2f.so</code> en <code>/etc/pam.d/sudo</code></li><li>2. Prueba con/ sin llave; añade llave de respaldo</li></ol>
<b>Lab 3 — Debian/Ubuntu: faillock con pam-auth-update</b> <ol style="list-style-type: none"><li>1. Crea el perfil en <code>/usr/share/pam-configs/faillock</code> (ver Apéndice E) y ejecuta <code>pam-auth-update</code>.</li><li>2. Ajusta <code>/etc/security/faillock.conf</code> y verifica el directorio del <code>tally</code>.</li><li>3. Genera intentos fallidos con <code>ssh</code>; revisa <code>faillock -user</code> y eventos con <code>ausearch/aureport</code>.</li></ol>
<b>Lab 4 — Smartcard/PIV con SSSD (conceptual)</b> <ol style="list-style-type: none"><li>1. Instalar <code>pcscd</code> y habilitar certificado (IdM/AD).</li><li>2. Probar mapeo de certificados y <code>pam_cert_auth</code>.</li></ol>

**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.

---

```
1  -----
2 - name: Plantilla de /etc/security/faillock.conf
3   copy:
4     dest: /etc/security/faillock.conf
5     content: |
6       deny = 5
7       fail_interval = 900
8       unlock_time = 600
9       audit
10  notify: Reiniciar servicios relacionados
```

---

## Apéndice C

# Reglas de auditd sugeridas

---

```
1 -----  
2 - name: Plantilla de /etc/security/faillock.conf  
3   copy:  
4     dest: /etc/security/faillock.conf  
5     content: |  
6       deny = 5  
7       fail_interval = 900  
8       unlock_time = 600  
9       audit  
10  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

---

```
1  -----
2  - name: Plantilla de /etc/security/faillock.conf
3  copy:
4      dest: /etc/security/faillock.conf
5      content: |
6          deny = 5
7          fail_interval = 900
8          unlock_time = 600
9          audit
10     notify: Reiniciar servicios relacionados
```

---

## D.2. Opcional: pam\_u2f para sudo

---

```
1  -----
2  - name: Plantilla de /etc/security/faillock.conf
3  copy:
4      dest: /etc/security/faillock.conf
5      content: |
6          deny = 5
7          fail_interval = 900
8          unlock_time = 600
9          audit
10     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

---

```
1  pam_policy:
2    minlen: 12
3    dcredit: -1
4    ucredit: -1
5    lcredit: -1
6    ocredit: -1
7    maxrepeat: 2
8    minclass: 3
9  faillock:
10    deny: 5
11    fail_interval: 900
12    unlock_time: 600
13    # dir: /var/log/faillock  # si quieres persistencia
14  use_u2f: false
```

---

### D.3.2. Tareas para RHEL/derivados

---

```
1  ---
2  - name: Ensure pwquality.conf
3    ansible.builtin.copy:
4      dest: /etc/security/pwquality.conf
5      mode: '0644'
6      content: |
7        minlen = {{ pam_policy.minlen }}
8        dcredit = {{ pam_policy.dcredit }}
9        ucredit = {{ pam_policy.ucredit }}
10       lcredit = {{ pam_policy.lcredit }}
11       ocredit = {{ pam_policy.ocredit }}
12       maxrepeat = {{ pam_policy.maxrepeat }}
13       minclass = {{ pam_policy.minclass }}
14
15  - name: Ensure faillock.conf
16    ansible.builtin.copy:
17      dest: /etc/security/faillock.conf
```

```

18     mode: '0644'
19     content: |
20         deny = {{ faillock.deny }}
21         fail_interval = {{ faillock.fail_interval }}
22         unlock_time = {{ faillock.unlock_time }}
23         audit
24
25 - name: Check current authselect profile
26   ansible.builtin.command: authselect current
27   register: authselect_current
28   changed_when: false
29   failed_when: false
30
31 - name: Set authselect profile (sssd with-faillock) if not active
32   ansible.builtin.command: authselect select sssd with-faillock --force
33   when: authselect_current.rc != 0 or ('with-faillock' not in
34         ← authselect_current.stdout)
35   notify: restart audited
36
37 - name: Deploy audit rules for PAM
38   ansible.builtin.copy:
39     dest: /etc/audit/rules.d/40-pam.rules
40     mode: '0644'
41     content: |
42         -w /etc/pam.d/ -p wa -k pam_changes
43         -w /etc/security/ -p wa -k pam_security
44         -w /etc/ssh/sshd_config -p wa -k ssh_config
45         -w /etc/sudoers -p wa -k sudo_config
46         -w /etc/sudoers.d/ -p wa -k sudo_config
47   notify: restart audited

