4.9 Benutzer-, Dienstkonto- und Berechtigungsfehler

Jeder Dienst arbeitet in einem bestimmten Sicherheitskontext. Dieser entscheidet, auf welche Dateien, Verzeichnisse, Netzwerkressourcen, Zertifikate, Ports und Betriebssystemfunktionen der Dienst zugreifen darf.

> **Grundsatz:**
> Ein erfolgreicher Zugriff als Administrator beweist nicht, dass das tatsächliche Dienstkonto denselben Zugriff besitzt.

Ziele dieser Seite

Nach dieser Seite sollst du:

- das verwendete Dienstkonto bestimmen können,
- Benutzer, Gruppen und Sicherheitskennungen prüfen können,
- Dateirechte und ACLs auswerten können,
- lokale und entfernte Berechtigungen unterscheiden können,
- fehlende Anmelderechte und gesperrte Konten erkennen können,
- den Zugriff im tatsächlichen Dienstkontext testen können,
- SELinux-, AppArmor- und macOS-Datenschutzbeschränkungen berücksichtigen können,
- Berechtigungsprobleme beheben, ohne unnötig weitreichende Rechte zu vergeben.

---

1. Sicherheitskontext eines Dienstes verstehen

Der Sicherheitskontext kann unter anderem enthalten:

- Benutzerkonto,
- primäre Gruppe,
- zusätzliche Gruppen,
- Sicherheitskennung beziehungsweise UID,
- Zugriffstoken,
- Benutzerrechte und Privilegien,
- Dateisystem-ACLs,
- Netzwerkidentität,
- Integritäts- und Sicherheitsstufe,
- SELinux- oder AppArmor-Kontext,
- systemd-Sandboxing,

- macOS-Datenschutzfreigaben,
- Containerbenutzer und Capabilities.

Bei jedem Zugriff bewertet das System mehrere Faktoren:

```
Dienstkonto
  + Gruppenmitgliedschaften
  + lokale Rechte
  + ACLs des Zielobjekts
  + übergeordnete Verzeichnisrechte
  + Sicherheitsrichtlinien
  + Dienst- oder Containerbeschränkungen
  = effektiver Zugriff
```

---

2. Authentifizierung und Autorisierung unterscheiden

Begriff	Leitfrage	Beispiel
Authentifizierung	Wer ist der Dienst?	Anmeldung als Dienstkonto
Autorisierung	Was darf dieses Konto?	Leserechte auf Konfigurationsdatei
Identifikation	Welches Konto wird verwendet?	UID, SID oder Benutzername
Gruppenmitgliedschaft	Welche Rollen besitzt das Konto?	Mitglied der Datenbankgruppe
Privileg	Welche Systemoperation ist erlaubt?	Binden an privilegierten Port
ACL	Welche Rechte gelten auf einem Objekt?	Schreiben in Datenverzeichnis
Sicherheitsrichtlinie	Welche zusätzliche Kontrolle greift?	SELinux oder macOS TCC

Typische Trennung:

```
Anmeldung fehlgeschlagen
  → Authentifizierungsproblem

Anmeldung erfolgreich, Zugriff verweigert
  → Autorisierungsproblem
```

---

3. Typische Fehlerbilder eines Dienstkontos

Fehlerbild	Mögliche Ursache
Dienst startet nicht	Konto ungültig oder Anmelderecht fehlt
Zugriff verweigert	Datei-, ACL- oder Sicherheitsrichtlinienfehler
Lokaler Zugriff funktioniert, Netzwerkzugriff nicht	Andere Netzwerkidentität oder fehlende Remote-Berechtigung
Manueller Start funktioniert	Administrator besitzt mehr Rechte als Dienstkonto
Dienst funktioniert bis zum Kennwortwechsel	Hinterlegtes Dienstkennwort ist veraltet
Nur eine Datei ist nicht zugänglich	Abweichender Eigentümer oder ACL
Schreiben scheitert, Lesen funktioniert	Schreibrecht auf Datei oder Verzeichnis fehlt
Datei kann gelesen, aber nicht ersetzt werden	Änderungsrecht auf Verzeichnis fehlt
Programm ist ausführbar, startet aber nicht	Bibliothek, Arbeitsverzeichnis oder Sicherheitsrichtlinie blockiert
Zugriff funktioniert nach Anmeldung, aber nicht beim Boot	Benutzersitzung oder Netzlaufwerk fehlt
macOS-Dienst erhält trotz POSIX-Rechten keinen Zugriff	Datenschutzkontrolle oder Sandbox blockiert
Linux-Dienst erhält trotz Modus `777` keinen Zugriff	SELinux, AppArmor, Mountoption oder systemd-Sandboxing

---

4. Dienstkonto unter Windows bestimmen

Dienstkonto und Status anzeigen:

```
[RO] Get-CimInstance Win32_Service -Filter "Name='<DIENSTNAME>'" |
    Select-Object Name,
                  DisplayName,
                  State,
                  StartMode,
                  StartName,
                  ProcessId
```

Alternative mit sc.exe:

```
[RO][SENS] sc.exe qc "<DIENSTNAME>"
```

Relevant ist:

```
SERVICE_START_NAME
```

Typische Windows-Dienstkonten:

Konto	Grundlegende Einordnung
`LocalSystem`	Sehr weitreichende lokale Rechte
`LocalService`	Begrenzte lokale Rechte
`NetworkService`	Begrenzte lokale Rechte, Netzwerkzugriff typischerweise als Computerkonto
Virtuelles Dienstkonto	Dienstbezogene lokale Identität
Lokales Benutzerkonto	Gilt primär auf dem lokalen System
Domänenkonto	Kann abhängig von Berechtigungen auf Domänenressourcen zugreifen
Gruppenverwaltetes Dienstkonto	Durch Active Directory verwaltetes Dienstkonto

> Das Konto sollte nur die für den Dienst benötigten Rechte besitzen. Die Vergabe von Administratorrechten ist keine geeignete Standardlösung für ein Berechtigungsproblem.

