5.4 Linux – Werkzeuge zur Fehleranalyse

Linux-Systeme stellen zahlreiche integrierte Werkzeuge für die Fehleranalyse bereit. Die konkrete Verfügbarkeit und Ausgabe hängt unter anderem ab von:

- Distribution und Version
- Kernel-Version
- Init-System
- installierten Paketen
- Netzwerkverwaltung
- Dateisystem
- Virtualisierung oder Containerisierung
- Berechtigungen
- Sicherheitsmechanismen wie SELinux oder AppArmor

Auf modernen Serverdistributionen wird häufig `systemd` eingesetzt. Andere Systeme können jedoch alternative Init- und Protokollierungssysteme verwenden.

Kennzeichnung der Befehle

Kennzeichnung	Bedeutung
`[RO]`	grundsätzlich lesende Abfrage
`[TEST]`	aktiver Test, der Netzwerk- oder Systemlast erzeugen kann
`[CHANGE]`	verändert den Systemzustand
`[PRIV]`	benötigt häufig Root-Rechte oder `sudo`
`[REMOTE]`	kommuniziert mit einem anderen System oder Dienst
`[LOG]`	erzeugt möglicherweise umfangreiche Protokolldaten
`[SENS]`	Ausgabe kann interne oder personenbezogene Informationen enthalten
`[DANGER]`	kann bei falscher Anwendung Dienste, Daten oder das gesamte System gefährden
`[RESTART]`	kann einen Dienst- oder Systemneustart auslösen

> Die Kennzeichnungen sind Sicherheitshinweise und kein Bestandteil des eigentlichen Befehls.

---

1. Diagnose vor der Änderung

Eine Linux-Fehleranalyse sollte nicht mit einem Neustart, einer Paketaktualisierung oder dem Löschen von Dateien beginnen. Zuerst muss der fehlerhafte Zustand gesichert werden.

Grundfragen

- Was funktioniert nicht?
- Wie lautet die vollständige Fehlermeldung?
- Seit wann besteht der Fehler?
- Ist er reproduzierbar?
- Welche Benutzer, Dienste oder Systeme sind betroffen?
- Was wurde zuletzt geändert?
- Ist nur eine Anwendung oder das gesamte System betroffen?
- Tritt der Fehler lokal und remote auf?
- Besteht ein zeitlicher Zusammenhang mit Updates, Neustarts oder Sicherungen?
- Gibt es ausreichend Speicherplatz, Inodes und Arbeitsspeicher?
- Läuft das System auf Hardware, in einer VM oder in einem Container?

Grundlegende Systeminformationen

```
[RO][SENS] hostnamectl
[RO] uname -a
[RO] cat /etc/os-release
[RO] uptime
[RO][SENS] who
[RO][SENS] last -x
```

`last -x` kann unter anderem Anmeldungen, Neustarts und Herunterfahrvorgänge anzeigen. Die verfügbaren Daten hängen von der Protokollierung und Rotation der zugrunde liegenden Dateien ab.

2. Hilfe und Dokumentation direkt auf dem System verwenden

Vor der Verwendung unbekannter Optionen sollte die lokal installierte Dokumentation geprüft werden.

```
[RO] man <BEFEHL>
[RO] info <BEFEHL>
[RO] <BEFEHL> --help
```

```
[RO] apropos "<SUCHBEGRIFF>"
[RO] man -k "<SUCHBEGRIFF>"
[RO] type <BEFEHL>
[RO] command -V <BEFEHL>
```

Warum die lokale Dokumentation wichtig ist

- Sie passt normalerweise zur installierten Programmversion.
- Optionen können sich zwischen Versionen unterscheiden.
- Distributionen können Programme anders konfigurieren.
- Nicht jeder im Internet gefundene Befehl ist auf dem Zielsystem vorhanden.
- Gleichnamige Programme können aus unterschiedlichen Paketen stammen.

> Befehle aus fremden Anleitungen dürfen nicht ungeprüft mit Root-Rechten ausgeführt werden.

3. Systemzustand und Auslastung überblicken

```
[RO] uptime
[RO] top
[RO] free -h
[RO] vmstat 1 10
[RO] cat /proc/loadavg
[RO] nproc
[RO] lscpu
```

Falls installiert:

```
[RO] htop
[RO] mpstat -P ALL 1 10
[RO] pidstat 1 10
```

Load Average richtig einordnen

Die drei Load-Average-Werte beziehen sich üblicherweise auf ungefähr 1, 5 und 15 Minuten. Sie erfassen nicht ausschließlich CPU-Auslastung, sondern unter Linux auch Tasks, die nicht unterbrechbar warten, beispielsweise auf bestimmte I/O-Vorgänge.

Die Bewertung muss deshalb berücksichtigen:

- Anzahl der logischen CPUs
- CPU-Auslastung
- I/O-Wartezeiten
- Prozesszustände
- Dauer der Belastung
- Normalzustand des Systems
- virtuelle CPU-Zuteilung
- mögliche Drosselung

> Ein hoher Load-Wert beweist allein weder eine CPU-Überlastung noch einen bestimmten Hardwarefehler.

4. Prozesse untersuchen

```
[RO][SENS] ps aux
[RO][SENS] ps -ef
[RO][SENS] ps -eo pid,ppid,user,stat,%cpu,%mem,etime,cmd
[RO][SENS] pstree -ap
[RO] pgrep -a "<NAME>"
[RO][PRIV][SENS] lsof -p <PID>
```

Prozesszustände

Zustand	Grundbedeutung
R	läuft oder ist ausführbar
S	unterbrechbarer Schlafzustand
D	nicht unterbrechbarer Schlafzustand, häufig im Zusammenhang mit I/O
T	angehalten oder verfolgt
Z	Zombieprozess

Ein Zombieprozess hat seine Ausführung beendet, wurde aber vom Elternprozess noch nicht vollständig abgeholt. Das Beenden des Zombies selbst löst die Ursache normalerweise nicht; der Elternprozess muss untersucht werden.