```

---

### D.3.3. Tareas para Debian/Ubuntu

```

1  -----
2 - name: Ensure pwquality.conf
3   ansible.builtin.copy:
4     dest: /etc/security/pwquality.conf
5     mode: '0644'
6     content: |
7         minlen = {{ pam_policy.minlen }}
8         dcredit = {{ pam_policy.dcredit }}
9         ucredit = {{ pam_policy.ucredit }}
10        lcredit = {{ pam_policy.lcredit }}
11        ocredit = {{ pam_policy.ocredit }}
12        maxrepeat = {{ pam_policy.maxrepeat }}
13        minclass = {{ pam_policy.minclass }}
14
15 - name: Install faillock profile for pam-auth-update
16   ansible.builtin.copy:
17     dest: /usr/share/pam-configs/faillock
18     mode: '0644'
19     content: |
20         Name: Faillock (lock after failed logins)
21         Default: yes
22         Priority: 900
23         Auth-Type: Primary
24         Auth:

```

```

25      requisite pam_faillock.so preauth
26      [default=die] pam_faillock.so authfail
27      sufficient pam_faillock.so authsucc
28      Account-Type: Primary
29      Account:
30          required pam_faillock.so
31
32 - name: Check if faillock is enabled
33     ansible.builtin.command: pam-auth-update --list --package
34     changed_when: false
35     register: pam_list
36
37 - name: Enable faillock via pam-auth-update
38     ansible.builtin.command: pam-auth-update --enable faillock --package --force
39     when: "'[+] faillock' not in pam_list.stdout"
40
41 - name: Ensure faillock.conf
42     ansible.builtin.copy:
43         dest: /etc/security/faillock.conf
44         mode: '0644'
45         content: |
46             deny = {{ faillock.deny }}
47             fail_interval = {{ faillock.fail_interval }}
48             unlock_time = {{ faillock.unlock_time }}
49             audit
50
51 - name: Deploy audit rules for PAM
52     ansible.builtin.copy:
53         dest: /etc/audit/rules.d/40-pam.rules
54         mode: '0644'
55         content: |
56             -w /etc/pam.d/ -p wa -k pam_changes
57             -w /etc/security/ -p wa -k pam_security
58             -w /etc/ssh/sshd_config -p wa -k ssh_config
59             -w /etc/sudoers -p wa -k sudo_config
60             -w /etc/sudoers.d/ -p wa -k sudo_config
61     notify: restart audited

```

---

## D.4. Pruebas con Molecule + Testinfra

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

```

1 ---
2 dependency:
3     name: galaxy
4 driver:
5     name: docker
6 platforms:
7     - name: rockylinux9
8         image: docker.io/rockylinux:9
9         privileged: true
10        pre_build_image: true
11    - name: ubuntu2404
12        image: docker.io/library/ubuntu:24.04
13        privileged: true
14        pre_build_image: true

```

```

15   - name: debian12
16     image: docker.io/library/debian:12
17     privileged: true
18     pre_build_image: true
19   provisioner:
20     name: ansible
21     playbooks:
22       converge: converge.yml
23   verifier:
24     name: testinfra

```

---

#### D.4.2. Playbook de convergencia

```

1  -----
2  - name: Converge
3    hosts: all
4    become: true
5    gather_facts: true
6    roles:
7      - role: ../../ansible/roles/pam_hardening

```

---

#### D.4.3. Tests

```

1 import os
2 import pytest
3
4 def test_pwquality_conf(host):
5     f = host.file('/etc/security/pwquality.conf')
6     assert f.exists
7     assert f.contains('minlen = 12')
8
9 def test_faillock_conf(host):
10    f = host.file('/etc/security/faillock.conf')
11    assert f.exists
12    assert f.contains('deny = 5')
13
14 def test_pam_stack_has_faillock(host):
15     # RHEL-like include via system-auth; Debian/Ubuntu via common-auth
16     pamd = host.file('/etc/pam.d')
17     assert pamd.is_directory

```

---

### D.5. CI con GitHub Actions

```

1 name: CI
2 on:
3   push:
4     pull_request:
5   jobs:
6     test:
7       runs-on: ubuntu-latest
8       steps:
9         - uses: actions/checkout@v4
10           - name: Set up Python

```

```

11      uses: actions/setup-python@v5
12      with:
13        python-version: '3.11'
14      - name: Install deps
15        run: |
16          python -m pip install --upgrade pip
17          pip install ansible ansible-lint yamllint molecule molecule-plugins[docker]
18          ↪ testinfra
19      - name: Molecule test
20        run: |
21          cd iac/molecule/default
22          molecule test

```

---

## D.6. Cumplimiento con InSpec

### D.6.1. inspec.yml

```

1 name: pam-profile
2 title: PAM Policy Profile
3 version: 0.1.0
4 supports:
5   - os-family: unix

