8.4 Linux-Server – Dienste, Prozesse, Protokolle, Ressourcen und Speicher

---

1. Ziel dieser Seite

Diese Seite beschreibt die strukturierte Fehleranalyse auf Linux-Servern.

Typische Störungen betreffen:

- systemd-Dienste;
- Prozesse;
- Konfigurationsdateien;
- Benutzer und Berechtigungen;
- SELinux oder AppArmor;
- Netzwerk und Ports;
- DNS und Namensauflösung;
- CPU, RAM und Swap;
- Datenträger und Dateisysteme;
- Inodes;
- Mounts;
- LVM und Software-RAID;
- Paketverwaltung;
- Cronjobs und systemd-Timer;
- Kernel, Treiber und Hardware;
- Bootvorgang;
- Zeit- und Zertifikatsprobleme.

Das Ziel ist nicht, einen Dienst möglichst schnell neu zu starten. Zuerst werden Fehlerzustand, Ursache, Auswirkungen und Rückweg dokumentiert.

---

2. Diagnosegrundsatz

```
Symptom erfassen
      ↓
Umfang bestimmen
      ↓
Systemzustand und Zeitpunkt prüfen
      ↓
Dienst, Prozess und Port untersuchen
      ↓
```

```
Protokolle auswerten
        ↓
Abhängigkeiten prüfen
        ↓
Konfiguration und Berechtigungen prüfen
        ↓
Ressourcen und Storage prüfen
        ↓
Ursache nachweisen
        ↓
Eine kontrollierte Maßnahme durchführen
        ↓
Funktion und Nebenwirkungen kontrollieren
```

Ein erfolgreicher Prozessstart beweist noch keine funktionierende Anwendung. Ebenso beweist ein offener Port nicht, dass die Anwendung korrekte Antworten liefert.

---

3. Fehlerbild exakt erfassen

Zu dokumentieren sind:

- Hostname;
- Distribution;
- Betriebssystemversion;
- Kernelversion;
- Architektur;
- betroffener Dienst;
- betroffene Anwendung;
- vollständiger Fehlertext;
- Fehlercode;
- Zeitpunkt mit Zeitzone;
- betroffener Benutzer;
- Quell- und Zielsystem;
- Port und Protokoll;
- betroffene Funktion;
- Umfang der Störung;
- letzte funktionierende Nutzung;
- letzte Änderungen;
- installierte Updates;
- Neustarts;
- Ressourcenstatus;
- bisherige Diagnose- und Änderungsversuche.

Wichtige Abgrenzungsfragen:

- Ist nur ein Benutzer betroffen?
- Ist nur ein Client betroffen?
- Ist nur eine Anwendung betroffen?
- Ist nur eine Instanz betroffen?
- Betrifft es einen oder mehrere Server?
- Funktioniert der Dienst lokal?
- Funktioniert der Dienst über das Netzwerk?
- Ist nur IPv4 oder IPv6 betroffen?
- Tritt der Fehler dauerhaft oder sporadisch auf?
- Begann der Fehler nach einer Änderung?
- Funktioniert die Anwendung, obwohl der Dienststatus einen Fehler meldet?

---

4. System identifizieren

Hostname:

```
hostnamectl
```

Kurzer Hostname:

```
hostname
```

Betriebssysteminformationen:

```
cat /etc/os-release
```

Kernelversion und Architektur:

```
uname -a
```

Nur Kernelversion:

```
uname -r
```

Architektur:

```
uname -m
```

Systemlaufzeit und Last:

```
uptime
```

Letzter Systemstart:

```
who -b
```

Aktuelle Zeit und Zeitzone:

```
timedatectl
```

Diese Informationen sind wichtig, weil Befehle, Dateipfade, Paketmanager, Sicherheitsmodule und Protokollierung je nach Distribution und Version unterschiedlich sein können.

---

5. Letzte Änderungen bestimmen

Mögliche Ursachen kurz vor Beginn einer Störung:

- Paketupdate;
- Kernelupdate;
- Konfigurationsänderung;
- Zertifikatsaustausch;
- Änderung von Benutzer oder Gruppe;
- Änderung von Dateirechten;
- Firewalländerung;
- DNS-Änderung;
- neue Mount-Konfiguration;
- Storage-Erweiterung;
- Neustart;
- Deployment;
- geänderte Umgebungsvariable;
- neuer systemd-Drop-in;
- geänderte SELinux- oder AppArmor-Regel;
- abgelaufenes Kennwort oder Zertifikat.

Anmeldehistorie:

```
last
```

Fehlgeschlagene Anmeldungen, sofern unterstützt und berechtigt:

```
lastb
```

Letzte Neustarts:

```
last reboot
```

Zeitstempel einer Datei:

```
stat /etc/example/application.conf
```

RPM-basierte Systeme:

```
dnf history
```

Debian- und Ubuntu-Systeme:

```
grep -E ' install | upgrade | remove ' /var/log/dpkg.log
```

Rotierte Protokolle müssen gegebenenfalls zusätzlich geprüft werden.

---

6. systemd-Gesamtzustand prüfen

Fehlgeschlagene Units:

```
systemctl --failed
```

Systemzustand:

```
systemctl is-system-running
```

Alle geladenen Dienste:

```
systemctl list-units --type=service --all
```

Installierte Unit-Dateien:

```
systemctl list-unit-files --type=service
```

Abweichende oder lokal angepasste Units:

```
systemd-delta
```

Ein System kann als `degraded` angezeigt werden, obwohl die vom Benutzer benötigte Anwendung funktioniert. Deshalb müssen die fehlgeschlagenen Units einzeln bewertet werden.

---

7. Dienststatus untersuchen

Beispieldienst:

```
systemctl status example.service
```

Aktivitätsstatus:

```
systemctl is-active example.service
```

Autostartstatus:

```
systemctl is-enabled example.service
```

Fehlerzustand:

```
systemctl is-failed example.service
```

Ausgewählte Eigenschaften:

```
systemctl show example.service \
    -p ActiveState \
    -p SubState \
    -p Result \
    -p ExecMainCode \
    -p ExecMainStatus \
    -p MainPID \
    -p User \
```

```
  -p Group \
  -p FragmentPath
```

Zu unterscheiden sind:

- `loaded`: Unit-Datei wurde geladen;
- `active`: Unit ist aktiv;
- `inactive`: Unit ist nicht aktiv;
- `failed`: vorheriger Start oder Betrieb endete fehlerhaft;
- `enabled`: Unit ist für automatischen Start eingebunden;
- `disabled`: kein automatischer Start über die übliche Aktivierung;
- `masked`: Start wurde durch eine Verknüpfung auf `/dev/null` blockiert;
- `static`: Unit besitzt normalerweise keinen eigenen Installationsabschnitt und wird durch andere Units gestartet;
- `activating`: Startvorgang läuft;
- `deactivating`: Beendigung läuft.

