ADMIN:DOCS

Linux Administration #2: Fortgeschrittene Benutzerverwaltung

Dieser Artikel behandelt fortgeschrittene Aspekte der Linux-Benutzerverwaltung, von PAM-Konfiguration über Benutzerquotas bis hin zu ACLs und der Überwachung von Benutzeraktivitäten.

Kategorie: Linux-Administration **Veröffentlicht:** 27.10.2024 **Zuletzt geprüft:** 02.11.2026

Willkommen zum zweiten Teil unserer technischen Wiki-Serie über Linux-Administration!

Nachdem wir im ersten Artikel die Grundlagen der Benutzer- und Rechteverwaltung kennengelernt haben, tauchen wir nun tiefer in die fortgeschrittenen Aspekte ein. Die fortgeschrittene Benutzerverwaltung umfasst alle Werkzeuge und Konzepte, die über die grundlegende Erstellung und Löschung von Benutzerkonten hinausgehen.

Stell dir ein großes Unternehmen vor

Die einfache Benutzerverwaltung wäre wie ein Pförtner, der Ausweise ausstellt. Die fortgeschrittene Benutzerverwaltung hingegen ist wie ein komplettes Sicherheitssystem mit Zugangskontrolle, Überwachung und automatischer Verwaltung.

> ⚠ **Hinweis:** In diesem Artikel verwenden wir Ubuntu/Debian als Beispiel-Distribution. Die grundlegenden Konzepte sind auf allen Linux-Systemen gleich, aber die Installation von Paketen und einige Konfigurationspfade können sich je nach Distribution unterscheiden.

Quelle: https://admindocs.de/de/linux-administration/linux-administration-fortgeschrittene-benutzerverwaltung

> Wenn du eine andere Distribution verwendest, konsultiere bitte die entsprechende Dokumentation für die spezifischen Installationsbefehle und Pfade.

Benötigte Pakete installieren

Installation der PAM-Module für erweiterte Passwort-Policies:

```
sudo apt update
sudo apt install libpam-pwquality cracklib-runtime
```

Überprüfen der Installation:

```
dpkg -l | grep libpam-pwquality
dpkg -l | grep cracklib
```

Funktion der Komponenten:

- `libpam-pwquality`: PAM-Modul für konfigurierbare Passwort-Qualitätsprüfungen (Länge, Komplexität, Wiederholungen).
- `cracklib-runtime`: Wörterbuch-Bibliothek zur automatischen Erkennung trivialer, erratbarer Passwörter.

Konfigurationsdateien:

- `/etc/security/pwquality.conf`: Hauptkonfiguration der Passwort-Qualitätskriterien
- `/etc/security/pwquality.conf.d/`: Drop-in-Verzeichnis für modulare Konfigurationen

Passwort-Policies verstehen

Eine Passwort-Policy definiert verbindliche Regeln für die Erstellung, Sicherheit und den Lebenszyklus von Benutzerzugangsdaten:

- **Komplexität & Entropie:** Minimale Länge, Zeichenvielfalt, Vermeidung von Tastaturmustern und Wörterbucheinträgen.
- **Gültigkeit & Lebenszyklus:** Vorwarnzeiten, Inaktivitätsfristen und temporäre Kontobeschränkungen.
- **Historie & Wiederverwendung:** Speicherung bisheriger Passwort-Hashes zur Verhinderung direkter Wiederverwendung.

PAM-Konfiguration für Passwort-Policies

PAM (Pluggable Authentication Modules) ist das zentrale Authentifizierungs-Framework unter Linux. Es entkoppelt Systemprogramme von den konkreten Authentifizierungsmechanismen und gliedert sich in vier standardisierte Management-Gruppen:

Sitzungsverwaltung (session)

Konfiguriert Benutzerumgebung, Dateisystem-Mounts und Ressourcen-Limits

Passwortverwaltung (password)

Passwortverwaltung (password)

Prüft Token-Gültigkeit, erzwingt Qualitätsregeln und speichert Hashes

Account-Statusprüfung (account)

Account-Statusprüfung (account)

Überprüft Gültigkeitsfristen, Ablaufdaten und Login-Beschränkungen

Identitätsprüfung (auth)

Authentifiziert den Benutzer (Passwortabfrage, PAM-Tokens oder MFA/FIDO2)

Passwort-Qualitätsrichtlinien einrichten

Die Vorgaben für neue Passwörter werden zentral über `pwquality` gesteuert:

```
sudo nano /etc/security/pwquality.conf
```

Wichtige Parameter im Überblick:

```
# Minimale Passwortlänge (moderne Empfehlung: min. 14 Zeichen)
minlen = 14

# Mindestens 3 verschiedene Zeichenklassen (Groß, Klein, Ziffern, Sonderzeichen)
minclass = 3

# Maximal 2 aufeinanderfolgende gleiche Zeichen
maxrepeat = 2

# Verhindert Passwörter, die den Benutzernamen enthalten
gecoscheck = 1

# Aktiviert die Wörterbuchprüfung über Cracklib
dictcheck = 1
```