Signale kontrolliert verwenden

```
[CHANGE] kill -TERM <PID>
[CHANGE] kill -HUP <PID>
[CHANGE][DANGER] kill -KILL <PID>
```

- `SIGTERM` fordert einen kontrollierten Abbruch an.
- `SIGHUP` kann abhängig vom Programm eine Konfiguration neu einlesen oder eine andere Bedeutung besitzen.
- `SIGKILL` beendet einen Prozess ohne eigene Aufräumlogik des Prozesses.

> `kill -9` sollte nicht als Standardmaßnahme verwendet werden. Offene Transaktionen, temporäre Dateien oder gemeinsam genutzte Ressourcen können in einem problematischen Zustand verbleiben.

5. Systemd-Dienste prüfen

```
[RO] systemctl status <DIENST>
[RO] systemctl is-active <DIENST>
[RO] systemctl is-enabled <DIENST>
[RO] systemctl is-failed <DIENST>
[RO] systemctl list-units --type=service --state=failed
[RO] systemctl list-dependencies <DIENST>
[RO] systemctl show <DIENST>
[RO] systemctl cat <DIENST>
```

Wichtige Unterscheidung

Zustand	Aussage
`active`	Unit ist entsprechend ihrem Typ aktiv
`inactive`	Unit ist derzeit nicht aktiv
`failed`	Start oder Ausführung ist fehlgeschlagen
`enabled`	Unit soll über ihre Installationsverknüpfungen automatisch eingebunden werden
`disabled`	entsprechende automatische Einbindung ist nicht aktiviert

Zustand	Aussage
`masked`	Aktivierung und Start sind durch eine Verknüpfung auf `/dev/null` blockiert

`active` beweist nicht automatisch, dass die Anwendung fachlich funktioniert. Ein Webserver kann beispielsweise laufen, während eine abhängige Datenbank oder eine bestimmte Website nicht erreichbar ist.

Ändernde Aktionen

```
[CHANGE][PRIV] sudo systemctl start <DIENST>
[CHANGE][PRIV] sudo systemctl stop <DIENST>
[CHANGE][PRIV][RESTART] sudo systemctl restart <DIENST>
[CHANGE][PRIV] sudo systemctl reload <DIENST>
[CHANGE][PRIV] sudo systemctl reset-failed <DIENST>
```

`reload` funktioniert nur, wenn die Unit beziehungsweise das Programm dies unterstützt. `reset-failed` löscht den registrierten Fehlerzustand, behebt aber nicht dessen Ursache.

6. Änderungen an Systemd-Units richtig behandeln

Effektive Unit-Konfiguration anzeigen

```
[RO] systemctl cat <DIENST>
```

Abhängigkeiten untersuchen

```
[RO] systemctl list-dependencies <DIENST>
[RO] systemctl list-dependencies --reverse <DIENST>
```

Lokale Überschreibung bearbeiten

```
[CHANGE][PRIV] sudo systemctl edit <DIENST>
```

Unit-Dateien neu einlesen

```
[CHANGE][PRIV] sudo systemctl daemon-reload
```

`daemon-reload` liest Unit-Dateien und die systemd-Abhängigkeitsstruktur neu ein. Es startet den betroffenen Dienst nicht automatisch neu.

Vor einer Änderung müssen gesichert werden:

- Originalkonfiguration
- Inhalt bestehender Drop-ins
- Paketzugehörigkeit
- Abhängigkeiten
- Startbenutzer
- Umgebungsvariablen
- benötigte Dateien und Verzeichnisse
- Rückfallplan
- erwartete Auswirkungen eines Neustarts

> Paketverwaltete Unit-Dateien unter `/usr/lib/systemd/system` oder `/lib/systemd/system` sollten nicht direkt als dauerhafte Anpassungsmethode bearbeitet werden. Lokale Drop-ins verhindern, dass Paketaktualisierungen die Anpassung einfach überschreiben.

7. Journal mit Journalctl auswerten

```
[RO][SENS] journalctl
[RO][SENS] journalctl -b
[RO][SENS] journalctl -b -1
[RO][SENS] journalctl -p warning
[RO][SENS] journalctl -u <DIENST>
[RO][SENS] journalctl -u <DIENST> -b
[RO][SENS] journalctl --since "2026-08-01 14:00:00"
[RO][SENS] journalctl --since "30 minutes ago"
[RO][SENS] journalctl -k
[RO][SENS] journalctl -f
```

Zusätzliche Filter

```
[RO][SENS] journalctl _PID=<PID>
[RO][SENS] journalctl _UID=<UID>
```

```
[RO][SENS] journalctl _COMM=<PROGRAMM>
[RO][SENS] journalctl -o verbose
```

Speicherbelegung des Journals

```
[RO] journalctl --disk-usage
```

Zu beachten sind:

- richtiger Bootvorgang
- Zeitzone und Zeitstempel
- erste Fehlermeldung statt nur Folgefehler
- Dienstname und Prozess-ID
- Logrotation
- persistente oder flüchtige Speicherung
- Berechtigungen
- mögliche Geheimnisse oder personenbezogene Daten

> `journalctl -f` zeigt neu eintreffende Einträge. Es ersetzt nicht die Untersuchung älterer Ereignisse vor dem sichtbaren Fehler.