`enabled` bedeutet nicht, dass der Dienst gerade läuft. `active` bedeutet nicht automatisch, dass er beim nächsten Systemstart wieder gestartet wird.

---

8. Dienstprotokoll mit journalctl prüfen

Aktuelles Dienstprotokoll:

```
journalctl -u example.service
```

Nur aktueller Systemstart:

```
journalctl -u example.service -b
```

Vorheriger Systemstart:

```
journalctl -u example.service -b -1
```

Letzte 100 Einträge:

```
journalctl -u example.service -n 100
```

Fortlaufende Anzeige:

```
journalctl -u example.service -f
```

Bestimmter Zeitraum:

```
journalctl -u example.service \
    --since "2026-08-02 10:00:00" \
    --until "2026-08-02 11:00:00"
```

Warnungen und schwerere Meldungen des aktuellen Starts:

```
journalctl -b -p warning
```

Kernelmeldungen des aktuellen Starts:

```
journalctl -k -b
```

Ausführliche Ausgabe:

```
journalctl -u example.service -o verbose
```

Zu dokumentieren sind:

- genauer Zeitpunkt;
- Unit;
- Prozess-ID;
- Benutzer;
- Fehlercode;
- Signal;
- erste relevante Fehlermeldung;
- nachfolgende Kaskadenfehler;
- wiederholte Startversuche;
- betroffene Datei;
- betroffene Adresse oder Portnummer.

Die letzte Fehlermeldung muss nicht die eigentliche Ursache sein. Häufig steht die erste relevante Meldung weiter oben.

---

9. Klassische Protokolldateien

Je nach Distribution und Dienst können zusätzlich relevant sein:

```
/var/log/syslog
/var/log/messages
/var/log/auth.log
/var/log/secure
/var/log/kern.log
/var/log/audit/audit.log
/var/log/dmesg
/var/log/cron
/var/log/maillog
```

Anwendungsprotokolle befinden sich häufig unter:

```
/var/log/<Anwendung>/
```

Letzte Zeilen anzeigen:

```
tail -n 100 /var/log/syslog
```

Fortlaufend beobachten:

```
tail -f /var/log/example/application.log
```

Komprimierte rotierte Protokolle durchsuchen:

```
zgrep -i 'error' /var/log/example/application.log*.gz
```

Nicht jede Distribution verwendet alle genannten Dateien. Auf ausschließlich journalbasierten Systemen können einzelne klassische Protokolldateien fehlen.

---

10. Unit-Datei und Überschreibungen prüfen

Wirksame Unit einschließlich Drop-ins anzeigen:

```
systemctl cat example.service
```

Pfad und Drop-ins:

```
systemctl show example.service \
    -p FragmentPath \
    -p DropInPaths
```

Abhängigkeiten:

```
systemctl list-dependencies example.service
```

Umgekehrte Abhängigkeiten:

```
systemctl list-dependencies --reverse example.service
```

Startreihenfolge und Abhängigkeiten können beeinflusst werden durch:

- `Requires=`;
- `Wants=`;
- `After=`;
- `Before=`;
- `BindsTo=`;
- `Condition...=`;
- `Assert...=`;
- Socket-Aktivierung;
- Mount-Abhängigkeiten;
- Netzwerk-Targets;
- Drop-in-Dateien;
- Environment-Dateien.

`After=` legt eine Reihenfolge fest, erzeugt allein aber keine Startabhängigkeit.

---

11. systemd-Konfiguration verifizieren

Unit-Dateien prüfen:

```
systemd-analyze verify example.service
```

Nach einer Änderung an Unit-Dateien:

```
systemctl daemon-reload
```

`daemon-reload` lädt die systemd-Konfiguration neu. Es startet den betroffenen Dienst nicht automatisch neu.

Vor einem Neustart des Dienstes sind zu prüfen:

- wurde nur die Unit-Datei oder auch die Anwendungskonfiguration geändert;
- besitzt die Anwendung einen eigenen Syntaxprüfer;
- verarbeitet der Dienst produktive Verbindungen;
- gehen Sitzungen oder Daten verloren;
- existiert ein Wartungsfenster;
- ist ein Rückweg vorbereitet.

---

12. Anwendungsidentität prüfen

Der Dienst kann unter einem anderen Benutzer laufen als erwartet.

systemd-Konfiguration:

```
systemctl show example.service \
    -p User \
    -p Group \
    -p SupplementaryGroups \
    -p DynamicUser
```

Prozessbenutzer:

```
ps -eo user,group,pid,ppid,stat,lstart,cmd
```

Bestimmter Prozess:

```
ps -o user,group,pid,ppid,stat,lstart,cmd -p <PID>
```

Benutzerinformationen:

```
id exampleuser
```

Gruppen eines Benutzers:

```
id -nG exampleuser
```

Kontoeintrag:

```
getent passwd exampleuser
```

Gruppeneintrag:

```
getent group examplegroup
```

`getent` berücksichtigt die konfigurierte Namensdienstauflösung und ist deshalb bei LDAP-, Active-Directory- oder SSSD-Umgebungen häufig aussagekräftiger als das alleinige Lesen von `/etc/passwd`.

---

13. Prozesse untersuchen

Alle Prozesse:

```
ps aux
```

Hierarchische Darstellung:

```
ps -ef --forest
```

Prozesse eines Benutzers:

```
ps -u exampleuser -f
```

Prozess anhand des Namens suchen:

```
pgrep -a example
```

Prozessbaum:

```
pstree -ap
```

Dynamische Anzeige:

```
top
```

Falls installiert:

```
htop
```

Zu prüfen sind:

- Prozess vorhanden;
- richtige ausführbare Datei;
- richtiger Benutzer;
- Elternprozess;
- Startzeit;
- CPU-Verbrauch;
- Speicherverbrauch;
- Prozesszustand;
- Anzahl der Threads;
- wiederholte Neustarts;
- Zombie-Prozesse;
- nicht unterbrechbarer Schlafzustand;
- unerwartete Kommandozeilenparameter.