PAM-Module für Passwörter konfigurieren

In Debian und Ubuntu wird das Passwortmodul über `/etc/pam.d/common-password` in den System-Stack eingebunden:

```
sudo nano /etc/pam.d/common-password
```

Moderne Konfigurationszeile:

```
password    requisite   pam_pwquality.so retry=3
password    sufficient  pam_unix.so use_authtok yescrypt shadow
```

> **Standard auf modernen Distributionen – yescrypt:** Moderne Distributionen (Debian 11+, Ubuntu 22.04+) nutzen in `pam_unix.so` standardmäßig den Algorithmus **`yescrypt`** anstelle des älteren `sha512`. Yescrypt ist durch speicherharte Schlüsselausleitung (Memory-Hardness) resistent gegen massiv-parallele Angriffe über GPUs und ASICs.

Passwort-Historie und Ablaufverwaltung

> **Sicherheitsstandard (BSI & NIST SP 800-63B): Keine starre 90-Tage-Rotation**
> Sowohl das BSI als auch das US-amerikanische NIST raten heute explizit von **erzwungenen, kalenderbasierten Passwortabläufen** (wie `PASS_MAX_DAYS 90`) ab. Die Praxis zeigt, dass wiederkehrende Wechsel zu vorhersehbaren Mustern und schwächeren Passwörtern führen. Die moderne Vorgabe lautet: **Ausreichende Länge (mindestens 12–16 Zeichen oder Passphrasen)**, Schutz durch Passwort-Manager und Multi-Faktor-Authentifizierung (MFA) sowie Passwortänderung **nur bei Verdacht auf Kompromittierung**. Die Aging-Parameter (`chage`, `login.defs`) werden in Unternehmen vor allem für zeitlich begrenzte Gast- oder Praktikantenkonten genutzt.

Passwort-Historie konfigurieren

Um zu verhindern, dass Benutzer kürzlich verwendete Kennwörter sofort wiederverwenden, wird `pam_pwhistory.so` in `/etc/pam.d/common-password` ergänzt:

```
password required pam_pwhistory.so remember=5 enforce_for_root use_authtok
```

- `remember=5`: Speichert die Hashes der letzten 5 Passwörter in `/etc/security/opasswd`
- `enforce_for_root`: Bindet auch administrative Kennwortänderungen durch Root an die Historie
- `use_authtok`: Übergibt das vom vorherigen Qualitätsmodul verifizierte Token

Passwort-Ablauf einrichten

Systemweite Standardwerte für neue Benutzerkonten liegen in `/etc/login.defs`:

```
PASS_MAX_DAYS   90      # Maximales Alter in Tagen (für Konten mit Ablaufpflicht)
PASS_MIN_DAYS   1       # Minimaler Abstand zwischen zwei Änderungen
PASS_WARN_AGE   14      # Vorwarnzeit vor Ablauf in Tagen
```

Ablauf für ein spezifisches Benutzerkonto konfigurieren:

```
sudo chage -M 90 -m 1 -W 14 username
```

Detaillierte Ablaufkonfiguration prüfen

Quelle: https://admindocs.de/de/linux-administration/linux-administration-fortgeschrittene-benutzerverwaltung

```
sudo chage -l username

# Beispielausgabe:
# Last password change                    : Jan 01, 2026
# Password expires                        : Apr 01, 2026
# Password inactive                       : never
# Account expires                         : never
# Minimum number of days between changes  : 1
# Maximum number of days between changes  : 90
# Number of days of warning before expire : 14
```

Häufige Probleme bei der Passwort-Policy

Problem: Passwortänderung schlägt fehl

```
# Fehlermeldung bei Passwortänderung:
passwd

# Ausgabe: "Password does not meet complexity requirements"
# Überprüfen der aktuellen Policy:
sudo cat /etc/security/pwquality.conf | grep -v '^#'
```

Häufige Ursachen:

- Passwort zu kurz
- Fehlende Zeichenklassen
- Enthält Benutzername
- Zu ähnlich zum alten Passwort
- Passwort-Historie verletzt

Problem: Benutzer kann sich nicht anmelden

```
# Überprüfen des Passwort-Status
sudo chage -l username

# Häufige Ursachen und Lösungen:
# 1. Passwort abgelaufen
sudo chage -d 0 username

# Erzwingt Änderung beim Login
# 2. Account gesperrt
sudo usermod -U username

# Account entsperren
# 3. PAM-Konfiguration prüfen
sudo cat /etc/pam.d/common-auth
```