---

5. Aktuellen Windows-Benutzerkontext anzeigen

Diese Befehle zeigen den Kontext der aktuellen Sitzung – nicht automatisch den Kontext des Dienstes.

Benutzername anzeigen:

```
[RO] whoami
```

Benutzer und SID anzeigen:

```
[RO] whoami /user
```

Gruppen anzeigen:

```
[RO][SENS] whoami /groups
```

Privilegien anzeigen:

```
[RO][SENS] whoami /priv
```

Gesamten aktuellen Sicherheitskontext anzeigen:

```
[RO][SENS] whoami /all
```

> Die Ausgabe ist als Vergleich hilfreich. Sie darf aber nicht auf das Dienstkonto übertragen werden, wenn der Befehl in einer Administrator- oder Benutzersitzung ausgeführt wurde.

---

6. Lokales Windows-Konto und Gruppen prüfen

Lokales Konto suchen:

```
[RO] Get-LocalUser -Name "<BENUTZER>" -ErrorAction SilentlyContinue
```

Status eines lokalen Kontos anzeigen:

```
[RO][SENS] Get-LocalUser -Name "<BENUTZER>" |
    Select-Object Name,
                  Enabled,
                  LastLogon,
                  PasswordExpires,
                  PasswordRequired,
                  UserMayChangePassword
```

Mitgliedschaften einer lokalen Gruppe anzeigen:

```
[RO][SENS] Get-LocalGroupMember -Group "<GRUPPE>"
```

Alle lokalen Gruppen anzeigen:

```
[RO] Get-LocalGroup
```

Einschränkungen:

- `Get-LocalUser` prüft keine Domänenkonten.
- Domänenkonten benötigen Active-Directory-Werkzeuge oder eine Abfrage durch zuständige Administratoren.
- Gruppenmitgliedschaften können verschachtelt sein.
- Änderungen an Gruppenmitgliedschaften werden nicht in jedem laufenden Zugriffstoken sofort wirksam.
- Ein Dienstneustart kann erforderlich sein, damit ein neues Zugriffstoken erzeugt wird.

---

7. Windows-Dateirechte mit Get-Acl prüfen

ACL einer Datei anzeigen:

```
[RO][FILE][SENS] Get-Acl -LiteralPath "<DATEI>" |
    Format-List
```

Zugriffsregeln übersichtlich anzeigen:

```
[RO][FILE][SENS] (Get-Acl -LiteralPath "<DATEI>").Access |
    Select-Object IdentityReference,
                  FileSystemRights,
                  AccessControlType,
                  IsInherited,
                  InheritanceFlags,
                  PropagationFlags
```

ACL eines Verzeichnisses anzeigen:

```
[RO][FILE][SENS] (Get-Acl -LiteralPath "<VERZEICHNIS>").Access |
    Select-Object IdentityReference,
                  FileSystemRights,
                  AccessControlType,
                  IsInherited
```

Wichtige Felder:

Feld	Bedeutung
`IdentityReference`	Konto oder Gruppe
`FileSystemRights`	Zugewiesene Rechte
`Allow`	Erlaubende Regel
`Deny`	Verweigernde Regel
`IsInherited`	Regel wurde von übergeordnetem Objekt geerbt
`Owner`	Eigentümer des Objekts

> Eine explizite Verweigerung kann eine erlaubende Berechtigung überstimmen. Die effektive Bewertung hängt jedoch von der vollständigen ACL und dem Zugriffstoken ab.

---

8. Windows-Dateirechte mit icacls prüfen

ACL einer Datei oder eines Verzeichnisses anzeigen:

```
[RO][FILE][SENS] icacls "<PFAD>"
```

ACLs unterhalb eines Verzeichnisses anzeigen:

```
[RO][FILE][SENS] icacls "<VERZEICHNIS>" /T /C
```

`/T` durchsucht Unterverzeichnisse rekursiv und kann bei großen Verzeichnisbäumen eine sehr umfangreiche Ausgabe erzeugen.

Typische Rechtekürzel:

Kürzel	Bedeutung
F	Vollzugriff
M	Ändern
RX	Lesen und Ausführen
R	Lesen
W	Schreiben
D	Löschen
N	Kein Zugriff
I	Geerbte Regel

Typische Vererbungskennzeichnungen:

Kürzel	Bedeutung
OI	Vererbung an Dateien
CI	Vererbung an Unterverzeichnisse
IO	Regel gilt nur durch Vererbung
NP	Keine weitere Vererbung

> Auf dieser Diagnoseseite wird `icacls` nur lesend verwendet. Parameter wie `/grant`, `/deny`, `/remove` oder `/reset` verändern Berechtigungen.

---

9. Effektive Windows-Rechte mit AccessChk untersuchen

`AccessChk` ist ein zusätzliches Microsoft-Sysinternals-Werkzeug und nicht Bestandteil jeder Windows-Installation.