```

---

### D.6.2. Controles

```

1 control 'pwquality-1' do
2   impact 1.0
3   title 'Password min length is 12'
4   desc 'Enforce password complexity via pwquality'
5   describe file('/etc/security/pwquality.conf') do
6     it { should exist }
7     its('content') { should match /minlen\s*=\s*12/ }
8   end
9 end
10
11 control 'faillock-1' do
12   impact 1.0
13   title 'Faillock deny is 5'
14   desc 'Lock accounts after failed attempts'
15   describe file('/etc/security/faillock.conf') do
16     it { should exist }
17     its('content') { should match /deny\s*=\s*5/ }
18   end
19 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

---

```
1 ansible/roles/pam_hardening/
2   defaults/main.yml
3   handlers/main.yml
4   meta/main.yml
5   tasks/{main.yml, rhel.yml, debian.yml}
6   templates/{faillock.conf.j2, pwquality.conf.j2}
7   molecule/default/{molecule.yml, prepare.yml, converge.yml, verify.yml,
8     ↳ tests/test_pam.py}
```

---

## D.9. Variables por defecto

---

```
1 -----
2 # Defaults for PAM hardening
3 pam_pwquality:
4   minlen: 12
5   dccredit: -1
6   uccredit: -1
7   lccredit: -1
8   occredit: -1
9   maxrepeat: 2
10  minclass: 3
11
12 pam_faillock:
13   deny: 5
14   fail_interval: 900
15   unlock_time: 600
16   # dir: /var/log/faillock  # uncomment to make tally persistent
17   audit: true
18
19 pam_limits:
20   - { domain: '*', type: 'soft', item: 'nofile', value: 65536 }
21
22 enable_u2f_for_sudo: false
23 u2f_authfile: /etc/u2f_keys
```

---

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

### RHEL/derivados

---

```

1  ---
2  # RHEL/Alma/Rocky - use authselect and managed conf files
3  - name: Ensure authselect profile is selected (sssd with faillock)
4    ansible.builtin.command: authselect select sssd with-faillock --force
5    register: authselect_result
6    changed_when: "'Profile was selected' in authselect_result.stdout or 'with-faillock'
7      ↳ in authselect_result.stderr"
8    failed_when: authselect_result.rc not in [0]
9    check_mode: no
10
11 - name: Deploy /etc/security/faillock.conf
12   ansible.builtin.template:
13     src: faillock.conf.j2
14     dest: /etc/security/faillock.conf
15     owner: root
16     group: root
17     mode: '0644'
18     notify: restart auditd
19
20 - name: Deploy /etc/security/pwquality.conf
21   ansible.builtin.template:
22     src: pwquality.conf.j2
23     dest: /etc/security/pwquality.conf
24     owner: root
25     group: root
26     mode: '0644'
27
28 - name: Ensure useful audit rules for PAM
29   ansible.builtin.copy:
30     dest: /etc/audit/rules.d/40-pam.rules
31     content: |
32       -w /etc/pam.d/ -p wa -k pam_changes
33       -w /etc/security/ -p wa -k pam_security
34       -w /etc/ssh/sshd_config -p wa -k ssh_config
35       -w /etc/sudoers -p wa -k sudo_config
36       -w /etc/sudoers.d/ -p wa -k sudo_config
37     notify: restart auditd
38
39 - name: Optionally enable pam_u2f in sudo
40   ansible.builtin.lineinfile:
41     path: /etc/pam.d/sudo
42     regexp: '^s*auth\s+required\s+pam_u2f\.so'
43     line: 'auth required pam_u2f.so authfile={{ u2f_authfile }} cue'
44     insertafter: BOF
45     state: "{{ 'present' if enable_u2f_for_sudo else 'absent' }}"
46     when: enable_u2f_for_sudo | bool
47
48 - name: Ensure pam limits
49   community.general.pam_limits:
50     domain: "{{ item.domain }}"
51     limit_type: "{{ item.type }}"
      limit_item: "{{ item.item }}"

```

```

52     value: "{{ item.value }}"
53   loop: "{{ pam_limits }}"

```

---

## Debian/Ubuntu

---