Praktische Beispiele

Szenario 1: Entwicklungsteam-Setup

```
# Passwort-Policy für Entwickler
sudo nano /etc/security/pwquality.conf

minlen = 14
minclass = 3
maxrepeat = 2
enforce_for_root

# Passwort-Alter für sensible Projekte
sudo chage -M 60 -m 1 -W 7 developer1
```

Szenario 2: Temporäre Mitarbeiter

```
# Ablaufdatum setzen (z. B. Jahresende 2026)
sudo useradd -e 2026-12-31 tempuser

# oder für einen existierenden Benutzer anpassen
sudo chage -E 2026-12-31 tempuser

# Status überprüfen
sudo chage -l tempuser
```

Szenario 3: Automatische Passwortänderung erzwingen

Neue Mitarbeiter sollen ihr initiales Einmalpasswort direkt beim ersten Login ändern müssen:

```
# Sofortige Passwort-Änderung beim nächsten Login erzwingen
sudo chage -d 0 newuser

# Überprüfen der Einstellungen
sudo chage -l newuser
```

Szenario 4: Temporäre Zugänge

Praktikanten erhalten Zugang für exakt 3 Monate ab Anlagezeitpunkt:

```
# Benutzer mit relativem Ablaufdatum erstellen
sudo useradd -e $(date -d "+3 months" +%Y-%m-%d) intern01

# Vorwarnung 14 Tage vor Ablauf aktivieren
sudo chage -W 14 intern01
```

```
# Status überprüfen
sudo chage -l intern01
```

Szenario 5: Passwort-Historie durchsetzen

```
# PAM-Konfiguration anpassen
sudo nano /etc/pam.d/common-password

# Historien-Modul ergänzen:
password required pam_pwhistory.so remember=5 use_authtok
```

Benutzerquotas implementieren

Quotas begrenzen den Speicherplatz oder die Anzahl anlegbarer Dateien für einzelne Benutzer oder ganze Gruppen, um eine Überfüllung von Dateisystemen (Denial of Service durch volle Platten) zu verhindern.

Wichtige Quota-Typen und Konzepte:

- **Block-Quotas:** Begrenzt das nutzbare Datenvolumen (Speicherplatz in KB/MB/GB).
- **Inode-Quotas:** Begrenzt die Anzahl anlegbarer Dateisystemobjekte (Dateien, Ordner, Links).
- **Soft-Limit:** Schwellenwert mit Karenzzeit (Grace Period); löst Warnungen aus, erlaubt temporäres Überschreiten bis zum Ablauf der Frist.
- **Hard-Limit:** Unwiderrufliche Obergrenze; Schreiboperationen brechen bei Erreichen sofort mit `Disk quota exceeded` ab.

> **Moderne In-Kernel-Quotas (ext4 & XFS):** Auf modernen Linux-Systemen werden ext4-Quotas direkt über interne Inodes im Dateisystem-Journal geführt (`tune2fs -O quota /dev/sdX`). Das historische Werkzeug `quotacheck` und externe Dateien (`aquota.user`) sind auf zeitgemäßen ext4-Systemen obsolet. Auf Enterprise-Dateisystemen wie **XFS** sind Quotas fester Bestandteil der Metadaten und werden nativ über Mount-Optionen (`uquota,gquota`) sowie das dedizierte Tool **`xfs_quota`** administriert.

Quota-System aktivieren

Quota-Tools installieren

```
sudo apt update
sudo apt install quota quotatool
```

`/etc/fstab` für Mount-Optionen anpassen

```
# Füge usrquota und grpquota zu den Optionen der Partition hinzu
sudo nano /etc/fstab

# Beispiel-Eintrag für /home:
# UUID=xxxx-xxxx  /home  ext4  defaults,usrquota,grpquota  0  2
```

Dateisystem neu einbinden

```
sudo mount -o remount /home
```

Quotas einrichten

Klassische Quota-Datenbanken initialisieren & aktivieren

```
# Nur bei traditionellem ext4 nötig (bei modernem In-Kernel-ext4 entfällt quotacheck):
sudo quotacheck -cugm /home

# Quotas aktivieren
sudo quotaon -v /home
```

Quota für Benutzer interaktiv bearbeiten

```
sudo edquota -u username
```

Erweiterte Quota-Konfigurationen

Soft und Hard Limits verstehen

Beim Aufruf von `sudo edquota -u username` öffnet sich der Standard-Editor mit folgender Struktur:

```
Filesystem  blocks  soft    hard   inodes  soft  hard
/dev/sda1   500000  524288  786432  1000   1500  2000
```

- `blocks`: Aktueller Speicherverbrauch (in 1-KB-Blöcken)
- `soft`: Warnschwelle vor Beginn der Karenzzeit
- `hard`: Absolute Sperrgrenze für Speicherplatz
- `inodes`: Aktuelle Dateianzahl (inklusive Soft- und Hard-Limits für Dateien)