Grundlegende Prüfung eines Kontos auf einen bekannten Pfad:

```
[RO][FILE][SENS] accesschk.exe `
    -nobanner `
    -v `
    "<KONTO>" `
    "<PFAD>"
```

Prüfung eines Verzeichnisses:

```
[RO][FILE][SENS] accesschk.exe `
  -nobanner `
  -d `
  -v `
  "<KONTO>" `
  "<VERZEICHNIS>"
```

Vor Verwendung prüfen:

- stammt das Werkzeug direkt von Microsoft Sysinternals?
- ist seine Verwendung im Unternehmen erlaubt?
- wird die passende Werkzeugversion verwendet?
- wird tatsächlich das Dienstkonto geprüft?
- enthält die Ausgabe sensible ACL- und Kontoinformationen?

> Wenn AccessChk nicht vorhanden oder nicht freigegeben ist, wird es nicht ungeprüft heruntergeladen oder auf einem Produktivserver installiert.

---

10. Erforderliche Windows-Dienstprivilegien prüfen

Vom Dienst angeforderte Privilegien anzeigen:

```
[RO] sc.exe qprivs "<DIENSTNAME>"
```

Wiederherstellungs- und Dienstkonfiguration ergänzend prüfen:

```
[RO] sc.exe qfailure "<DIENSTNAME>"
```

```
[RO][SENS] sc.exe qc "<DIENSTNAME>"
```

Wichtig:

- `sc.exe qprivs` zeigt für den Dienst konfigurierte erforderliche Privilegien.

- Es beweist nicht allein, dass das Konto alle notwendigen Objektberechtigungen besitzt.
- Benutzerrechte wie „Anmelden als Dienst“ werden durch lokale oder domänenbasierte Sicherheitsrichtlinien vergeben.
- Domänenrichtlinien können lokale Einstellungen überschreiben.
- Fehlende Rechte erscheinen häufig im System- oder Sicherheitsprotokoll.

---

11. Windows-Anmeldefehler des Dienstkontos untersuchen

Typische Ursachen:

- falsches Kennwort,
- Kennwort abgelaufen,
- Konto deaktiviert,
- Konto gesperrt,
- „Anmelden als Dienst“ fehlt,
- „Anmelden als Dienst verweigern“ greift,
- Domänencontroller nicht erreichbar,
- Zeitabweichung verhindert Authentifizierung,
- verwaltetes Dienstkonto nicht korrekt eingerichtet,
- Dienst verwendet altes hinterlegtes Kennwort.

Service-Control-Manager-Ereignisse prüfen:

```
[RO][SENS] Get-WinEvent -FilterHashtable @{
    LogName      = "System"
    ProviderName = "Service Control Manager"
    StartTime    = (Get-Date).AddHours(-2)
} |
    Select-Object TimeCreated,
                  Id,
                  LevelDisplayName,
                  Message
```

Sicherheitsereignisse erfordern entsprechende Berechtigungen:

```
[RO][SENS][PRIV] Get-WinEvent -FilterHashtable @{
    LogName   = "Security"
    StartTime = (Get-Date).AddHours(-2)
```

```
} |
    Select-Object TimeCreated,
                  Id,
                  ProviderName,
                  Message
```

> Das Sicherheitsprotokoll kann sehr viele sensible Informationen enthalten. Es sollte mit passenden Ereignis-IDs, Zeitfenstern und zuständigen Sicherheitsadministratoren ausgewertet werden.

---

12. Windows-Netzwerkidentität des Dienstkontos berücksichtigen

Der lokale Kontoname ist nicht zwingend die Identität, die ein entfernter Server sieht.

Dienstkonto	Mögliche Netzwerkidentität
Lokales Benutzerkonto	Auf entferntem System nicht automatisch bekannt
`LocalService`	Netzwerkzugriff typischerweise anonym beziehungsweise sehr begrenzt
`NetworkService`	Zugriff im Domänenumfeld typischerweise als Computerkonto
`LocalSystem`	Zugriff im Domänenumfeld typischerweise als Computerkonto
Domänenkonto	Eigenes Domänenkonto
gMSA	Verwaltete Domänenidentität

Prüffragen:

- Welche Identität sieht der Fileserver oder Datenbankserver?
- besitzt das Computerkonto Zugriff?
- besitzt nur der angemeldete Administrator Zugriff?
- wird Kerberos oder NTLM verwendet?
- stimmt der verwendete Servername mit dem erwarteten Dienstprinzipal überein?
- wird ein Laufwerksbuchstabe statt eines UNC-Pfads verwendet?
- ist das Dienstkonto auf der Zielressource berechtigt?
- existiert eine lokale und eine Freigabeberechtigung?

> Bei einer Windows-Freigabe müssen sowohl die Freigabeberechtigung als auch die NTFS-Berechtigung den benötigten Zugriff erlauben.

---

13. Dienstkonto unter Linux bestimmen

systemd-Dienstkonto anzeigen:

```
[RO] systemctl show "<DIENST>" \
  --property=User,Group,DynamicUser,SupplementaryGroups
```

Wirksame Unit prüfen:

```
[RO][FILE][SENS] systemctl cat "<DIENST>"
```

Benutzerkonto auflösen:

```
[RO] getent passwd "<DIENSTBENUTZER>"
```

Gruppenkonto auflösen:

```
[RO] getent group "<DIENSTGRUPPE>"
```

UID, primäre und zusätzliche Gruppen anzeigen:

```
[RO] id "<DIENSTBENUTZER>"
```

Wichtig:

- eine leere `User=`-Eigenschaft bedeutet bei einem Systemdienst normalerweise, dass keine abweichende Benutzeridentität konfiguriert wurde,
- `DynamicUser=yes` erzeugt eine dynamisch verwaltete Identität,
- zusätzliche Gruppen können über `SupplementaryGroups=` gesetzt werden,
- ein Container kann innerhalb seines Namespaces andere UID-Zuordnungen verwenden.