---

14. Prozesszustände bewerten

Wichtige Zustände in `ps`:

Zustand	Bedeutung
`R`	läuft oder ist ausführbar
`S`	unterbrechbarer Schlafzustand
`D`	nicht unterbrechbarer Schlafzustand, häufig I/O-bezogen
`T`	angehalten oder verfolgt
`Z`	Zombieprozess
`I`	inaktiver Kernelthread

Ein Prozess im Zustand `D` kann häufig nicht sofort beendet werden, weil er auf eine Kernel- oder I/O-Operation wartet.

Ein Zombie verbraucht normalerweise kaum Ressourcen. Er zeigt jedoch, dass der Elternprozess den Beendigungsstatus des Kindprozesses noch nicht abgeholt hat.

---

15. Signale und Prozessbeendigung

Normale Beendigungsanforderung:

```
kill -TERM <PID>
```

Standardsignal von `kill`:

```
kill <PID>
```

Erzwungene Beendigung:

```
kill -KILL <PID>
```

Signale anzeigen:

```
kill -l
```

`SIGTERM` erlaubt einem Prozess eine kontrollierte Beendigung. `SIGKILL` kann nicht abgefangen werden und verhindert eine geordnete Bereinigung.

Vor einer Prozessbeendigung prüfen:

- besitzt der Prozess ungespeicherte Daten;
- hält er Datenbanktransaktionen;
- verwaltet systemd den Prozess;
- wird er automatisch neu gestartet;
- existieren Kindprozesse;
- ist der Prozess im Zustand `D`;
- welcher Benutzer ist betroffen;
- entstehen Ausfälle oder Inkonsistenzen.

Ein von systemd verwalteter Prozess sollte normalerweise über die zuständige Unit und nicht ausschließlich über seine PID gesteuert werden.

---

16. Offene Dateien und Arbeitsverzeichnisse prüfen

Offene Dateien eines Prozesses:

```
lsof -p <PID>
```

Prozesse mit Zugriff auf eine Datei:

```
lsof /var/lib/example/data.db
```

Alternativ:

```
fuser -v /var/lib/example/data.db
```

Prozesse auf einem Mountpoint:

```
fuser -vm /mnt/data
```

Gelöschte, aber weiterhin geöffnete Dateien:

```
lsof +L1
```

Eine gelöschte Datei kann weiterhin Speicherplatz belegen, solange ein Prozess sie geöffnet hält. Der Speicher wird normalerweise erst nach dem Schließen des Deskriptors freigegeben.

---

17. Ausführbare Datei und Bibliotheken prüfen

Verwendete Binärdatei:

```
readlink -f /proc/<PID>/exe
```

Arbeitsverzeichnis:

```
readlink -f /proc/<PID>/cwd
```

Kommandozeile:

```
tr '\0' ' ' < /proc/<PID>/cmdline
```

Umgebungsvariablen, nur mit erforderlicher Berechtigung:

```
tr '\0' '\n' < /proc/<PID>/environ
```

Bibliotheksabhängigkeiten:

```
ldd /usr/local/bin/example
```

`ldd` sollte nicht ungeprüft auf nicht vertrauenswürdige ausführbare Dateien angewendet werden. Je nach Datei und Implementierung kann dabei ein Sicherheitsrisiko entstehen.

---

18. Anwendungskonfiguration prüfen

Zu prüfen sind:

- Pfad der tatsächlich geladenen Konfiguration;
- Syntax;
- Dateirechte;
- Eigentümer;
- Umgebungsvariablen;
- Include-Dateien;
- relative Pfade;
- Zertifikate und Schlüssel;
- Port und Bind-Adresse;
- Datenbankverbindung;
- DNS-Namen;
- temporäre Verzeichnisse;
- PID- und Socket-Dateien;
- Abhängigkeiten;
- Zeichencodierung;
- Unterschiede zwischen Soll- und Ist-Konfiguration.

Datei anzeigen:

```
sed -n '1,200p' /etc/example/application.conf
```

Dateieigenschaften:

```
stat /etc/example/application.conf
```

Viele Dienste besitzen einen eigenen Syntaxprüfer. Beispiele:

```
sshd -t
```

```
nginx -t
```

```
apachectl configtest
```

Der passende Prüfbefehl muss zur tatsächlich installierten Anwendung und Version passen.

---

19. Dateirechte prüfen

Lange Darstellung:

```
ls -la /var/lib/example
```

Numerische IDs:

```
ls -lan /var/lib/example
```

Alle Bestandteile eines Pfades prüfen:

```
namei -l /var/lib/example/data/file.db
```

Dateistatus:

```
stat /var/lib/example/data/file.db
```

ACL anzeigen:

```
getfacl /var/lib/example/data/file.db
```

Zu prüfen sind:

- Eigentümer;
- Gruppe;
- klassische Modusbits;
- ACL;
- ACL-Maske;
- Rechte aller übergeordneten Verzeichnisse;
- Setuid;
- Setgid;
- Sticky Bit;
- Dateisystem nur lesbar;
- SELinux-Kontext;

- AppArmor-Profil;
- NFS-Einschränkungen;
- Container- oder Namespace-Zuordnung.

Für das Betreten eines Verzeichnisses wird das Ausführungsrecht benötigt. Ausreichende Rechte auf der Datei allein genügen nicht, wenn ein übergeordnetes Verzeichnis nicht durchlaufen werden darf.

---

20. Zugriff als Dienstbenutzer testen

Kontrollierter Lesetest:

```
sudo -u exampleuser -- test -r /var/lib/example/data/file.db
```

Schreibbarkeit eines Verzeichnisses:

```
sudo -u exampleuser -- test -w /var/lib/example/data
```

Pfadzugriff:

```
sudo -u exampleuser -- namei -l /var/lib/example/data/file.db
```

Ein Test sollte möglichst dieselbe Identität, denselben Pfad und dieselbe Aktion wie die Anwendung verwenden.

Keine produktiven Dateien ungeprüft verändern oder überschreiben. Für Schreibtests ist eine dafür vorgesehene Testdatei zu verwenden.