Gruppenquotas verwalten

```
# Quota für eine Gruppe setzen
sudo edquota -g entwickler

# Quota-Status für eine Gruppe anzeigen
sudo quota -g entwickler

# Quota-Report für alle Gruppen auf /home ausgeben
sudo repquota -g /home
```

Karenzzeit (Grace Period) konfigurieren

```
sudo edquota -t
```

```
Filesystem    Block grace period    Inode grace period
/dev/sda1     7days                 7days
```

Praktische Quota-Szenarien

Entwicklungsteam-Setup

Großzügige Limits für Entwickler:

```
sudo edquota -u entwickler1
```

```
Filesystem  blocks   soft     hard     inodes  soft   hard
/dev/sda1   500000  5242880  6291456   10000   50000  75000
# Bedeutet: 5 GB Soft-Limit, 6 GB Hard-Limit, 50.000 Dateien Soft, 75.000 Dateien Hard
```

Praktikanten-Setup

Restriktive Limits für temporäre Benutzerkonten:

```
sudo edquota -u praktikant1
```

```
Filesystem  blocks  soft    hard    inodes  soft  hard
/dev/sda1   100000  524288  786432  1000    2000  3000
```

Limits von einem bestehenden Benutzer auf mehrere andere übertragen:

```
sudo edquota -p praktikant1 praktikant2 praktikant3
```

Projekt-Quotas

Gemeinsames Projektverzeichnis mit Gruppenquota limitieren:

```
sudo edquota -g projektgruppe
```

```
Filesystem  blocks    soft      hard      inodes  soft   hard
/dev/sda1   1000000  10485760  15728640   20000   50000  75000
```

Quota-Report für das Projekt anzeigen:

```
sudo repquota -g /home | grep projektgruppe
```

Quota-Überwachung und -Verwaltung

```
# Quota-Status für einen spezifischen Benutzer anzeigen
sudo quota -v username

# Systemweiter Quota-Report für alle Benutzer
sudo repquota -a /home

# Detaillierte Quota-Statistiken in lesbaren Einheiten
sudo quota -vs username
```

Praktisches Monitoring-Beispiel

```
#!/bin/bash
# Quota-Überwachungsskript: Benutzer über 80% Speicherauslastung finden
repquota -a | awk '$4 > 80 {print $1": "$4"%"}'
```

Access Control Lists (ACLs)

Access Control Lists (ACLs) erweitern das traditionelle dreistufige Linux-Berechtigungssystem (Owner, Group, Others) um feingranulare, rollenbasierte Zugriffsrechte für beliebige zusätzliche Benutzer und Gruppen.

Voraussetzungen (Paket installieren):

```
sudo apt update
sudo apt install acl
```

ACL-Konzepte verstehen

Das POSIX-ACL-System arbeitet mit drei wesentlichen Strukturkomponenten:

- **Erweiterte Zugriffs-ACLs:** Spezifische Berechtigungen für zusätzliche Benutzer (`u:benutzer:rwx`) oder Gruppen (`g:gruppe:rx`), unabhängig vom primären Dateibesitzer.
- **Berechtigungsmaske (`mask`):** Definiert die maximal wirksamen Rechte für alle erweiterten Einträge und fängt unbeabsichtigte Rechteausweitungen ab.
- **Vererbbare Default-ACLs (`d:g:...`):** Können auf Verzeichnissen hinterlegt werden; jede neu erstellte Unterdatei oder jeder Unterordner erbt diese Rechte automatisch.

ACL-Verwaltung

```
# Lese- und Schreibrechte für eine Gruppe auf einen gesamten Verzeichnisbaum setzen
sudo setfacl -R -m g:entwickler:rwx /projekte/webapp/
sudo setfacl -R -m g:qa:rx /projekte/webapp/

# Automatische Vererbung für zukünftige neue Dateien im Verzeichnis aktivieren
sudo setfacl -d -m g:entwickler:rwx /projekte/webapp/
```

Praktische ACL-Anwendungsfälle

Entwicklungsprojekt-Setup

```
# Projektstruktur erstellen
sudo mkdir -p /projekte/webapp/{src,docs,config}

# Basis-ACLs für das Entwicklerteam
sudo setfacl -m g:entwickler:rwx /projekte/webapp/src
sudo setfacl -m g:entwickler:rwx /projekte/webapp/docs

# Leserechte für das QA-Team
sudo setfacl -m g:qa:rx /projekte/webapp/src
sudo setfacl -m g:qa:r /projekte/webapp/docs

# Konfigurationsverzeichnis nur für Administratoren voll beschreibbar
sudo setfacl -m g:admin:rwx,g:entwickler:r /projekte/webapp/config
```