---

14. POSIX-Dateirechte unter Linux verstehen

Beispiel:

```
-rwxr-x--- 1 appuser appgroup 4096 Jul 31 09:00 application
```

Interpretation:

Bereich	Bedeutung
`-`	Reguläre Datei
`rwx`	Eigentümer darf lesen, schreiben und ausführen
`r-x`	Gruppe darf lesen und ausführen
`---`	Andere besitzen keine Rechte
`appuser`	Eigentümer
`appgroup`	Gruppe

Rechte:

Recht	Datei	Verzeichnis
`r`	Dateiinhalt lesen	Verzeichnisinhalt auflisten
`w`	Dateiinhalt verändern	Einträge anlegen oder entfernen
`x`	Datei ausführen	Verzeichnis durchlaufen und Objekte erreichen

> Schreibrecht auf einer Datei und Schreibrecht auf dem übergeordneten Verzeichnis sind unterschiedliche Berechtigungen. Das Ersetzen oder Löschen einer Datei hängt häufig vom Verzeichnisrecht ab.

---

15. Linux-Dateirechte und Pfadbestandteile prüfen

Dateirechte anzeigen:

```
[RO][FILE] ls -ld "<DATEI>"
```

Numerische und symbolische Rechte anzeigen:

```
[RO][FILE] stat -c '%U %G %A %a %n' "<DATEI>"
```

Alle Bestandteile eines Pfads prüfen:

```
[RO][FILE] namei -l "<DATEI>"
```

Verzeichnis und Zieldatei gemeinsam prüfen:

```
[RO][FILE] ls -ld "<VERZEICHNIS>" "<DATEI>"
```

Prüffragen:

- darf das Dienstkonto jedes übergeordnete Verzeichnis durchlaufen?
- darf es die Datei lesen?
- darf es das Verzeichnis beschreiben?
- stimmen Eigentümer und Gruppe?
- greift das Gruppenrecht wirklich, weil das Konto Mitglied ist?
- ist das Dateisystem schreibgeschützt?
- verhindern Mountoptionen die Ausführung?
- handelt es sich um einen symbolischen Link?

---

16. Linux-ACLs prüfen

ACL einer Datei anzeigen:

```
[RO][FILE][SENS] getfacl "<DATEI>"
```

ACL eines Verzeichnisses anzeigen:

```
[RO][FILE][SENS] getfacl "<VERZEICHNIS>"
```

Mögliche Ausgabe:

```
user::rw-
user:appuser:r--
group::r--
mask::r--
other::---
```

Wichtige Elemente:

Eintrag	Bedeutung
`user::`	Rechte des Eigentümers
`user:name:`	Rechte eines bestimmten Benutzers
`group::`	Rechte der Eigentümergruppe
`group:name:`	Rechte einer bestimmten Gruppe
`mask::`	Maximale wirksame Rechte benannter Benutzer und Gruppen
`other::`	Rechte aller übrigen Benutzer
`default:`	Standard-ACL für neu erstellte Unterobjekte

> Die ACL-Maske kann weitergehende angezeigte Einzelrechte begrenzen. Deshalb müssen sowohl Eintrag als auch `mask` bewertet werden.

---

17. Linux-Zugriff im tatsächlichen Benutzerkontext testen

Die folgenden Tests lesen oder verändern die Zieldatei nicht. Sie prüfen nur, ob der angegebene Zugriff laut Betriebssystem möglich ist.

Leserecht prüfen:

```
[TEST][PRIV] sudo -u "<DIENSTBENUTZER>" \
  test -r "<DATEI>"
```

Schreibrecht prüfen:

```
[TEST][PRIV] sudo -u "<DIENSTBENUTZER>" \
  test -w "<DATEI>"
```

Ausführungsrecht prüfen:

```
[TEST][PRIV] sudo -u "<DIENSTBENUTZER>" \
  test -x "<AUSFÜHRBARE_DATEI>"
```

Verzeichniszugriff prüfen:

```
[TEST][PRIV] sudo -u "<DIENSTBENUTZER>" \
  test -x "<VERZEICHNIS>"
```

Rückgabecode unmittelbar anzeigen:

```
[TEST][PRIV] sudo -u "<DIENSTBENUTZER>" \
  test -r "<DATEI>"

echo $?
```

Rückgabecode	Bedeutung
`0`	Bedingung erfüllt
ungleich `0`	Bedingung nicht erfüllt oder Prüfung nicht möglich

Einschränkung:

Ein erfolgreicher `test -r`- oder `test -w`-Befehl beweist nicht, dass SELinux, AppArmor, Anwendungssandboxing oder produktspezifische Regeln jeden späteren Zugriff erlauben.

---

18. Linux-Sonderrechte und Mountoptionen berücksichtigen

Mountoptionen des Zielpfads anzeigen:

```
[RO][FILE] findmnt --target "<PFAD>"
```

Mögliche relevante Optionen:

Option	Bedeutung
`ro`	Dateisystem ist schreibgeschützt
`rw`	Dateisystem ist beschreibbar
`noexec`	Direkte Ausführung von Dateien auf diesem Dateisystem wird verhindert
`nosuid`	Setuid- und Setgid-Wirkung wird eingeschränkt
`nodev`	Gerätedateien werden nicht interpretiert

Dateirechte einschließlich Sonderbits anzeigen:

```
[RO][FILE] stat -c '%A %a %U %G %n' "<DATEI>"
```

Mögliche Sonderbits:

- Setuid,
- Setgid,
- Sticky Bit.