---

21. SELinux prüfen

Status:

```
getenforce
```

Ausführlicher Status:

```
sestatus
```

Dateikontext:

```
ls -lZ /var/lib/example
```

Prozesskontext:

```
ps -eZ
```

Aktuelle AVC-Meldungen:

```
ausearch \
    -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR \
    -ts recent
```

Auswertung, sofern `setroubleshoot` installiert ist:

```
sealert -a /var/log/audit/audit.log
```

Sollkontext eines Pfades prüfen:

```
matchpathcon /var/lib/example/data
```

Kontext anhand der Richtlinie wiederherstellen:

```
restorecon -Rv /var/lib/example
```

SELinux darf nicht pauschal deaktiviert werden, nur um eine Anwendung zum Laufen zu bringen. Eine Ablehnung weist häufig auf einen falschen Pfad, falsches Label, eine unpassende Dienstkonfiguration oder eine fehlende gezielte Richtlinie hin.

Ein Wechsel in den permissiven Modus verändert den Sicherheitszustand des gesamten Systems und ist keine harmlose Standarddiagnose.

---

22. AppArmor prüfen

Status:

```
aa-status
```

Kernel- und Auditmeldungen:

```
journalctl -k | grep -i apparmor
```

Je nach System:

```
grep -i apparmor /var/log/syslog
```

Mögliche Ursachen:

- Anwendung verwendet einen neuen Pfad;
- Profil erlaubt eine benötigte Datei nicht;
- ausführbare Datei wurde verschoben;
- Include-Regel fehlt;
- Profil befindet sich im Enforce-Modus;
- Anwendung startet einen nicht erlaubten Unterprozess;
- Netzwerkzugriff ist eingeschränkt.

Ein Profil sollte nicht ungeprüft deaktiviert werden. Zuerst sind Ablehnung, betroffener Pfad und notwendige Mindestberechtigung zu bestimmen.

---

23. Port und Socket prüfen

Lauschende TCP- und UDP-Sockets:

```
ss -tulpn
```

Nur lauschende TCP-Sockets:

```
ss -ltnp
```

Bestimmter Port:

```
ss -ltnp 'sport = :443'
```

Bestehende TCP-Verbindungen:

```
ss -tanp
```

Unix-Sockets:

```
ss -lxnp
```

Zu unterscheiden sind:

- kein Prozess lauscht;
- Prozess lauscht nur auf `127.0.0.1`;
- Prozess lauscht nur auf IPv6;
- Prozess lauscht auf falscher Adresse;
- anderer Prozess belegt den Port;
- Socket-Aktivierung wird verwendet;
- Firewall blockiert den Zugriff;
- Anwendung lauscht, beantwortet Anfragen aber fehlerhaft.

---

24. Portkonflikt untersuchen

Belegung eines Ports:

```
ss -ltnp 'sport = :8080'
```

Alternativ:

```
lsof -nP -iTCP:8080 -sTCP:LISTEN
```

Mögliche Ursachen:

- zweite Instanz wurde gestartet;
- alter Prozess läuft noch;
- Testdienst belegt den Produktionsport;
- Container veröffentlicht denselben Hostport;
- systemd-Socket besitzt den Port;
- IPv4- und IPv6-Bindung kollidieren;
- Konfiguration verwendet einen falschen Port.

Ein Prozess darf nicht allein deshalb beendet werden, weil er den gewünschten Port belegt. Zuerst muss seine Funktion und Zuständigkeit geklärt werden.

---

25. Lokale Anwendungsfunktion prüfen

TCP-Verbindung:

```
nc -vz 127.0.0.1 8080
```

HTTP-Abfrage:

```
curl -v http://127.0.0.1:8080/
```

HTTPS einschließlich Zertifikatsprüfung:

```
curl -v https://server.example.local/
```

TLS-Verbindung untersuchen:

```
openssl s_client \
    -connect server.example.local:443 \
    -servername server.example.local
```

Bewertung:

Ergebnis	Wahrscheinlicher Bereich
lokal kein Listener	Dienst, Konfiguration oder Startfehler
lokal erreichbar, remote nicht	Firewall, Routing, Bind-Adresse oder Netzwerk
TCP funktioniert, HTTP fehlerhaft	Anwendung, Reverse Proxy oder Backend
HTTP über IP funktioniert, Name nicht	DNS, Virtual Host, SNI oder Zertifikat
TLS schlägt fehl	Zertifikat, Schlüssel, Protokoll oder Cipher
Antwort langsam	Anwendung, Backend, DNS, Storage oder Ressourcen

---

26. Netzwerkkonfiguration prüfen

Adressen:

```
ip address show
```

Kurzform:

```
ip -br address
```

Links:

```
ip -br link
```

Routingtabelle:

```
ip route show
```

IPv6-Routen:

```
ip -6 route show
```

Route zu einem konkreten Ziel:

```
ip route get 192.0.2.20
```

Nachbarn:

```
ip neigh show
```

Netzwerkstatistik:

```
ip -s link show
```

Zu prüfen sind:

- Interface aktiv;
- richtige IP-Adresse;
- Präfix;
- Default Gateway;
- richtige Route;
- Policy Routing;
- VLAN;
- Bond oder Bridge;
- Paketfehler;
- Drops;
- MTU;
- doppelte IP-Adresse;
- IPv4- beziehungsweise IPv6-Auswahl.

---

27. Namensauflösung prüfen

Resolverkonfiguration:

```
cat /etc/resolv.conf
```

Auf systemd-resolved-Systemen:

```
resolvectl status
```

Namensauflösung über NSS:

```
getent hosts server.example.local
```

DNS-Abfrage:

```
dig server.example.local
```

Bestimmten DNS-Server abfragen:

```
dig @192.0.2.53 server.example.local
```

Reverse Lookup:

```
dig -x 192.0.2.20
```

Zu unterscheiden sind:

- DNS-Server nicht erreichbar;
- falsche Suchdomäne;
- falscher A- oder AAAA-Eintrag;
- `/etc/hosts` überschreibt DNS;
- Split DNS;
- Cache enthält alten Wert;
- Anwendung verwendet einen eigenen Resolver;
- IPv6-Adresse wird bevorzugt, ist aber nicht erreichbar;
- DNS-Auflösung funktioniert, die Anwendung verwendet jedoch einen anderen Namen.