ACLs verwalten und analysieren

```
# Vorhandene ACL-Einträge einer Datei detailliert anzeigen
getfacl datei.txt

# Spezifische ACLs für Benutzer oder Gruppe zuweisen
setfacl -m u:anna:rw- projekt.txt
setfacl -m g:entwickler:r-x scripts/

# Mehrere Berechtigungen in einem Aufruf kombinieren
setfacl -m u:anna:rw-,g:projekt:r-- datei.txt

# Alle erweiterten ACLs restlos entfernen und Standardrechte wiederherstellen
setfacl -b datei.txt
```

Praktische ACL-Beispiele

Standard-ACLs für ein Projektverzeichnis

```
# Projektverzeichnis erstellen
sudo mkdir -p /projekte/webapp

# Basis-ACLs setzen
# Entwickler haben vollen Zugriff
# QA-Team hat Lese- und Ausführrechte
sudo setfacl -m g:entwickler:rwx,g:qa:rx /projekte/webapp

# Standard-ACLs für neue Dateien
sudo setfacl -d -m g:entwickler:rwx,g:qa:rx /projekte/webapp

# ACLs überprüfen
getfacl /projekte/webapp
```

Geteiltes Dokumentenverzeichnis

```
# Dokumentenstruktur
sudo mkdir -p /dokumente/{public,team,management}

# Public: Alle können lesen
sudo setfacl -m g:users:rx /dokumente/public
sudo setfacl -d -m g:users:rx /dokumente/public

# Team: Nur Teammitglieder
sudo setfacl -m g:team:rwx /dokumente/team
sudo setfacl -d -m g:team:rwx /dokumente/team

# Management: Nur bestimmte Personen
sudo setfacl -m u:chef:rwx,u:sekretariat:rx /dokumente/management
```

ACLs für verschiedene Anwendungsfälle

Gemeinsames Dokumentenverzeichnis:

```
# Verzeichnisstruktur:
/dokumente/
├── public/

# Alle können lesen
├── team/

# Nur Teammitglieder
└── private/

# Nur spezifische Benutzer
# ACLs setzen
sudo setfacl -R -m g:users:rx /dokumente/public
sudo setfacl -R -m g:team:rwx /dokumente/team
sudo setfacl -m u:chef:rwx,u:sekretariat:rx /dokumente/private
```

Weitere praktische ACL-Beispiele

Komplexe Projektstruktur mit ACLs

```
# Projektstruktur erstellen
sudo mkdir -p /projekte/webapp/{src,docs,config,tests}

# Basis-Berechtigungen
sudo chown -R root:entwickler /projekte/webapp
sudo chmod 2775 /projekte/webapp

# Differenzierte ACLs setzen
# Entwickler haben vollen Zugriff auf src und tests
sudo setfacl -R -m g:entwickler:rwx /projekte/webapp/src
sudo setfacl -R -m g:entwickler:rwx /projekte/webapp/tests

# QA-Team hat Lese- und Ausführrechte für tests
sudo setfacl -R -m g:qa:rx /projekte/webapp/tests
```

```
# Dokumentation für alle lesbar
sudo setfacl -R -m g:users:rx /projekte/webapp/docs

# Config nur für Administratoren
sudo setfacl -R -m g:admin:rwx,g:entwickler:r /projekte/webapp/config
```

ACL-Vererbung für neue Dateien

```
# Standard-ACLs für neue Dateien setzen
sudo setfacl -d -m g:entwickler:rwx /projekte/webapp/src
sudo setfacl -d -m g:qa:rx /projekte/webapp/tests

# ACL-Vererbung überprüfen
getfacl /projekte/webapp/src
```

Benutzeraktivitäten überwachen

Die Überwachung von Benutzeraktivitäten und Authentifizierungsereignissen ist unverzichtbar für die Erkennung von Sicherheitsvorfällen und unautorisierten Zugriffen.

Voraussetzungen (Audit-System installieren):

```
sudo apt update
sudo apt install auditd audispd-plugins
```

> **Hinweis zur auditd-Architektur:** In modernen Versionen des Linux-Audit-Frameworks wurde der Dispatcher `audispd` direkt in den Hauptdämon `auditd` integriert. Plugins (wie `audisp-syslog`) werden heute modular über Konfigurationsdateien unter `/etc/audit/plugins.d/` aktiviert.