> Sonderrechte dürfen nicht als schnelle Problembehebung gesetzt werden. Eine falsche Verwendung kann erhebliche Sicherheitsrisiken erzeugen.

---

19. SELinux als zusätzliche Zugriffskontrolle prüfen

Diese Befehle sind nur auf Systemen mit installiertem SELinux-Werkzeug relevant.

SELinux-Status anzeigen:

```
[RO] getenforce
```

Mögliche Ergebnisse:

Zustand	Bedeutung
`Enforcing`	Richtlinien werden durchgesetzt

Zustand	Bedeutung
Permissive	Verstöße werden protokolliert, aber nicht blockiert
Disabled	SELinux ist deaktiviert

SELinux-Kontext einer Datei anzeigen:

```
[RO][FILE][SENS] ls -lZ "<DATEI>"
```

Prozesskontext anzeigen:

```
[RO][SENS] ps -eZ |
    grep -- "<PROZESS>"
```

Aktuelle AVC-Verweigerungen suchen:

```
[RO][SENS][PRIV] sudo ausearch -m AVC -ts recent
```

Wichtig:

- klassische Dateirechte können korrekt sein, während SELinux blockiert,
- ein falscher Dateikontext kann nach manuellem Kopieren entstehen,
- SELinux sollte nicht pauschal deaktiviert werden,
- eine Richtlinienänderung muss auf den tatsächlich benötigten Zugriff begrenzt sein,
- automatisch erzeugte Freigaberegeln müssen fachlich und sicherheitstechnisch geprüft werden.

---

20. AppArmor als zusätzliche Zugriffskontrolle prüfen

Diese Befehle gelten nur, wenn AppArmor installiert und aktiv ist.

AppArmor-Status anzeigen:

```
[RO][PRIV] sudo aa-status
```

Kernel- und Systemmeldungen nach Verweigerungen durchsuchen:

```
[RO][SENS][PRIV] sudo journalctl \
  -k \
  --since "1 hour ago" |
    grep -i -- "apparmor"
```

Mögliche Zustände:

Zustand	Bedeutung
Enforce	Profilregeln werden durchgesetzt
Complain	Verstöße werden protokolliert, aber normalerweise nicht blockiert
Unconfined	Prozess wird nicht durch ein AppArmor-Profil eingeschränkt

> AppArmor darf nicht pauschal deaktiviert werden. Zuerst müssen Profil, verweigerter Pfad und tatsächlich erforderlicher Zugriff bestimmt werden.

---

21. systemd-Sandboxing und Capabilities prüfen

Auch ohne SELinux oder AppArmor kann systemd einen Dienst zusätzlich einschränken.

Sicherheitsrelevante Eigenschaften anzeigen:

```
[RO][SENS] systemctl show "<DIENST>" \
  --
property=NoNewPrivileges,ProtectSystem,ProtectHome,PrivateTmp,PrivateDevices,ReadOnlyPaths,ReadWr
itePaths,InaccessiblePaths,CapabilityBoundingSet,AmbientCapabilities
```

Mögliche Einschränkungen:

Eigenschaft	Mögliche Wirkung
`ProtectSystem=`	Teile des Dateisystems werden schreibgeschützt
`ProtectHome=`	Zugriff auf Benutzerverzeichnisse wird eingeschränkt
`PrivateTmp=`	Dienst erhält eigenes temporäres Verzeichnis

Eigenschaft	Mögliche Wirkung
`PrivateDevices=`	Zugriff auf Geräte wird eingeschränkt
`ReadOnlyPaths=`	Bestimmte Pfade werden schreibgeschützt
`ReadWritePaths=`	Ausgewählte Pfade werden beschreibbar gemacht
`InaccessiblePaths=`	Pfade werden unzugänglich
`NoNewPrivileges=`	Erwerb neuer Privilegien wird verhindert
`CapabilityBoundingSet=`	Verfügbare Linux-Capabilities werden begrenzt

> Ein erfolgreicher Zugriff aus einer normalen Shell beweist nicht, dass derselbe Pfad innerhalb der systemd-Sandbox verfügbar ist.

---

22. Dienstkonto unter macOS bestimmen

Systemweiten launchd-Job anzeigen:

```
[RO] launchctl print "system/<LABEL>"
```

launchd-plist anzeigen:

```
[RO][FILE][SENS] plutil -p "/Library/LaunchDaemons/<LABEL>.plist"
```

Relevante Schlüssel:

```
UserName
GroupName
Program
ProgramArguments
WorkingDirectory
EnvironmentVariables
```

Falls `UserName` bei einem systemweiten LaunchDaemon nicht gesetzt ist, muss der tatsächliche Ausführungskontext anhand von launchd-Konfiguration und laufendem Prozess geprüft werden.

Ausführungskonto eines laufenden Prozesses anzeigen:

```
[RO][SENS] ps -p <PID> \
  -o user,uid,gid,pid,ppid,command
```

---

23. Benutzer und Gruppen unter macOS prüfen

UID, GID und Gruppen eines Kontos anzeigen:

```
[RO] id "<BENUTZER>"
```

Lokales Benutzerkonto über Directory Service anzeigen:

```
[RO][SENS] dscl . -read "/Users/<BENUTZER>"
```

Bestimmte lokale Gruppe anzeigen:

```
[RO][SENS] dscl . -read "/Groups/<GRUPPE>"
```

Gruppenmitgliedschaft prüfen:

```
[RO] dsmemberutil checkmembership \
  -U "<BENUTZER>" \
  -G "<GRUPPE>"
```

Hinweise:

- `dscl .` bezieht sich auf den lokalen Verzeichnisknoten,
- Netzwerk- oder Verzeichnisdienstkonten können weitere Werkzeuge erfordern,
- Gruppenmitgliedschaften können aus mehreren Verzeichnisquellen stammen,
- der aktuelle Prozess kann noch ein älteres Gruppen- und Berechtigungstoken verwenden.