---

28. Firewall prüfen

Je nach System können unterschiedliche Werkzeuge verwendet werden.

firewalld:

```
firewall-cmd --get-active-zones
```

```
firewall-cmd --list-all
```

nftables:

```
nft list ruleset
```

iptables-Kompatibilitätsansicht:

```
iptables -S
```

Zu prüfen sind:

- zuständige Firewalltechnik;
- aktive Zone;
- Interface-Zuordnung;
- Quellnetz;
- Zielport;
- Protokoll;
- IPv4 und IPv6;
- Eingangs- und Ausgangsrichtung;
- Standardpolicy;
- NAT;
- Containerregeln;
- Cloud- oder vorgelagerte Firewall;
- Hostfirewall auf dem Gegenserver.

Ein geöffneter Port in der Hostfirewall beweist nicht, dass vorgelagerte Firewalls, Routing oder die Anwendung funktionieren.

---

29. Paketmitschnitt

Beispiel für TCP-Port 443:

```
tcpdump -ni any tcp port 443
```

Bestimmter Host und Port:

```
tcpdump -ni any \
    host 192.0.2.20 and tcp port 443
```

In Datei schreiben:

```
tcpdump -ni any \
    -s 0 \
    -w /tmp/example-443.pcap \
    tcp port 443
```

Zu untersuchen sind:

- kommen Anfragen am Server an;
- sendet der Server Antworten;
- TCP-SYN;
- SYN/ACK;
- Reset;
- Retransmissions;
- ICMP-Fehler;
- falsches Interface;
- falsche Quell- oder Zieladresse;
- TLS-Handshake;
- Zeitabstände;
- MTU- oder Fragmentierungsprobleme.

Paketmitschnitte können vertrauliche IP-Adressen, Hostnamen, Nutzdaten, Tokens und Anmeldeinformationen enthalten und müssen geschützt gespeichert sowie kontrolliert gelöscht werden.

---

30. CPU und Load prüfen

Überblick:

```
uptime
```

Dynamische Prozessansicht:

```
top
```

Logische Prozessoren:

```
nproc
```

CPU-Informationen:

```
lscpu
```

Falls `sysstat` installiert ist:

```
mpstat -P ALL 1 5
```

Prozessbezogene Statistik:

```
pidstat 1 5
```

Load Average beschreibt nicht ausschließlich CPU-Auslastung. Je nach Linux-Zustand können auch Tasks in nicht unterbrechbarem Zustand zur Last beitragen.

Zu unterscheiden sind:

- ein einzelner CPU-intensiver Prozess;
- gleichmäßige Last auf allen CPUs;
- Single-Thread-Limit;
- hohe I/O-Wartezeit;
- virtuelle CPU wird vom Hypervisor nicht ausreichend ausgeführt;
- Interruptlast;
- Prozessschleife;
- zu viele gleichzeitig ausführbare Tasks.

---

31. Arbeitsspeicher prüfen

Speicherübersicht:

```
free -h
```

Detaillierte Kernelwerte:

```
cat /proc/meminfo
```

Prozesse nach Speicherverbrauch:

```
ps aux --sort=-%mem | head
```

Falls verfügbar:

```
vmstat 1 5
```

Wichtige Unterscheidungen:

- `free` ist nicht der allein verfügbare Speicher;
- Cache kann bei Bedarf teilweise freigegeben werden;
- `available` ist für die praktische Bewertung meist aussagekräftiger;
- hoher Swap-Verbrauch beweist allein keine aktuelle Speicherknappheit;
- aktive Swap-Ein- und -Auslagerung kann auf Speicherdruck hinweisen;
- Speicherlimit eines Containers oder einer systemd-Unit kann unterhalb des Hostspeichers liegen.

---

32. OOM Killer untersuchen

Kernelmeldungen:

```
journalctl -k -b | grep -Ei 'out of memory|oom|killed process'
```

Systemweites Journal:

```
journalctl -b | grep -Ei 'out of memory|oom|killed process'
```

Mögliche Ursachen:

- physischer Arbeitsspeicher erschöpft;
- kein oder zu wenig Swap;
- Speicherleck;
- zu viele Prozesse;
- Container- oder cgroup-Limit;
- große temporäre Last;
- falsche Anwendungskonfiguration;
- übermäßiger Page Cache in Verbindung mit anderem Druck;
- Kernel kann benötigten Speicher nicht bereitstellen.

Zu dokumentieren sind:

- beendeter Prozess;

- Zeitpunkt;
- Speicherverbrauch;
- cgroup oder Container;
- Host- und cgroup-Limit;
- vorherige Last;
- wiederholtes Auftreten;
- Anwendungsprotokoll unmittelbar vor dem Kill.

Ein automatischer Dienstneustart kann den OOM-Kill verdecken. Der Dienst erscheint anschließend wieder aktiv, obwohl die Ursache weiter besteht.

---

33. Swap prüfen

Aktive Swap-Bereiche:

```
swapon --show
```

Speicherübersicht:

```
free -h
```

Aktivität:

```
vmstat 1 10
```

Bei `vmstat` sind insbesondere zu beachten:

- `si`: Swap-In;
- `so`: Swap-Out;
- `wa`: I/O-Wartezeit;
- `r`: ausführbare Prozesse;
- `b`: blockierte Prozesse.

Swap darf nicht ungeprüft mit `swapoff` deaktiviert werden. Der Inhalt muss in den Arbeitsspeicher übernommen werden können; andernfalls drohen starke Last oder Fehler.

---

34. Speicherplatz prüfen

Dateisysteme:

```
df -hT
```

Bestimmter Pfad:

```
df -hT /var/lib/example
```

Verzeichnisgrößen:

```
du -xhd1 /var
```

Größte Unterverzeichnisse:

```
du -xhd1 /var |
   sort -h
```

Zu unterscheiden sind:

- Dateisystem voll;
- Thin Pool voll;
- Snapshotbereich voll;
- Quote erreicht;
- gelöschte, aber geöffnete Datei;
- reservierter Speicher;
- anderes darunterliegendes Dateisystem;
- temporäres Dateisystem voll;
- Container-Overlay voll;
- Logdatei wächst unkontrolliert;
- Anwendung schreibt in einen unerwarteten Pfad.

`df` und `du` können unterschiedliche Werte zeigen, beispielsweise wegen gelöschter, aber geöffneter Dateien oder wegen Mounts innerhalb des untersuchten Verzeichnisbaums.