Login-Überwachung

> **Moderne Login-Überwachung mit systemd-journald:** Auf modernen minimalen Distributionen (wie Ubuntu 24.04 LTS und Debian 12/13) ist `rsyslog` nicht mehr vorinstalliert; `/var/log/auth.log` existiert auf frischen Systemen nicht. Der zeitgemäße Standard für Log-Abfragen ist **`journalctl`**:
>
> - SSH-Logins in Echtzeit: `journalctl -u ssh.service -f`
> - Fehlgeschlagene Anmeldeversuche: `journalctl -u ssh.service -g "Failed password"`
> - Anmeldesitzungen verfolgen: `journalctl -u systemd-logind -f`

```
# Aktuelle interaktive Sitzungen anzeigen
who
w

# Login-Historie (erfolgreiche Anmeldungen)
last -n 20

# Fehlgeschlagene Anmeldeversuche (Bad Logins)
sudo lastb -n 20

# SSH-Logins in Echtzeit über systemd-journald verfolgen
journalctl -u ssh.service -f
```

```
# Traditionelle Textdatei (nur falls rsyslog nachinstalliert wurde)
# tail -f /var/log/auth.log | grep sshd
```

Benutzeraktivitäten aufzeichnen

```
# Audit-System aktivieren
sudo apt install auditd

# Audit-Regeln hinzufügen
sudo auditctl -w /etc/passwd -p wa -k user-modify
sudo auditctl -w /etc/group -p wa -k group-modify

# Audit-Logs anzeigen
sudo ausearch -k user-modify
sudo aureport --auth

# Authentifizierungsbericht
```

Ressourcennutzung überwachen

```
# Prozesse eines Benutzers
ps -u username

# Ressourcenverbrauch
top -u username
iotop -u username

# I/O-Aktivität
# Detaillierte Prozessanalyse
sudo ps auxf | grep username
```

Erweiterte Monitoring-Tools

Prozess-Monitoring mit auditd:

```
# Prozess-Überwachung einrichten
sudo auditctl -a exit,always -F arch=b64 -S execve -k cmd_exec

# Benutzeraktionen überwachen
sudo ausearch -k cmd_exec
sudo aureport -x --summary

# Detaillierte Prozessanalyse
sudo ausearch -k cmd_exec -i | grep username
```

Login-Monitoring mit fail2ban:

```
# Installation
sudo apt install fail2ban

# Konfiguration prüfen
sudo fail2ban-client status
sudo fail2ban-client status sshd

# Geblockte IPs anzeigen
sudo fail2ban-client get sshd banip
```

Ressourcen-Monitoring:

```
# CPU und Memory pro Benutzer
top -U username

# Disk-Nutzung pro Benutzer
sudo du -sh /home/*

# Netzwerkaktivität
sudo nethogs
sudo iftop
```

Echtzeit-Überwachung

System Activity Reporter (SAR):

```
# Installation
sudo apt install sysstat

# Aktivitäten aufzeichnen
sudo sar -u 1 5

# CPU-Nutzung alle 1 Sekunde, 5 Mal
sudo sar -r 1 5

# Speichernutzung
sudo sar -b 1 5

# I/O-Statistiken
```

Prozess-Tracking:

```
# Prozess-Baum eines Benutzers
pstree username

# Detaillierte Prozessinformationen
ps auxf | grep username

# Kontinuierliche Überwachung
watch -n 1 'ps aux | grep username'
```

Integration Monitoring-Tools

Zentrales Logging-System (rsyslog)

Falls externe SIEM-Kollektoren oder bestehende Skripte zwingend traditionelle Text-Logdateien wie `/var/log/auth.log` voraussetzen, kann `rsyslog` nachinstalliert werden:

```
sudo apt update
sudo apt install rsyslog
sudo systemctl enable --now rsyslog
```

Konfiguration unter `/etc/rsyslog.d/auth-logging.conf`:

```
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
```

Automatisiertes Monitoring-Skript (Journald-basiert)

```
#!/bin/bash
# user-monitor.sh - Sicherheitsüberwachung via systemd-journald
REPORT="/var/log/user-report.txt"

echo "=== Täglicher Benutzeraktivitätsbericht ===" > "$REPORT"
date >> "$REPORT"

echo -e "\n--- Fehlgeschlagene SSH-Logins (letzte 24h) ---" >> "$REPORT"
journalctl -u ssh.service --since "24 hours ago" -g "Failed password" --no-pager >>
"$REPORT"

echo -e "\n--- Sudo-Befehlsausführungen ---" >> "$REPORT"
journalctl --since "24 hours ago" -g "sudo:" --no-pager >> "$REPORT"

echo -e "\n--- Top 5 CPU-Prozesse ---" >> "$REPORT"
ps aux --sort=-%cpu | head -n 6 >> "$REPORT"
```

Troubleshooting

Häufige PAM-Probleme und Lösungen

PAM-Konfigurationsprobleme

Problem: Fehlermeldung `Authentication token manipulation error`

Diagnoseschritte:

```
# 1. PAM-Logs im systemd-Journal prüfen
journalctl -u systemd-logind -f
```

```
# 2. PAM-Modul-Abhängigkeiten prüfen
ldd /usr/lib/x86_64-linux-gnu/security/pam_pwquality.so

# 3. PAM-Konfiguration sicher über das Wartungswerkzeug aktualisieren
sudo pam-auth-update --verbose
```