8. Klassische Protokolldateien untersuchen

Abhängig von der Distribution und Konfiguration können relevante Dateien beispielsweise unter `/var/log` liegen.

```
[RO][PRIV][SENS] sudo ls -lah /var/log
[RO][PRIV][SENS] sudo tail -n 200 <LOGDATEI>
[RO][PRIV][SENS] sudo tail -f <LOGDATEI>
[RO][PRIV][SENS] sudo less <LOGDATEI>
[RO][PRIV][SENS] sudo grep -iE "error|fail|warning" <LOGDATEI>
```

Mögliche Protokolle sind unter anderem:

- `/var/log/syslog`
- `/var/log/messages`
- `/var/log/auth.log`
- `/var/log/secure`

- `/var/log/kern.log`
- anwendungsspezifische Unterverzeichnisse

Diese Pfade sind nicht auf jeder Distribution vorhanden. Ein System kann Ereignisse ausschließlich oder zusätzlich im systemd-Journal speichern.

Rotierte Protokolle

```
[RO][PRIV][SENS] sudo zless <LOGDATEI>.gz
[RO][PRIV][SENS] sudo zgrep -i "<SUCHTEXT>" <LOGDATEI>.gz
```

> Protokolldateien dürfen nicht gelöscht oder geleert werden, bevor Ursache, Aufbewahrungspflichten und Auswirkungen auf laufende Prozesse geklärt sind.

9. Kernelmeldungen und Hardwareereignisse prüfen

```
[RO][PRIV][SENS] sudo dmesg
[RO][PRIV][SENS] sudo dmesg --ctime
[RO][PRIV][SENS] sudo dmesg --level=err,warn
[RO][PRIV][SENS] sudo journalctl -k -b
[RO][PRIV][SENS] sudo journalctl -k -b -1
```

Typische Suchbegriffe:

```
[RO][PRIV][SENS] sudo dmesg --ctime | grep -iE "error|fail|timeout|reset|oom|I/O"
```

Zu untersuchen sind beispielsweise:

- I/O-Fehler
- Geräte-Resets
- Dateisystemfehler
- Treiberprobleme
- Link-Änderungen
- Out-of-Memory-Ereignisse
- Kernel-Warnungen
- Machine-Check-Ereignisse

- USB-Verbindungsabbrüche
- blockierte Tasks

> Der Kernel-Ringpuffer ist begrenzt. Ältere Meldungen können überschrieben worden sein. Außerdem dürfen Zeitangaben aus `dmesg` nicht ungeprüft mit Zeitstempeln anderer Protokolle gleichgesetzt werden.

10. Arbeitsspeicher und OOM-Killer untersuchen

```
[RO] free -h
[RO] cat /proc/meminfo
[RO][SENS] ps -eo pid,user,%mem,rss,vsz,cmd --sort=-rss
[RO] vmstat 1 10
[RO][PRIV][SENS] sudo journalctl -k | grep -iE "out of memory|oom|killed process"
```

Wichtige Bereiche:

- verfügbarer Speicher
- Page Cache
- Swap-Nutzung
- Major Page Faults
- Prozess-RSS
- cgroup- oder Containergrenzen
- Speicherwachstum über die Zeit
- OOM-Auswahl des Kernels
- NUMA-Zuordnung
- reservierter Speicher einer VM

Linux verwendet freien Arbeitsspeicher sinnvoll als Cache. Ein geringer Wert in der Spalte `free` bedeutet daher nicht automatisch Speichermangel. Für die erste Bewertung ist bei `free` insbesondere `available` relevant.

Swap-Nutzung anzeigen

```
[RO] swapon --show
[RO] cat /proc/swaps
```

> Das Leeren von Caches oder Deaktivieren von Swap ist keine allgemeine Speicherreparatur und kann zusätzliche Last oder einen Systemausfall verursachen.

11. Datenträger, Partitionen und Einhängepunkte prüfen

```
[RO][SENS] lsblk -f
[RO][SENS] findmnt
[RO] df -hT
[RO] df -i
[RO][PRIV][SENS] sudo fdisk -l
[RO][PRIV][SENS] sudo blkid
```

Warum `df -h` und `df -i` benötigt werden

Ein Dateisystem kann trotz freier Datenblöcke keine neuen Dateien mehr anlegen, wenn keine freien Inodes verfügbar sind.

Große Verzeichnisse ermitteln

```
[RO][PRIV][SENS] sudo du -xhd1 <VERZEICHNIS> | sort -h
```

Gelöschte, aber noch geöffnete Dateien suchen

```
[RO][PRIV][SENS] sudo lsof +L1
```

Wenn ein Prozess eine gelöschte Datei noch geöffnet hält, kann der belegte Speicherplatz bis zum Schließen des Dateideskriptors weiterhin belegt bleiben.

> “ `du` und `df` messen unterschiedliche Sachverhalte. Abweichungen beweisen nicht automatisch einen Dateisystemfehler.

12. Datenträger-I/O und Gerätezustand analysieren

Falls die erforderlichen Werkzeuge installiert sind:

```
[RO] iostat -xz 1 10
[RO][PRIV] sudo smartctl -a /dev/<GERÄT>
[RO][PRIV] sudo nvme smart-log /dev/<NVME-GERÄT>
```

Zu prüfen sind:

- Latenz
- Warteschlangen
- Auslastung
- Durchsatz
- I/O-Fehler
- Geräte-Resets
- Medienfehler
- Temperatur
- verbleibende Reserve
- RAID- oder Storage-Layer
- virtuelle Datenträger
- parallele Sicherungen

SMART-Werte müssen geräte- und herstellerspezifisch interpretiert werden. Ein bestandener allgemeiner SMART-Status schließt einen Defekt nicht sicher aus.