---

35. Inodes prüfen

Inode-Nutzung:

```
df -ih
```

Ein Dateisystem kann freien Speicherplatz besitzen und trotzdem keine neuen Dateien mehr aufnehmen, wenn keine freien Inodes vorhanden sind.

Typische Ursachen:

- sehr viele kleine Dateien;
- Cache-Verzeichnis;
- Mailqueue;
- Sessiondateien;
- temporäre Dateien;
- fehlerhafte Rotation;
- Anwendung erzeugt Dateien, löscht sie aber nicht;
- Containerlayer;
- Build- oder Paketcache.

Anzahl der Einträge in einem kontrollierten Verzeichnisbaum:

```
find /var/tmp/example -xdev -type f |
    wc -l
```

Bei sehr großen Verzeichnisbäumen kann bereits das Zählen erhebliche I/O-Last verursachen.

---

36. Mounts prüfen

Aktuelle Mounts:

```
findmnt
```

Bestimmter Pfad:

```
findmnt -T /var/lib/example
```

Blockgeräte und Dateisysteme:

```
lsblk -f
```

Persistente Konfiguration:

```
cat /etc/fstab
```

fstab-Konfiguration ohne tatsächliches Mounten prüfen:

```
findmnt --verify
```

systemd-Mount-Units:

```
systemctl list-units --type=mount --all
```

Zu prüfen sind:

- richtiges Gerät oder UUID;
- richtiger Mountpoint;
- Dateisystemtyp;
- Mountoptionen;
- `ro` statt `rw`;
- Netzwerkabhängigkeit;
- `_netdev`;
- Automount;
- Berechtigungen;
- verschachtelte Mounts;
- nicht mehr erreichbares NFS- oder SMB-Ziel;
- veralteter Handle;
- Timeout;
- lokales Verzeichnis wird sichtbar, weil der erwartete Mount fehlt.

Eine Anwendung kann scheinbar korrekt in einen lokalen Ordner schreiben, obwohl das vorgesehene Netzwerk- oder Datenvolume nicht eingehängt ist.

---

37. Schreibgeschütztes Dateisystem

Mountoptionen:

```
findmnt -o TARGET,SOURCE,FSTYPE,OPTIONS
```

Kernelmeldungen:

```
journalctl -k -b |
    grep -Ei 'read-only|I/O error|filesystem|ext4|xfs'
```

Mögliche Ursachen:

- Dateisystem wurde wegen Fehlern schreibgeschützt eingehängt;
- Datenträgerfehler;
- Storage-Verbindung unterbrochen;
- falsche Mountoption;
- Snapshot oder Image ist absichtlich read-only;

- Container-Dateisystem ist schreibgeschützt;
- NFS-Export ist read-only;
- Cluster- oder Schutzmechanismus verhindert Schreibzugriff.

Ein ungeprüftes Remount als `rw` kann eine bestehende Storage- oder Dateisystemstörung verschärfen. Zuerst müssen Kernelmeldungen und Datenträgerzustand bewertet werden.

---

38. Datenträger- und I/O-Fehler prüfen

Blockgeräte:

```
lsblk -o NAME,TYPE,SIZE,FSTYPE,MOUNTPOINTS,MODEL,SERIAL
```

Kernelmeldungen:

```
journalctl -k -b |
    grep -Ei 'I/O error|blk_update|reset|timeout|nvme|ata|scsi|ext4|xfs'
```

Falls `sysstat` installiert ist:

```
iostat -xz 1 5
```

S.M.A.R.T.-Informationen, sofern unterstützt:

```
smartctl -a /dev/sda
```

NVMe-Informationen, sofern das Werkzeug installiert ist:

```
nvme smart-log /dev/nvme0
```

Zu prüfen sind:

- I/O-Fehler;
- Timeouts;
- Resets;
- Medienfehler;
- hohe Latenz;
- hohe Queue;
- fehlendes Gerät;
- Multipath-Fehler;

- RAID-Degradation;
- Dateisystemfehler;
- virtuelle oder externe Storage-Abhängigkeit.

S.M.A.R.T.-Tests und Reparaturwerkzeuge können Last oder Zustandsänderungen verursachen. Sie sind nicht ungeprüft auf produktivem Storage auszuführen.

---

39. Dateisystemprüfung sicher planen

Dateisystemtyp feststellen:

```
findmnt -no FSTYPE,SOURCE,TARGET /var/lib/example
```

Ein klassisches `fsck` darf normalerweise nicht auf einem schreibend eingehängten Dateisystem ausgeführt werden.

Vor einer Reparatur sind zu klären:

- Dateisystemtyp;
- Herstellerverfahren;
- Mountzustand;
- Backup;
- Wartungsfenster;
- Ausfallwirkung;
- Storage- und RAID-Zustand;
- erwartete Reparaturdauer;
- Rückweg;
- Notfallzugang.

Bei XFS, ext4, Btrfs und anderen Dateisystemen unterscheiden sich Diagnose- und Reparaturwerkzeuge erheblich.

---

40. LVM prüfen

Übersicht:

```
pvs
```

```
vgs
```

```
lvs
```

Ausführlicher:

```
lvs -a -o +devices
```

Zu prüfen sind:

- Physical Volume vorhanden;
- Volume Group vollständig;
- Logical Volume aktiv;
- freier Platz in der Volume Group;
- Thin Pool;
- Thin-Metadaten;
- Snapshot;
- zugrunde liegende Geräte;
- Dateisystemgröße;
- Mountzustand.

Ein vergrößertes Logical Volume bedeutet nicht automatisch, dass auch das darin enthaltene Dateisystem vergrößert wurde.

---

41. Software-RAID prüfen

Status:

```
cat /proc/mdstat
```

Detailinformationen:

```
mdadm --detail /dev/md0
```

Zu prüfen sind:

- RAID-Level;
- aktive Geräte;
- fehlende Geräte;
- degradiert;
- Rebuild;
- Resync;
- Geschwindigkeit;
- Fehlerzähler;
- Ersatzgerät;
- zugrunde liegender Datenträgerzustand.

Ein Rebuild erhöht die I/O-Last und ersetzt kein Backup. Ein zweiter Fehler während der Wiederherstellung kann je nach RAID-Level zum Datenverlust führen.