---

24. POSIX-Rechte und ACLs unter macOS prüfen

Rechte, Eigentümer und ACL anzeigen:

```
[RO][FILE][SENS] ls -lde "<DATEI>"
```

Erweiterte Attribute zusätzlich anzeigen:

```
[RO][FILE][SENS] ls -lde@ "<DATEI>"
```

Dateiflags anzeigen:

```
[RO][FILE][SENS] ls -ldeO "<DATEI>"
```

Metadaten strukturiert anzeigen:

```
[RO][FILE] stat -f '%Su %Sg %Sp %N' "<DATEI>"
```

Mögliche zusätzliche Einflussfaktoren:

- POSIX-Rechte,
- ACL-Einträge,
- erweiterte Attribute,
- Dateiflags,
- schreibgeschütztes Volume,
- Datenschutzkontrollen,
- Sandbox,
- System Integrity Protection.

> Ein `+` in der Ausgabe von `ls -l` weist auf ACL-Einträge hin. Ein `@` weist auf erweiterte Attribute hin.

---

25. Zugriff unter macOS im Benutzerkontext testen

Leserecht prüfen:

```
[TEST][PRIV] sudo -u "<DIENSTBENUTZER>" \
  test -r "<DATEI>"
```

Schreibrecht prüfen:

```
[TEST][PRIV] sudo -u "<DIENSTBENUTZER>" \
  test -w "<DATEI>"
```

Ausführungsrecht prüfen:

```
[TEST][PRIV] sudo -u "<DIENSTBENUTZER>" \
  test -x "<AUSFÜHRBARE_DATEI>"
```

Rückgabecode anzeigen:

```
[TEST][PRIV] sudo -u "<DIENSTBENUTZER>" \
  test -r "<DATEI>"

echo $?
```

> Dieser Test bildet nicht automatisch den vollständigen launchd-, Sandbox- oder Datenschutzkontext eines Dienstes nach.

---

26. macOS-Datenschutzkontrollen berücksichtigen

macOS schützt unter anderem Zugriffe auf:

- Schreibtisch,
- Dokumente,
- Downloads,
- iCloud Drive,
- Netzwerkvolumes,
- Wechselmedien,
- Daten anderer Anwendungen,
- Kontakte und Kalender,
- Kamera und Mikrofon,
- Automation,
- Bedienungshilfen,
- vollständigen Festplattenzugriff.

Ein Zugriff kann trotz korrekter POSIX-Rechte und ACLs verweigert werden.

Protokolle nach dem Prozess durchsuchen:

```
[RO][SENS][PRIV] sudo log show \
  --last 1h \
  --predicate 'process == "<PROZESS>"' \
  --style compact \
  --no-pager
```

Nach möglichen Zugriffsverweigerungen suchen:

```
[RO][SENS][PRIV] sudo log show \
  --last 1h \
  --predicate 'process == "<PROZESS>" AND (messageType == error OR messageType == fault)' \
  --style compact \
  --no-pager
```

Wichtig:

- Datenschutzfreigaben werden über Systemeinstellungen oder Unternehmensverwaltung vergeben,
- Full Disk Access ist eine weitreichende Berechtigung,
- Freigaben dürfen nur vertrauenswürdigen und tatsächlich benötigten Programmen erteilt werden,
- die Datenschutzdatenbank darf nicht manuell manipuliert werden,
- ein Terminaltest kann andere Datenschutzrechte als der eigentliche Dienst besitzen.

---

27. macOS System Integrity Protection und Dateiflags berücksichtigen

SIP-Status anzeigen:

```
[RO] csrutil status
```

System Integrity Protection schützt bestimmte Systembereiche auch vor weitreichenden Benutzerkonten.

Dateiflags prüfen:

```
[RO][FILE] ls -ldeO "<DATEI>"
```

Mögliche Einflussfaktoren:

- geschützter Systempfad,
- unveränderliches Dateiflag,
- schreibgeschütztes Systemvolume,
- signierter Systembestandteil,
- Sandbox- oder Datenschutzregel.

> SIP darf nicht als allgemeine Fehlerbehebung deaktiviert werden. Zuerst muss geklärt werden, warum eine Anwendung auf einen geschützten Systembereich zugreifen möchte.

---

28. Netzwerkfreigaben und entfernte Ressourcen prüfen

Bei einer entfernten Ressource existieren mindestens zwei Berechtigungsebenen:

```
Lokaler Dienstkontext
  → Netzwerkidentität
  → Authentifizierung am Ziel
  → Freigabe- oder Dienstberechtigung
  → Dateisystem- oder Objektberechtigung
```

Prüffragen:

- welche Identität wird am Zielsystem verwendet?
- existiert das Konto dort?
- ist das Konto gesperrt oder abgelaufen?
- besitzt es Freigabe- und Dateisystemrechte?
- ist Kerberos, NTLM, LDAP oder Zertifikatsauthentifizierung vorgesehen?
- funktioniert nur interaktiver Zugriff?
- wird ein benutzerbezogener Mount verwendet?
- ist das Ziel beim Systemstart bereits verfügbar?
- ist das verwendete Secret aktuell?
- protokolliert das Zielsystem den abgelehnten Zugriff?