> Selbsttests, Reparaturbefehle und Schreibtests können Last erzeugen oder ein vorgeschädigtes Medium zusätzlich beanspruchen. Vorher müssen Sicherungsstand und Auswirkungen geklärt werden.

13. Dateisystemfehler sicher behandeln

Dateisystemtyp und Einhängezustand prüfen

```
[RO][SENS] findmnt
[RO][SENS] lsblk -f
[RO] mount
[RO][PRIV][SENS] sudo journalctl -k | grep -iE "filesystem|ext4|xfs|btrfs|I/O error"
```

Eine Dateisystemprüfung muss zum tatsächlichen Dateisystem passen. Werkzeuge für ext-Dateisysteme, XFS und Btrfs sind nicht beliebig austauschbar.

```
[TEST][PRIV][DANGER] sudo fsck <BLOCKGERÄT>
```

> `fsck` darf nicht unüberlegt auf einem schreibend eingehängten Dateisystem ausgeführt werden. Vor einer Reparatur müssen Dateisystemtyp, Geräteziel, Einhängezustand, Sicherung und Wiederherstellungsplan eindeutig geklärt sein.

Bei Root-Dateisystemen kann eine Prüfung im Rettungsmodus, über ein Wartungssystem oder beim Start erforderlich sein.

14. Netzwerkadapter und Adressen untersuchen

```
[RO][SENS] ip -br link
[RO][SENS] ip -br address
[RO][SENS] ip address show
[RO][SENS] ip -s link
[RO][SENS] ip route show
[RO][SENS] ip rule show
[RO][SENS] ip neigh show
```

Zu prüfen sind:

- administrativer und physischer Link-Zustand
- MAC-Adresse
- IPv4- und IPv6-Adressen
- Präfixlängen
- Standardroute
- Routingtabellen
- Policy Routing
- Interfacefehler und Drops
- Nachbartabelle
- VLANs, Bonds und Bridges
- virtuelle Interfaces
- Network Namespace

>

> Die früher verbreiteten Werkzeuge `ifconfig`, `route` und `netstat` werden auf vielen modernen Linux-Systemen durch Werkzeuge aus `iproute2`, insbesondere `ip` und `ss`, ersetzt.

15. Erreichbarkeit und Pfad testen

```
[TEST][REMOTE] ping -c 4 <IP-ODER-HOSTNAME>
[TEST][REMOTE] ping -6 -c 4 <IPv6-ODER-HOSTNAME>
[TEST][REMOTE][SENS] tracepath <ZIEL>
[TEST][REMOTE][SENS] traceroute <ZIEL>
[TEST][REMOTE][SENS] mtr <ZIEL>
```

Ein erfolgreicher Ping beweist nur, dass die verwendete ICMP-Kommunikation in dieser Richtung und zu diesem Zeitpunkt funktioniert hat. Er beweist nicht, dass:

- ein bestimmter TCP- oder UDP-Port erreichbar ist
- DNS korrekt arbeitet
- die Anwendung antwortet
- TLS funktioniert
- Authentifizierung erfolgreich ist
- der Rückweg für alle Verbindungen identisch ist

Ein fehlgeschlagener Ping beweist umgekehrt keinen vollständigen Ausfall, da ICMP gefiltert oder begrenzt werden kann.

16. DNS-Auflösung untersuchen

```
[RO][SENS] cat /etc/resolv.conf
[RO][SENS] getent hosts <HOSTNAME>
[TEST][REMOTE][SENS] dig <HOSTNAME>
[TEST][REMOTE][SENS] dig <HOSTNAME> A
[TEST][REMOTE][SENS] dig <HOSTNAME> AAAA
[TEST][REMOTE][SENS] dig -x <IP-ADRESSE>
[TEST][REMOTE][SENS] dig @<DNS-SERVER> <HOSTNAME>
[TEST][REMOTE][SENS] resolvectl query <HOSTNAME>
[RO][SENS] resolvectl status
```

`getent hosts` berücksichtigt die konfigurierte Namensdienstauflösung des Systems und kann daher für die tatsächliche Anwendungssicht aussagekräftiger sein als eine isolierte DNS-Abfrage.

Zu prüfen sind:

- Inhalt von `/etc/nsswitch.conf`
- konfigurierte Resolver
- Suchdomänen
- Split DNS
- lokale Hosts-Datei
- Cache
- VPN-Konfiguration
- systemd-resolved
- NetworkManager
- A- und AAAA-Einträge
- Reverse Lookup
- DNSSEC-Fehler
- unterschiedliche Antworten verschiedener Server

> `/etc/resolv.conf` kann eine automatisch erzeugte Datei oder ein symbolischer Link sein. Manuelle Änderungen können überschrieben werden und sind nicht zwingend die richtige dauerhafte Konfigurationsmethode.

17. Ports, Sockets und Verbindungen prüfen

```
[RO][PRIV][SENS] sudo ss -lntup
[RO][SENS] ss -tan
[RO][SENS] ss -s
[RO][PRIV][SENS] sudo lsof -i
[TEST][REMOTE] nc -vz <HOST> <PORT>
[TEST][REMOTE][SENS] curl -v <URL>
```

Zu prüfen sind:

- lauscht der Dienst überhaupt?
- an welcher Adresse lauscht er?
- IPv4, IPv6 oder beides?
- stimmt der Port?