---

42. Netzwerk-Mounts prüfen

NFS-Mounts:

```
findmnt -t nfs,nfs4
```

SMB-/CIFS-Mounts:

```
findmnt -t cifs
```

RPC-Dienste eines NFS-Servers:

```
rpcinfo -p <Server>
```

NFS-Exporte, sofern erreichbar:

```
showmount -e <Server>
```

Zu prüfen sind:

- Namensauflösung;
- Route;
- Port;
- Serverzustand;
- Export oder Freigabe;
- Protokollversion;
- Mountoptionen;
- Benutzer- und ID-Zuordnung;
- Kerberos;
- Firewall;
- Timeout;
- Hard- oder Soft-Mount;
- veralteter NFS-Handle;
- blockierte Prozesse im Zustand `D`.

Ein hängender Netzwerk-Mount kann Befehle wie `df`, `du`, `ls` oder Anwendungen blockieren.

---

43. Paketverwaltung prüfen

Debian- und Ubuntu-Systeme:

```
apt update
```

Installierten Paketstatus prüfen:

```
dpkg -l
```

Unterbrochene Paketkonfiguration:

```
dpkg --audit
```

RPM-basierte Systeme:

```
dnf check
```

Installiertes Paket bestimmen:

```
rpm -q <Paketname>
```

Datei einem Paket zuordnen:

```
rpm -qf /usr/bin/example
```

Debian-Paket zuordnen:

```
dpkg -S /usr/bin/example
```

Mögliche Ursachen:

- Repository nicht erreichbar;
- DNS- oder Proxyfehler;
- abgelaufenes Repository-Zertifikat;
- falsche Systemzeit;
- Signaturfehler;
- Paketdatenbank gesperrt;
- abgebrochene Transaktion;
- inkompatible Paketversionen;
- fehlende Bibliothek;

- Drittanbieter-Repository;
- Paket wurde manuell überschrieben;
- Konfigurationsdatei wurde bei einem Upgrade ersetzt oder nicht übernommen.

Paketdatenbanken und Lockdateien dürfen nicht ungeprüft gelöscht werden. Zuerst ist zu bestimmen, ob ein legitimer Paketprozess aktiv ist.

---

44. Paketdateien verifizieren

RPM-Paket:

```
rpm -V <Paketname>
```

Debian- und Ubuntu-Systeme, falls `debsums` installiert ist:

```
debsums <Paketname>
```

Abweichungen können entstehen durch:

- legitime Konfigurationsänderung;
- Update;
- beschädigte Datei;
- manuelle Änderung;
- Sicherheitsvorfall;
- falsche Berechtigung;
- falscher Eigentümer.

Eine Abweichung beweist allein weder einen Defekt noch eine Manipulation. Dateityp und erwartete lokale Anpassung müssen berücksichtigt werden.

---

45. Cronjobs prüfen

Systemweite Crontab:

```
cat /etc/crontab
```

Systemverzeichnisse:

```
ls -la /etc/cron.d/
```

Benutzer-Crontab:

```
crontab -l
```

Crontab eines anderen Benutzers:

```
sudo crontab -u exampleuser -l
```

Cron-Dienst:

```
systemctl status cron
```

Je nach Distribution:

```
systemctl status crond
```

Cron-Protokoll:

```
journalctl -u cron
```

Oder:

```
journalctl -u crond
```

Typische Ursachen:

- falscher Benutzer;
- eingeschränkte `PATH`-Variable;
- relatives Arbeitsverzeichnis;
- fehlende Umgebungsvariable;
- falsche Zeitzone;
- Shell-Unterschied;
- fehlendes Ausführungsrecht;
- Skript verwendet interaktive Eingaben;
- Ausgabe und Fehler werden nicht protokolliert;
- Datei besitzt falsche Zeilenenden;
- überlappende Ausführungen;
- Zielmount ist zum Ausführungszeitpunkt nicht verfügbar.

Ein erfolgreicher manueller Skriptstart beweist nicht, dass der Cronjob mit seiner reduzierten Umgebung funktioniert.

---

46. systemd-Timer prüfen

Timer anzeigen:

```
systemctl list-timers --all
```

Bestimmten Timer:

```
systemctl status example.timer
```

Zugehörigen Dienst:

```
systemctl status example.service
```

Protokolle:

```
journalctl -u example.timer -u example.service
```

Konfiguration:

```
systemctl cat example.timer
```

Zu prüfen sind:

- Timer aktiv;
- Timer aktiviert;
- nächste Ausführung;
- letzte Ausführung;
- zugehörige Service-Unit;
- Kalenderausdruck;
- `Persistent=`;
- Zufallsverzögerung;
- Zeitzone;
- Fehler der aufgerufenen Service-Unit.

Der Timer kann erfolgreich ausgelöst haben, obwohl die zugehörige Service-Unit fehlgeschlagen ist.

---

47. Zeit und Zeitsynchronisation prüfen

Systemzeit:

```
date --iso-8601=seconds
```

Status:

```
timedatectl
```

Bei chrony:

```
chronyc tracking
```

```
chronyc sources -v
```

Bei systemd-timesyncd:

```
timedatectl timesync-status
```

Mögliche Auswirkungen falscher Zeit:

- TLS-Zertifikate erscheinen ungültig;
- Kerberos schlägt fehl;
- Protokollereignisse lassen sich falsch zuordnen;
- Tokens sind noch nicht oder nicht mehr gültig;
- Cronjobs laufen zum falschen Zeitpunkt;
- Replikation oder Clustermechanismen werden gestört;
- Dateien besitzen unplausible Zeitstempel.

Die Zeit darf auf produktiven Datenbank-, Cluster- oder Authentifizierungssystemen nicht ungeprüft sprunghaft verändert werden.

---

48. Zertifikate prüfen

Zertifikatsdatei:

```
openssl x509 \
    -in /etc/example/server.crt \
    -noout \
    -subject \
```

```
    -issuer \
    -dates \
    -fingerprint
```

Private-Key-Dateirechte:

```
stat /etc/example/server.key
```

Zertifikat und Schlüssel vergleichen, beispielhaft für RSA:

```
openssl x509 \
    -noout \
    -modulus \
    -in /etc/example/server.crt |
    openssl sha256
```

```
openssl rsa \
    -noout \
    -modulus \
    -in /etc/example/server.key |
    openssl sha256
```

Zu prüfen sind:

- Ablaufdatum;
- Gültigkeitsbeginn;
- Hostname beziehungsweise SAN;
- Aussteller;
- Zertifikatskette;
- passender privater Schlüssel;
- Dateirechte;
- tatsächlich vom Dienst geladenes Zertifikat;
- SNI;
- Systemzeit;
- vertrauenswürdige CA.