>

> Die Berechtigungsprüfung muss auch auf dem Zielsystem erfolgen. Der Quellserver kann häufig nur feststellen, dass der Zugriff abgelehnt wurde.

---

29. Containerbenutzer und Hostrechte berücksichtigen

Ein Containerprozess kann innerhalb und außerhalb des Containers unterschiedliche Identitätsdarstellungen besitzen.

Zu prüfen sind:

- UID und GID im Container,
- UID- und GID-Zuordnung auf dem Host,
- Eigentümer des eingebundenen Volumes,
- Rootless- oder Rootful-Betrieb,
- User-Namespace,
- Container-Capabilities,
- schreibgeschützte Mounts,
- SELinux-Label des Volumes,
- Secrets und Config-Mounts,
- Netzwerkidentität des Dienstes.

Typisches Fehlerbild:

```
Datei auf dem Host gehört UID 1000
Containerprozess läuft als UID 1001
Volume ist nur für UID 1000 beschreibbar
Anwendung meldet „Permission denied“
```

> Ein Wechsel des Containers auf Benutzer `root` ist keine geeignete Standardlösung. Zuerst müssen vorgesehene UID, GID und Volume-Berechtigungen aus der Image- beziehungsweise Herstellerdokumentation ermittelt werden.

---

30. Leserecht, Schreibrecht und Änderungsrecht praktisch unterscheiden

Gewünschte Aktion	Typischer benötigter Zugriff
Dateiinhalt lesen	Leserecht auf Datei und Durchlaufrecht auf Verzeichnisse
Dateiinhalt verändern	Schreibrecht auf Datei
Neue Datei anlegen	Schreib- und Durchlaufrecht auf Verzeichnis
Datei ersetzen	Rechte auf Datei und beziehungsweise oder Verzeichnis, abhängig vom Verfahren
Datei löschen	Lösch- beziehungsweise Verzeichnisrecht
Programm ausführen	Ausführungsrecht und Zugriff auf Bibliotheken
Verzeichnis auflisten	Leserecht auf Verzeichnis
Pfad durchlaufen	Ausführungsrecht auf Verzeichnis
Logdatei erzeugen	Schreibrecht auf Zielverzeichnis
Socketdatei erstellen	Schreibrecht auf Laufzeitverzeichnis

> Viele Anwendungen speichern eine Konfiguration, indem sie eine neue Datei erzeugen und anschließend die alte Datei ersetzen. Dafür kann Schreibrecht auf der ursprünglichen Datei allein unzureichend sein.

---

31. Berechtigungsproblem sicher reproduzieren

Ein geeigneter Test verändert möglichst wenig und verwendet das tatsächliche Konto.

Testablauf:

1. Dienstkonto bestimmen
2. Zielobjekt und benötigte Aktion bestimmen
3. Aktuelle Rechte und ACLs sichern
4. Sicherheitsrichtlinien prüfen
5. Lesenden Zugriff im Dienstkontext testen
6. Protokolle auf Quelle und Ziel beobachten
7. Nur bei Bedarf einen freigegebenen Schreibtest verwenden
8. Ergebnis und Rückgabecode dokumentieren
9. Testdateien vollständig entfernen
10. Keine Rechte auf Verdacht erweitern

Ungeeigneter Test:

```
Als Administrator Datei öffnen
```

Besserer Test:

```
Mit tatsächlicher Dienstidentität exakt den benötigten Zugriff prüfen
```

---

32. Warum chmod 777 oder Vollzugriff keine geeignete Diagnose ist

Eine pauschale Rechteerweiterung:

- verschleiert die ursprünglich fehlende Einzelberechtigung,
- kann vertrauliche Daten offenlegen,
- erhöht das Risiko einer Manipulation,
- kann Sicherheitsrichtlinien verletzen,
- behebt SELinux, AppArmor oder TCC möglicherweise trotzdem nicht,
- kann nach Updates oder Wiederherstellungen bestehen bleiben,
- erschwert die spätere Rücknahme,
- widerspricht dem Prinzip der minimalen Rechte.

Besseres Vorgehen:

1. benötigte Aktion bestimmen,
2. tatsächliches Dienstkonto bestimmen,
3. aktuell fehlende Einzelberechtigung bestimmen,
4. vorgesehene Herstellerberechtigung prüfen,
5. kleinste ausreichende Änderung planen,
6. Änderung dokumentieren,
7. Funktion testen,
8. überflüssige Rechte ausschließen.

---

33. Gruppenmitgliedschaft richtig bewerten

Eine neue Gruppenmitgliedschaft ist nicht immer sofort in jedem Prozess wirksam.

Mögliche Gründe:

- laufender Prozess besitzt ein altes Zugriffstoken,
- Benutzer muss sich neu anmelden,
- Dienst muss kontrolliert neu gestartet werden,
- Gruppenmitgliedschaft wird zwischengespeichert,
- verschachtelte Gruppe wird verzögert ausgewertet,
- Domänencontroller sind noch nicht repliziert,
- Container besitzt eigene Benutzer- und Gruppendatenbank.

Prüfung:

```
Konto Mitglied der Gruppe?
  ↓
Gruppe auf Zielobjekt berechtigt?
  ↓
ACL-Maske oder Deny-Regel berücksichtigt?
  ↓
Aktuelles Zugriffstoken enthält die Gruppe?
  ↓
Dienstprozess nach Änderung neu authentifiziert?
```

> Ein Neustart darf erst nach Erfassung des Ausgangszustands und Prüfung der Auswirkungen erfolgen.