- welcher Prozess besitzt den Socket?
- ist nur `127.0.0.1` beziehungsweise `::1` gebunden?
- erreicht der Client den Port?
- kommt eine Anwendungsantwort zurück?
- scheitert TLS oder Authentifizierung?
- existieren viele halboffene oder wartende Verbindungen?

> Ein offener TCP-Port beweist nur, dass eine Verbindung angenommen werden kann. Die fachliche Funktion der Anwendung muss separat getestet werden.

18. Netzwerkverkehr mit Tcpdump erfassen

```
[TEST][PRIV][REMOTE][LOG][SENS] sudo tcpdump -i <INTERFACE>
[TEST][PRIV][REMOTE][LOG][SENS] sudo tcpdump -i <INTERFACE> host <IP-ADRESSE>
[TEST][PRIV][REMOTE][LOG][SENS] sudo tcpdump -i <INTERFACE> port <PORT>
[TEST][PRIV][REMOTE][LOG][SENS] sudo tcpdump -i <INTERFACE> -nn -s 0 -w <DATEI>.pcap
```

Vor einer Aufzeichnung müssen festgelegt werden:

- richtiges Interface
- geeigneter Filter
- Aufzeichnungsdauer
- Speicherort
- maximaler Speicherbedarf
- Zeitpunkt des Fehlers
- Datenschutz
- sichere Übertragung
- Löschfrist

Eine Paketaufzeichnung kann enthalten:

- IP-Adressen
- DNS-Anfragen
- Hostnamen
- unverschlüsselte Anwendungsdaten
- Sitzungsinformationen
- interne Netzwerkstrukturen
- Kommunikationsbeziehungen

> Ohne Begrenzung kann eine Aufzeichnung sehr groß werden. Bei produktiven Systemen sollten rotierende Dateien, zeitliche Begrenzung und möglichst enge Capture-Filter verwendet werden.

19. Firewall und Paketfilter prüfen

Je nach System können unterschiedliche Frameworks oder Verwaltungswerkzeuge eingesetzt werden.

```
[RO][PRIV][SENS] sudo nft list ruleset
[RO][PRIV][SENS] sudo iptables -S
[RO][PRIV][SENS] sudo iptables -t nat -S
[RO][PRIV][SENS] sudo firewall-cmd --list-all
[RO][PRIV][SENS] sudo ufw status verbose
```

Zu prüfen sind:

- tatsächlich aktives Firewall-Backend
- Eingangs-, Ausgangs- und Weiterleitungsregeln
- Standardrichtlinien
- Regelreihenfolge
- Interfaces und Zonen
- Quell- und Zielnetze
- Ports und Protokolle
- NAT
- Connection Tracking
- IPv4 und IPv6
- Container- oder Virtualisierungsregeln
- persistente gegenüber aktuell geladenen Regeln

> Mehrere Werkzeuge können auf dasselbe oder auf unterschiedliche Backends zugreifen. Eine leere `iptables`-Ausgabe beweist daher nicht automatisch, dass keine Filterregeln aktiv sind.

20. NetworkManager untersuchen

```
[RO][SENS] nmcli general status
[RO][SENS] nmcli device status
[RO][SENS] nmcli connection show
[RO][SENS] nmcli device show <INTERFACE>
[RO][SENS] nmcli connection show "<VERBINDUNG>"
[RO][PRIV][SENS] journalctl -u NetworkManager -b
```

Zu prüfen sind:

- verwaltete und nicht verwaltete Interfaces
- aktive Verbindung
- Verbindungsprofil
- DHCP-Daten
- DNS-Konfiguration
- Route und Metrik
- Autoconnect
- VLAN, Bond oder Bridge
- Zeitpunkt eines Linkwechsels
- Konflikte mit anderen Netzwerkverwaltungen

> Änderungen mit `nmcli` können eine SSH-Verbindung sofort unterbrechen. Vor Remoteänderungen sind Konsolenzugang und Rückfallplan erforderlich.

21. Benutzer, Gruppen und Berechtigungen prüfen

```
[RO][SENS] id <BENUTZER>
[RO][SENS] groups <BENUTZER>
[RO][SENS] getent passwd <BENUTZER>
[RO][SENS] getent group <GRUPPE>
[RO][SENS] namei -l <PFAD>
[RO][SENS] stat <PFAD>
[RO][SENS] ls -ld <PFAD>
[RO][SENS] getfacl <PFAD>
```

Bei einem Pfadproblem müssen alle übergeordneten Verzeichnisse berücksichtigt werden. Für das Durchlaufen eines Verzeichnisses ist das Ausführungsrecht am Verzeichnis entscheidend.

Zu unterscheiden sind:

- Eigentümer
- Gruppe
- klassische Modusbits
- ACLs
- umask
- effektive Benutzer- und Gruppen-ID
- ergänzende Gruppen
- Mount-Optionen
- SELinux oder AppArmor
- Container- und Namespace-Zuordnung
- NFS- oder SMB-Berechtigungen

> `chmod 777` ist keine fachgerechte Standardlösung. Es kann Daten ungewollt für alle lokalen Benutzer veränderbar machen und verdeckt die eigentliche Ursache.

22. SELinux und AppArmor berücksichtigen

SELinux

```
[RO] getenforce
[RO] sestatus
[RO][PRIV][SENS] sudo ausearch -m AVC,USER_AVC -ts recent
[RO][PRIV][SENS] sudo journalctl | grep -i "avc:"
```

AppArmor

```
[RO] aa-status
[RO][PRIV][SENS] sudo journalctl -k | grep -i apparmor
```

Ein Unix-Dateirecht kann korrekt erscheinen, während eine Mandatory-Access-Control-Richtlinie den Zugriff trotzdem verhindert.