Berechtigungsprobleme bei der Shadow-Datei

```
# Dateirechte von /etc/shadow prüfen (muss -rw-r----- root:shadow sein)
ls -l /etc/shadow

# Falls Rechte abweichen, korrigieren:
sudo chown root:shadow /etc/shadow
sudo chmod 640 /etc/shadow
```

Praktische Notfall-Lösungen

Passwort-Historie temporär umgehen (für Notfälle):

```
sudo nano /etc/pam.d/common-password
# remember=5 temporär auskommentieren
```

Gesperrtes Root-Konto wieder entsperren:

```
sudo passwd -u root
```

Vergessenes Benutzerpasswort als Root neu vergeben:

```
sudo passwd username
```

Best Practices für das Troubleshooting

Systematische Fehlersuche

Checkliste für Authentifizierungs- und Zugriffsprobleme:

- [ ] Sind die benötigten PAM-Module (`pam_pwquality.so` , `pam_unix.so`) installiert?
- [ ] Besitzen `/etc/passwd` (644) und `/etc/shadow` (640) die korrekten Zugriffsrechte?
- [ ] Wurden die Logs via `journalctl -u ssh.service` bzw. `journalctl -u systemd-logind` auf Fehler geprüft?
- [ ] Ist das Benutzerkonto nicht gesperrt (`sudo passwd -S <user>`)?
- [ ] Wurde das Quota-Limit überprüft (`sudo quota -v <user>`)?

Logging und Monitoring

```
# Echtzeit-Authentifizierungsüberwachung
```

```
journalctl -u ssh.service -u systemd-logind -f

# Fehlgeschlagene Anmeldeversuche filtern
journalctl -u ssh.service -g "Failed password"
```

Notfall-Wiederherstellung

```
# Temporäre Policy-Deaktivierung (nur im Notfall!)
sudo mv /etc/pam.d/common-password /etc/pam.d/common-password.bak
sudo cp /etc/pam.d/common-password.orig /etc/pam.d/common-password

# Backup wiederherstellen
sudo mv /etc/pam.d/common-password.bak /etc/pam.d/common-password
```

Dokumentation und Reporting

```
# Aktuelle Konfiguration dokumentieren
sudo cp /etc/security/pwquality.conf /root/pwquality.conf.$(date +%F)

# Konfigurationsänderungen protokollieren
echo "$(date): PAM-Konfiguration angepasst" >> /var/log/security-changes.log

# Regelmäßige Überprüfungen
sudo aureport --auth

# Authentifizierungsberichte
sudo aureport --failed

# Fehlgeschlagene Anmeldungen
```

Übung: Sichere Benutzerumgebung einrichten

Szenario: Du richtest für ein Entwicklungsteam eine Benutzerumgebung mit modernen Passwort-Policies und Zugriffskontrollen ein.

Anforderungen:

- Passwort-Mindestlänge: 14 Zeichen mit mindestens 3 Zeichenklassen
- Passwort-Historie: Die letzten 5 Kennwörter dürfen nicht wiederverwendet werden
- Gruppen: `entwickler`, `praktikanten`, `externe`
- Benutzer mit differenzierten Fristen und Gültigkeiten anlegen

Musterlösung:

```
#!/bin/bash
# 1. Passwort-Qualität über pwquality festlegen
sudo nano /etc/security/pwquality.conf
# minlen = 14
# minclass = 3
# maxrepeat = 2
# enforce_for_root

# 2. Passwort-Historie in /etc/pam.d/common-password aktivieren
# password required pam_pwhistory.so remember=5 use_authtok

# 3. Gruppen anlegen
sudo groupadd entwickler
sudo groupadd praktikanten
sudo groupadd externe

# 4. Entwickler anlegen (Standardkonto)
sudo useradd -m -s /bin/bash -G entwickler dev1

# 5. Praktikant mit 30 Tagen Frist anlegen
sudo useradd -m -s /bin/bash -G praktikanten praktikant1
sudo chage -M 30 -m 1 -W 7 praktikant1
```

Quelle: https://admindocs.de/de/linux-administration/linux-administration-fortgeschrittene-benutzerverwaltung

```
# 6. Externer Berater mit festem Ablaufdatum in 90 Tagen
sudo useradd -m -s /bin/bash -G externe extern1
sudo chage -E $(date -d "+90 days" +%Y-%m-%d) extern1
```

Verifikations-Checkliste:

- [ ] `pwquality.conf` erzwingt mindestens 14 Zeichen
- [ ] Historien-Sperre in `/etc/pam.d/common-password` aktiv
- [ ] Gruppenmitgliedschaften via `id dev1` geprüft
- [ ] Ablaufdatum via `sudo chage -l extern1` überprüft