---

34. Typische Fehlinterpretationen

Fehlinterpretation	Richtige Bewertung
„Administrator kann zugreifen, also stimmen die Rechte.“	Dienstkonto besitzt einen anderen Sicherheitskontext
„Die Datei hat Leserecht, also kann der Dienst sie öffnen.“	Übergeordnete Verzeichnisse und Sicherheitsrichtlinien zählen ebenfalls
„Der Benutzer steht in der ACL, also ist der Zugriff erlaubt.“	Deny-Regeln, Gruppen und ACL-Masken können das Ergebnis verändern
„Linux-Modus 777 erlaubt immer alles.“	SELinux, AppArmor, Mountoptionen und Sandbox können weiter blockieren
„Root darf auf jeden macOS-Pfad zugreifen.“	SIP und Datenschutzkontrollen können zusätzliche Grenzen setzen

Fehlinterpretation	Richtige Bewertung
„Lokales Dienstkonto verwendet denselben Namen am Fileserver.“	Die Netzwerkidentität kann eine andere sein
„Gruppenänderung ist sofort wirksam.“	Laufender Prozess kann ein altes Token besitzen
„Schreibrecht auf Datei erlaubt das Ersetzen der Datei.“	Dafür können Rechte auf dem Verzeichnis erforderlich sein
„Full Disk Access ist die einfachste Lösung.“	Es ist eine sehr weitreichende Sicherheitsfreigabe
„Der Dienst benötigt Administratorrechte.“	Meist muss nur eine konkrete Ressource passend berechtigt werden

35. Checkliste zur Berechtigungsanalyse

- [ ] Tatsächliches Dienstkonto bestimmt
- [ ] Benutzerkonto existiert und ist aktiviert
- [ ] Kontosperre und Kennwortstatus geprüft
- [ ] Primäre und zusätzliche Gruppen erfasst
- [ ] UID, GID beziehungsweise SID dokumentiert
- [ ] Dienstanmelderecht geprüft
- [ ] Verweigernde Richtlinien berücksichtigt
- [ ] Zieldatei und benötigte Aktion bestimmt
- [ ] Rechte aller Pfadbestandteile geprüft
- [ ] Eigentümer und Gruppe geprüft
- [ ] ACLs und Vererbung geprüft
- [ ] ACL-Maske beziehungsweise Deny-Regeln berücksichtigt
- [ ] Dateisystem-Mountoptionen geprüft
- [ ] Zugriff im tatsächlichen Konto getestet
- [ ] Lokalen und entfernten Zugriff getrennt geprüft
- [ ] Netzwerkidentität des Dienstes bestimmt
- [ ] Zielsystemprotokoll geprüft
- [ ] SELinux beziehungsweise AppArmor berücksichtigt
- [ ] systemd-Sandboxing berücksichtigt
- [ ] macOS-Datenschutz und SIP berücksichtigt
- [ ] Container-UID und Volume-Rechte berücksichtigt
- [ ] Keine pauschale Rechteerweiterung durchgeführt
- [ ] Kleinste notwendige Änderung bestimmt
- [ ] Ausgangsberechtigungen vor Änderung gesichert

```
[ ] Funktion und Sicherheit nach Änderung geprüft
```

Bewertung des Ergebnisses

Ergebnis	Nächster Schritt
Dienstkonto ungültig oder gesperrt	Kontostatus und Verwaltungsprozess prüfen
Dienstanmeldung schlägt fehl	Kennwort, Anmelderecht und Richtlinien prüfen
Datei kann nicht gelesen werden	Pfad-, Datei- und ACL-Rechte prüfen
Datei lesbar, aber nicht schreibbar	Datei- und Verzeichnisrechte getrennt prüfen
Lokaler Zugriff funktioniert, Remotezugriff nicht	Netzwerkidentität und Zielberechtigung prüfen
Klassische Rechte stimmen, Zugriff bleibt verweigert	SELinux, AppArmor, Sandbox oder macOS-Datenschutz prüfen
Gruppenrecht fehlt im laufenden Prozess	Token- beziehungsweise Prozessneuerstellung planen
Container kann Volume nicht beschreiben	UID-, GID-, Mount- und Sicherheitslabel prüfen
Zugriff funktioniert nur als Administrator	Fehlende Einzelberechtigung des Dienstkontos bestimmen
Alle Berechtigungen stimmen	Ressourcen, Konfiguration und Anwendungslogik untersuchen

---

Merksatz

> **Berechtigungen werden nicht danach bewertet, was ein Administrator darf, sondern danach, was das tatsächliche Dienstkonto im echten Laufzeit- und Zielsystemkontext darf.**

Weiterführende Quellen

- Microsoft Learn – Service User Accounts
- Microsoft Learn – About Service Logon Accounts
- Microsoft Learn – whoami
- Microsoft Learn – Get-Acl
- Microsoft Learn – icacls
- Microsoft Sysinternals – AccessChk
- Microsoft Learn – sc.exe qprivs
- Linux-Handbuch – credentials

- Linux-Handbuch – path_resolution
- Linux-Handbuch – acl
- Linux-Handbuch – getfacl
- systemd – systemd.exec
- Apple Platform Security – Zugriff von Apps auf Dateien
- Apple Support – Einstellungen für Datenschutz und Sicherheit
- Apple Support – Dateiberechtigungen und umask
- Apple – launchd.plist-Handbuchseite