> SELinux dauerhaft zu deaktivieren oder ein AppArmor-Profil ungeprüft abzuschalten ist keine geeignete Ursachenanalyse. Zuerst müssen die konkrete Ablehnung, der betroffene Kontext und die erwartete Richtlinie festgestellt werden.

23. Geöffnete Dateien und Systemaufrufe untersuchen

```
[RO][PRIV][SENS] sudo lsof -p <PID>
[RO][PRIV][SENS] sudo lsof <PFAD>
[RO][PRIV][SENS] sudo lsof -i :<PORT>
[TEST][PRIV][LOG][SENS] sudo strace -p <PID>
[TEST][PRIV][LOG][SENS] sudo strace -f -o <DATEI> <BEFEHL>
```

`strace` kann sichtbar machen, ob ein Prozess beispielsweise an folgenden Vorgängen scheitert:

- Datei öffnen
- Namensauflösung
- Socket-Verbindung
- Berechtigungsprüfung
- Lesen oder Schreiben
- Warten auf ein Ereignis
- Start eines Kindprozesses

Die Ausgabe kann sehr umfangreich sein und sensible Daten enthalten. Außerdem kann Tracing das Zeitverhalten eines Prozesses beeinflussen.

> Ein produktiver Prozess sollte nur mit geklärter Auswirkung verfolgt werden. Geheimnisse, Pfade und Nutzdaten können in der Ausgabe erscheinen.

24. Paketbestand und beschädigte Installationen prüfen

Debian- und Ubuntu-basierte Systeme

```
[RO][SENS] dpkg -l
[RO] dpkg -S <PFAD>
```

```
[RO] apt-cache policy <PAKET>
[RO][PRIV] sudo dpkg --verify
```

RPM-basierte Systeme

```
[RO][SENS] rpm -qa
[RO] rpm -qf <PFAD>
[RO] rpm -V <PAKET>
[RO] dnf info <PAKET>
```

Zu prüfen sind:

- installierte Version
- Paketquelle
- Abhängigkeiten
- geänderte Paketdateien
- unvollständige Transaktion
- zurückgehaltene Pakete
- gemischte Repositorys
- Architektur
- Zeitpunkt des Updates
- erforderlicher Dienst- oder Systemneustart

> Eine Paketverifikation zeigt Abweichungen, bewertet aber nicht automatisch, ob diese absichtlich, harmlos oder fehlerursächlich sind.

25. Bootvorgang untersuchen

```
[RO] systemd-analyze
[RO] systemd-analyze blame
[RO] systemd-analyze critical-chain
[RO][SENS] journalctl -b
[RO][SENS] journalctl -b -1
[RO] systemctl --failed
[RO][PRIV][SENS] sudo dmesg --ctime
```

Bei Startproblemen sind zu prüfen:

- Bootloader
- Kernelparameter
- initramfs
- Root-Dateisystem
- `/etc/fstab`
- Geräte-UUIDs
- Mount-Zeitlimits
- systemd-Abhängigkeiten
- Netzwerkabhängigkeiten
- verschlüsselte Datenträger
- ausgefallene Dienste
- vorheriger Bootvorgang

`systemd-analyze blame` zeigt die Initialisierungszeit einzelner Units, beweist aber nicht allein, dass die oben angezeigte Unit den gesamten Start entsprechend verzögert hat. Parallelisierung und Abhängigkeiten müssen berücksichtigt werden.

26. Zeit und Zeitsynchronisation prüfen

```
[RO] date
[RO] timedatectl
[RO] timedatectl timesync-status
[RO][SENS] chronyc tracking
[RO][SENS] chronyc sources -v
```

Die verfügbaren Befehle hängen vom eingesetzten Zeitdienst ab.

Zu prüfen sind:

- Systemzeit
- Zeitzone
- Hardwareuhr
- aktive Synchronisation
- verwendete Zeitquelle
- Offset
- Erreichbarkeit der Zeitserver
- Virtualisierungszeitquelle
- Zeitänderungen in den Protokollen
- Zertifikatsgültigkeit
- Kerberos- oder Verzeichnisdienstabhängigkeiten

> Eine falsche Systemzeit kann Protokollkorrelation, TLS, Authentifizierung, Datenbanken und geplante Aufgaben beeinträchtigen.

27. Geplante Aufgaben untersuchen

```
[RO][SENS] crontab -l
[RO][PRIV][SENS] sudo crontab -l
[RO][PRIV][SENS] sudo ls -la /etc/cron.*
[RO] systemctl list-timers --all
[RO] systemctl status <TIMER>.timer
[RO][SENS] journalctl -u <TIMER>.timer
[RO][SENS] journalctl -u <DIENST>.service
```

Typische Fehlerursachen:

- anderer Benutzerkontext
- eingeschränkte Umgebungsvariablen
- anderer `PATH`
- relative Pfade
- fehlende Ausführungsrechte
- nicht vorhandenes Arbeitsverzeichnis
- gesperrte Dateien
- Zeitzone
- überlappende Ausführungen
- fehlgeschlagene Weiterleitung der Ausgabe
- Timer oder Dienst deaktiviert

> Ein Skript, das in einer interaktiven Shell funktioniert, muss unter Cron oder einem systemd-Timer nicht mit derselben Umgebung ausgeführt werden.