Befehlsreferenz (Cheatsheet)

Für den schnellen Zugriff bei der Benutzer-, Gruppen- und Rechtekontrolle fasst die folgende Referenz die wichtigsten Kommandos der fortgeschrittenen Benutzerverwaltung zusammen:

Passwort-Ablaufdaten abfragen Zeigt detaillierte Ablaufdaten, Fristen und Warnungen an

$ `sudo chage -l <user>`

Ablaufzeit und Vorwarnung setzen Setzt maximale Gültigkeit (90 Tage) und Vorwarnzeit (14 Tage)

$ `sudo chage -M 90 -W 14 <user>`

Sofortigen Passwortwechsel erzwingen Erzwingt eine Kennwortänderung beim nächsten Login

$ `sudo chage -d 0 <user>`

Erweiterte ACLs anzeigen Listet alle erweiterten Zugriffssteuerungslisten einer Datei auf

$ `getfacl <datei>`

Benutzer-ACL setzen Weist einem spezifischen Benutzer Lese- und Schreibrechte zu

$ `setfacl -m u:<user>:rw- <datei>`

Vererbbare Default-ACL auf Ordner setzen Definiert vererbbare Standard-ACLs für alle neuen Dateien

$ `setfacl -d -m g:<gruppe>:rwx <ordner>`

Alle erweiterten ACLs entfernenLöscht alle erweiterten ACL-Einträge restlos

$ `setfacl -b <datei>`

Benutzer-Quota interaktiv bearbeiten Editiert Speicherplatz- und Inode-Limits für einen Benutzer

$ `sudo edquota -u <user>`

Quota-Limits per Befehl setzen Weist Grenzwerte direkt per Kommandozeile oder Skript zu

$ `sudo setquota -u <user> 500M 600M 0 0 /home`

Systemweiten Quota-Report generieren Erstellt Übersichtsbericht für alle Benutzer und Gruppen

$ `sudo repquota -avugs`

Sudoers-Konfiguration bearbeitenÖffnet Konfiguration mit automatischer Syntaxprüfung

$ `sudo visudo`

Sudoers-Syntax prüfenÜberprüft Konfigurationsdateien ohne sie zu öffnen

$ `sudo visudo -c`

Erfolgreiche Logins anzeigen Zeigt die letzten 20 erfolgreichen Systemanmeldungen

$ `last -n 20`

Fehlgeschlagene Logins anzeigen Listet fehlerhafte Anmeldeversuche (Bad Logins) auf

```
$ sudo lastb -n 20
```

Schreibzugriffe per Kernel-Audit überwachen Protokolliert Dateiänderungen über das Audit-Subsystem

```
$ sudo auditctl -w <pfad> -p wa -k datei-monitor
```

Weiterführende Ressourcen

Die folgenden Leitfäden, Spezifikationen und internen Kursmodule vertiefen die fortgeschrittene Benutzerverwaltung und Zugriffskontrolle:

Linux-PAM System Administrator's Guide Offizielles Handbuch zur modularen PAM-Architektur

POSIX Access Control Lists auf Linux Offizielle Man Page zu POSIX-ACLs und Berechtigungsmasken

Linux Quota Mini-HOWTO Ausführliche Dokumentation zum Quota-Subsystem unter Linux

Sudoers Manual & Syntax-Referenz Offizielle Dokumentation zur Sudoers-Konfiguration

Linux Administration #1: Grundlagen Das Fundament: Systemarchitektur, FHS, Mounten & Dateirechte

chmod & Dateiberechtigungen Leitfaden Fundamentale Vertiefung zu Berechtigungen und Oktalwerten

Fazit

Mit der Beherrschung von PAM-Authentifizierungs-Stacks, strikten Passwort-Policies via `pwquality`, granularen POSIX-ACLs, Speicherplatz-Quotas und einer professionellen Sudoers-Architektur bist du in der Lage, Multi-User-Systeme nach höchsten Unternehmens- und Compliance-Standards abzusichern und zu verwalten.

> **Praxis-Tipp:** Nutze für Sudo-Berechtigungen immer modulare Dateien unter `/etc/sudoers.d/` mit restriktiven Dateirechten (`chmod 440`), anstatt die Hauptdatei `/etc/sudoers` manuell aufzublähen. Das erleichtert das Konfigurationsmanagement via

> Ansible oder GitOps erheblich.

Im nächsten Modul unseres Administrationskurses widmen wir uns der Überwachung und Steuerung laufender Programme und Systemressourcen: **Weiter geht es mit:** Linux Administration #3: Prozesse und Ressourcenverwaltung

Zur Kursübersicht: Alle Linux-Administrations-Artikel & Module