```

1   ---
2   # Debian/Ubuntu - use pam-auth-update profiles
3   - name: Install libpam-modules if missing
4     ansible.builtin.package:
5       name: [ 'libpam-modules', 'libpam-modules-bin' ]
6       state: present
7
8   - name: Deploy faillock pam-auth-update profile
9     ansible.builtin.copy:
10      dest: /usr/share/pam-configs/faillock
11      owner: root
12      group: root
13      mode: '0644'
14      content: |
15        Name: Faillock (lock after failed logins)
16        Default: yes
17        Priority: 900
18        Auth-Type: Primary
19        Auth:
20          requisite pam_faillock.so preauth
21          [default=die] pam_faillock.so authfail
22          sufficient pam_faillock.so authsucc
23        Account-Type: Primary
24        Account:
25          required pam_faillock.so
26
27   - name: Check if faillock profile is enabled
28     ansible.builtin.command: pam-auth-update --list
29     register: pam_list
30     changed_when: false
31
32   - name: Enable faillock profile (idempotent)
33     ansible.builtin.command: pam-auth-update --enable faillock --package
34     when: "'faillock [enabled]    Faillock (lock after failed logins)' not in
35           → pam_list.stdout"
36
37   - name: Deploy /etc/security/faillock.conf
38     ansible.builtin.template:
39       src: faillock.conf.j2
40       dest: /etc/security/faillock.conf
41       owner: root
42       group: root
43       mode: '0644'
44       notify: restart audited
45
46   - name: Deploy /etc/security/pwquality.conf
47     ansible.builtin.template:
48       src: pwquality.conf.j2
49       dest: /etc/security/pwquality.conf
50       owner: root
51       group: root
52       mode: '0644'

```

```

52
53 - name: Optionally enable pam_u2f in sudo (Debian/Ubuntu)
54   ansible.builtin.lineinfile:
55     path: /etc/pam.d/sudo
56     regexp: '^s*auth\s+required\s+pam_u2f\.so'
57     line: 'auth required pam_u2f.so authfile={{ u2f_authfile }} cue'
58     insertafter: BOF
59     state: "{{ 'present' if enable_u2f_for_sudo else 'absent' }}"
60   when: enable_u2f_for_sudo | bool
61
62 - name: Ensure pam limits
63   community.general.pam_limits:
64     domain: "{{ item.domain }}"
65     limit_type: "{{ item.type }}"
66     limit_item: "{{ item.item }}"
67     value: "{{ item.value }}"
68   loop: "{{ pam_limits }}"

```

---

## D.11. Pruebas automatizadas con Molecule + Testinfra

```

1 ---
2 dependency:
3   name: galaxy
4 driver:
5   name: docker
6 lint: |
7   set -e
8   ansible-lint
9 platforms:
10  - name: rockylinux9
11    image: rockylinux:9
12    privileged: true
13    command: /sbin/init
14  - name: ubuntu2404
15    image: ubuntu:24.04
16    privileged: true
17    command: /sbin/init
18 provisioner:
19   name: ansible
20   playbooks:
21     converge: converge.yml
22 verifier:
23   name: testinfra

```

---

```

1 import os
2 import pytest
3
4 def test_pwquality_conf(host):
5     f = host.file('/etc/security/pwquality.conf')
6     assert f.exists
7     assert f.contains('minlen = 12')
8
9 def test_faillock_conf(host):
10    f = host.file('/etc/security/faillock.conf')
11    assert f.exists

```

```
12     assert 'deny = ' in f.content_string
13
14 def test_pam_stack_contains_faillock(host):
15     # Check common-auth or system-auth depending on distro
16     if host.file('/etc/pam.d/common-auth').exists:
17         cf = host.file('/etc/pam.d/common-auth')
18         assert 'pam_faillock.so' in cf.content_string
19     elif host.file('/etc/pam.d/system-auth').exists:
20         sf = host.file('/etc/pam.d/system-auth')
21         assert 'pam_faillock.so' in sf.content_string
```

---

## D.12. Uso

```
1 pipx install ansible ansible-lint molecule-plugins[docker] pytest-testinfra
2 cd ansible/roles/pam_hardening
3 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 <code>pam_pwquality</code> ; historial con <code>pam_pwhistory</code> .
NIST 800-53	AC-7	Bloqueo tras intentos fallidos con <code>pam_faillock</code> .
NIST 800-53	IA-2 (1), (2)	MFA con <code>pam_u2f</code> (U2F/FIDO2) en <code>sudo/login</code> .
NIST 800-53	AU-2, AU-6	Auditoría de cambios en PAM/sudo/SSH con <code>auditd</code> , análisis con <code>ausearch/aureport</code> .
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 <code>sudo</code> , <code>su</code> , y límites de sesión.
CIS L1 Linux	5.x, 6.x	<code>pwquality.conf</code> , <code>faillock.conf</code> , <code>auditd</code> 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:

---

```
1 cp -a /etc/pam.d /etc/pam.d.backup.$(date +%F-%T)
2 # RHEL-like (authselect)
3 authselect check || authselect select sssd --force
4 restorecon -R /etc/pam.d || true
5 # Debian/Ubuntu
6 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`.