28. Container und Virtualisierung berücksichtigen

```
[RO] systemd-detect-virt
[RO][SENS] cat /proc/1/cgroup
[RO][SENS] lsns
```

```
[RO][SENS] docker ps
[RO][SENS] podman ps
[RO][SENS] docker inspect <CONTAINER>
[RO][SENS] docker logs <CONTAINER>
```

Bei Containerproblemen müssen Host und Container getrennt untersucht werden:

- Prozess im Container
- Containerzustand
- Restart Policy
- Exit-Code
- Healthcheck
- Logs
- Portveröffentlichung
- Container-Netzwerk
- DNS
- Volumes und Bind Mounts
- Benutzer-ID
- cgroup-Grenzen
- Host-Firewall
- Host-Speicherplatz
- Host-Kernel

> Ein laufender Container beweist nicht, dass der darin betriebene Dienst erreichbar oder funktionsfähig ist. Ein Container verwendet grundsätzlich den Kernel des Hosts und ist keine vollständige virtuelle Maschine.

29. Bash-Skripte kontrolliert debuggen

Syntax prüfen

```
[RO] bash -n <SKRIPT>
```

Ablauf mit Trace ausführen

```
[TEST][LOG][SENS] bash -x <SKRIPT>
```

In einem Skript kann die Trace-Ausgabe abschnittsweise aktiviert werden:

```
set -x
<ZU_UNTERSUCHENDE_BEFEHLE>
set +x
```

Zu prüfen sind:

- Shebang
- Shell-Kompatibilität
- Variablen
- Quoting
- Leerzeichen in Pfaden
- Rückgabecodes
- Pipelines
- relative Pfade
- Benutzerkontext
- Umgebungsvariablen
- Dateirechte
- parallele Ausführung

> `set -x` kann Kennwörter, Tokens, Argumente und andere Geheimnisse in Protokolle schreiben. Es darf nicht unkontrolliert in produktiven Skripten aktiviert werden.

30. Typische Fehlerbilder und geeignete Werkzeuge

Fehlerbild	Geeignete Prüfungen
Dienst startet nicht	`systemctl status`, `journalctl -u`, Unit-Konfiguration, Abhängigkeiten, Rechte
Server reagiert langsam	`uptime`, `top`, `vmstat`, `iostat`, `free`, Prozess- und I/O-Analyse
Datenträger voll	`df -hT`, `df -i`, `du`, `lsof +L1`, Journalgröße
Prozess wird beendet	Kerneljournal, OOM-Meldungen, cgroup-Grenzen, Dienstkonfiguration
Port nicht erreichbar	`ss`, Dienst-Bindung, Firewall, Route, `nc`, `tcpdump`
Hostname wird falsch aufgelöst	`getent`, `dig`, `resolvectl`, `nsswitch.conf`, Hosts-Datei
Netzwerk fällt sporadisch aus	Linkstatistik, NetworkManager-Journal, Route, `mtr`, `tcpdump`

Fehlerbild	Geeignete Prüfungen
Datei kann nicht geöffnet werden	`namei`, `stat`, `getfacl`, SELinux/AppArmor, `lsof`, `strace`
System bootet langsam	`systemd-analyze`, vorheriges Journal, Mounts, Abhängigkeiten
Root-Dateisystem nur lesbar	Kernelmeldungen, I/O-Fehler, Dateisystemzustand, Storage
Cronjob läuft nicht	Benutzerkontext, Umgebung, absolute Pfade, Cronlogs, Exit-Code
Container startet ständig neu	Containerstatus, Exit-Code, Logs, Healthcheck, Volumes, Ressourcen
Anwendung funktioniert lokal, aber nicht remote	Bind-Adresse, Firewall, Routing, NAT, DNS, TLS
Zeitstempel sind unplausibel	`timedatectl`, Zeitdienst, Zeitzone, VM-Zeitquelle
Änderungen verschwinden nach Neustart	persistente Konfiguration, generierte Dateien, NetworkManager, systemd-Drop-ins

31. Empfohlener Gesamtablauf bei Linux-Problemen

1. vollständige Fehlermeldung erfassen.
2. Benutzer, Host, Dienst und Zeitpunkt dokumentieren.
3. Umfang und Reproduzierbarkeit bestimmen.
4. letzte Änderung feststellen.
5. Distribution, Version, Kernel und Laufzeitumgebung erfassen.
6. Systemzeit und Uptime prüfen.
7. CPU, Load, Arbeitsspeicher und I/O überblicken.
8. Speicherplatz und Inodes prüfen.
9. betroffenen Prozess oder Dienst bestimmen.
10. Dienststatus und Abhängigkeiten prüfen.
11. Journal im passenden Zeitfenster auswerten.
12. Kernelmeldungen kontrollieren.
13. Konfiguration und Berechtigungen prüfen.
14. SELinux oder AppArmor berücksichtigen.
15. bei Netzwerkproblemen Adapter, Adressen und Routen prüfen.
16. DNS-Auflösung über den tatsächlich verwendeten Resolver testen.
17. Ports und Bind-Adressen prüfen.
18. Firewall und Paketpfad untersuchen.
19. bei Bedarf eng gefilterte Paketaufzeichnung erstellen.
20. Hypothese aus mehreren Befunden formulieren.
21. kleinste reversible Maßnahme planen.
22. Sicherung und Rückfallplan prüfen.
23. Änderung kontrolliert durchführen.
24. denselben Fehlerfall erneut testen.

25. Dienste, Logs und Monitoring nachkontrollieren.
26. temporäres Debugging deaktivieren.
27. Ursache, Maßnahme und Ergebnis dokumentieren.

32. Gefährliche Fehlinterpretationen vermeiden

Beobachtung	Nicht automatisch bewiesen
Prozess erscheint in `ps`	Anwendung funktioniert
Dienst ist `active`	Dienst ist fachlich erreichbar
Port steht auf `LISTEN`	Remotezugriff funktioniert
Ping funktioniert	TCP, UDP, DNS und Anwendung funktionieren
DNS liefert eine Adresse	Adresse ist aktuell und der Dienst erreichbar
`free` zeigt wenig freien RAM	akuter Speichermangel
Load Average ist hoch	ausschließlich CPU-Überlastung
`df` zeigt freien Platz	genügend freie Inodes vorhanden
`du` ist kleiner als `df`	Dateisystem ist beschädigt
SMART meldet `PASSED`	Datenträger ist fehlerfrei
Firewallregel existiert	sie trifft in der gewünschten Richtung zu
Dateirechte sind korrekt	SELinux, AppArmor oder ACLs erlauben den Zugriff
Neustart behebt das Symptom	Ursache ist behoben
Paketdatei wurde verändert	Veränderung verursacht den Fehler
Container läuft	Anwendung im Container ist gesund
keine Fehlermeldung sichtbar	Vorgang war erfolgreich

33. Sicherheits- und Datenschutzanforderungen

Linux-Diagnosedaten können enthalten:

- Benutzernamen und Gruppen
- Hostnamen
- interne IP-Adressen
- MAC-Adressen
- DNS-Namen
- Prozessargumente
- Dateipfade

- Umgebungsvariablen
- Netzwerkverbindungen
- Paketmitschnitte
- SSH-Zugriffe
- Authentifizierungsereignisse
- Containerkonfigurationen
- Volume-Pfade
- Tokens und Zugangsdaten
- interne Anwendungsdaten

Vor einer Weitergabe müssen:

1. Zweck und Empfänger bestimmt werden.
2. nur erforderliche Daten erfasst werden.
3. Kennwörter, Schlüssel, Cookies und Tokens entfernt werden.
4. personenbezogene Angaben redigiert werden.
5. Paketmitschnitte besonders geschützt werden.
6. Dateien verschlüsselt übertragen werden.
7. Zugriffsrechte und Aufbewahrungsdauer festgelegt werden.
8. temporäres Debugging deaktiviert werden.
9. nicht mehr benötigte Diagnosedaten sicher gelöscht werden.
10. externe Analysedienste organisatorisch freigegeben sein.

> Shell-History, Prozesslisten und Trace-Ausgaben können Geheimnisse enthalten, wenn diese als Befehlsargumente übergeben wurden.

34. Dokumentationsvorlage

```
Ticket:
Datum und Uhrzeit:
Bearbeiter:

Hostname:
Distribution:
Version:
Kernel:
Architektur:
Physisch, VM oder Container:
```

Zeitzone:

Uptime:

Betroffener Benutzer:

Betroffener Dienst:

Betroffener Prozess:

PID:

Originale Fehlermeldung:

Fehlerzeitpunkt:

Reproduzierbar:

Betroffene Systeme:

Letzte funktionierende Nutzung:

Letzte Änderung:

CPU-Auslastung:

Load Average:

Arbeitsspeicher:

Swap:

OOM-Ereignis:

Datenträgerauslastung:

Freier Speicher:

Freie Inodes:

I/O-Wartezeit:

Dateisystem:

Mount-Optionen:

Kernelmeldungen:

Dienststatus:

Unit-Datei:

Drop-ins:

Abhängigkeiten:

Startbenutzer:

Exit-Code:

Journal-Zeitraum:

Erste relevante Fehlermeldung:

Interface:

IP-Adresse:

Präfix:

Gateway:

Route:

DNS-Server:

DNS-Ergebnis:

Lauschadresse:

Port:

Firewall:

Paketaufzeichnung:

Dateipfad:

Eigentümer:

Gruppe:

Modus:

ACL:

SELinux:

AppArmor:

Geöffnete Dateien:

Strace verwendet:

Paket:

Installierte Version:

Paketquelle:

Verifikation:

Letztes Update:

Vermutete Ursache:

Belege:

Geplante Maßnahme:

Sicherung:

Rückfallplan:

Durchgeführte Änderung:

Testergebnis:

Monitoring:

Abschluss:

Merksatz

> Bei der Linux-Fehleranalyse wird zuerst der Zustand gesichert und anschließend die Abhängigkeitskette geprüft: Hardware und Kernel stellen die Grundlage bereit, Dateisystem und Netzwerk ermöglichen den Zugriff, Benutzerrechte und Sicherheitsrichtlinien kontrollieren die Berechtigung, Dienste stellen die Funktion bereit und Protokolle verbinden Fehler, Zeitpunkt und Ursache.

Quellen und weiterführende Dokumentation

- freedesktop.org – systemctl
- freedesktop.org – journalctl
- freedesktop.org – systemd-journald
- freedesktop.org – systemd.service
- freedesktop.org – systemd.unit
- freedesktop.org – systemd-analyze
- Linux man-pages – Übersicht
- Linux man-pages – proc
- Linux man-pages – ps
- Linux man-pages – kill
- Linux man-pages – ip
- Linux man-pages – ss
- Linux man-pages – mount
- Linux man-pages – namespaces
- Linux Kernel Documentation – Administrator’s Guide
- Linux Kernel Documentation – Kernel Parameters
- Linux Kernel Documentation – Kdump
- Linux Kernel Documentation – Magic SysRq
- Red Hat – Network Troubleshooting and Performance Tuning
- Red Hat – NetworkManager Debugging