Private Schlüssel dürfen nicht in Diagnoseprotokolle oder Tickets kopiert werden.

---

49. Kernelmeldungen prüfen

Aktueller Systemstart:

```
journalctl -k -b
```

Warnungen und Fehler:

```
journalctl -k -b -p warning
```

Klassische Anzeige:

```
dmesg --level=emerg,alert,crit,err,warn
```

Mögliche relevante Meldungen:

- I/O-Fehler;
- Dateisystemfehler;
- OOM;
- Treiberfehler;
- Linkverlust;
- NIC-Reset;
- SCSI- oder NVMe-Timeout;
- Hardwarefehler;
- Kernel-Warnung;
- Soft Lockup;
- Hard Lockup;
- Hung Task;
- Segmentation Fault;
- AppArmor- oder SELinux-Ablehnung.

Der `dmesg`-Puffer kann ältere Meldungen überschreiben. Das persistente Journal oder ein zentraler Logserver kann deshalb zusätzliche Informationen liefern.

---

50. Kernelmodule und Hardware prüfen

Geladene Module:

```
lsmod
```

Modulinformationen:

```
modinfo <Modulname>
```

PCI-Geräte und Treiber:

```
lspci -k
```

USB-Geräte:

```
lsusb
```

Hardwareübersicht:

```
lscpu
```

```
lsblk
```

Mögliche Ursachen:

- benötigtes Modul fehlt;
- falsches Modul ist gebunden;
- Firmware fehlt;
- Kernelupdate passt nicht zu externem Modul;
- DKMS-Build fehlgeschlagen;
- Gerät wurde getrennt;
- virtuelle Hardware wurde geändert;
- Modulparameter sind falsch;
- Secure Boot verhindert das Laden eines nicht vertrauenswürdigen Moduls.

Kernelmodule dürfen nicht ungeprüft entladen werden. Ein Modul kann von Netzwerk-, Storage- oder anderen produktiven Geräten verwendet werden.

---

51. Bootfehler untersuchen

Bootzeit analysieren:

```
systemd-analyze
```

Langsame Units:

```
systemd-analyze blame
```

Kritische Kette:

```
systemd-analyze critical-chain
```

Fehlgeschlagene Units:

```
systemctl --failed
```

Protokoll des aktuellen Starts:

```
journalctl -b
```

Protokoll des vorherigen Starts:

```
journalctl -b -1
```

Verfügbare Starts:

```
journalctl --list-boots
```

Zu prüfen sind:

- Kernelkommandozeile;
- initramfs;
- Root-Dateisystem;
- fehlerhafte `/etc/fstab`;
- fehlendes Blockgerät;
- Netzwerk-Mount;
- systemd-Abhängigkeit;
- Timeout;
- Dateisystemprüfung;
- Treiber;
- verschlüsseltes Volume;
- falscher Default-Target;
- wiederholt abstürzender Dienst.

---

52. Unerwarteten Neustart oder Absturz prüfen

Letzte Neustarts:

```
last reboot
```

Vorheriger Boot:

```
journalctl -b -1 -e
```

Kernelmeldungen des vorherigen Boots:

```
journalctl -k -b -1
```

Mögliche Ursachen:

- geplanter Neustart;
- Paket- oder Kernelupdate;
- Stromausfall;
- Hypervisoraktion;
- Hardware-Watchdog;
- Kernel Panic;
- OOM;
- Temperaturproblem;
- manuelle Administratoraktion;
- Cloud- oder Hostingereignis;
- fehlerhafte Automation.

Fehlende Abschlussmeldungen im Journal können auf einen abrupten Ausfall hinweisen, beweisen aber keine bestimmte Ursache.

---

53. Hängendes System und Hung Tasks

Nach Meldungen suchen:

```
journalctl -k |
    grep -Ei 'blocked for more than|hung task|soft lockup|hard lockup'
```

Mögliche Ursachen:

- blockiertes Storage;
- hängender Netzwerk-Mount;
- Treiberfehler;

- defektes Gerät;
- Kernelproblem;
- hohe I/O-Latenz;
- Deadlock;
- Ressourcenerschöpfung.

Magic SysRq kann in schweren Störungsfällen Diagnose- oder Notfallfunktionen auslösen. Die verfügbaren Funktionen hängen von Kernelkonfiguration und Systemwerten ab.

Status:

```
cat /proc/sys/kernel/sysrq
```

SysRq-Aktionen können erhebliche Auswirkungen bis hin zu Prozessbeendigung oder Neustart besitzen. Sie dürfen nur nach einem dafür vorgesehenen Notfallverfahren verwendet werden.

---

54. Strace als gezielte Tiefenanalyse

Systemaufrufe eines gestarteten Programms:

```
strace -f -o /tmp/example.strace <Befehl>
```

An laufenden Prozess anhängen:

```
strace -f -p <PID>
```

Nach Dateioperationen filtern:

```
strace -f -e trace=file <Befehl>
```

Nach Netzwerkoperationen filtern:

```
strace -f -e trace=network <Befehl>
```

Zu erkennen sind beispielsweise:

- `ENOENT`: Datei oder Pfad nicht vorhanden;
- `EACCES`: Zugriff verweigert;
- `ECONNREFUSED`: Verbindung abgelehnt;
- `ETIMEDOUT`: Zeitüberschreitung;

- `ENOSPC`: kein Speicherplatz;
- `EROFS`: Dateisystem nur lesbar.

`strace` kann Leistung und Zeitverhalten beeinflussen sowie vertrauliche Dateinamen, Argumente und Daten sichtbar machen.

---

55. Typische Fehlerbilder

Symptom	Mögliche Ursache	Nächster Test
Dienst startet nicht	Syntax, Rechte, Abhängigkeit oder fehlende Datei	`systemctl status`, `journalctl -u`
Dienst beendet sich sofort	Anwendung meldet Fehler oder Prozess forkt anders als erwartet	Journal und `ExecMainStatus`
Dienst ist `active`, Anwendung funktioniert nicht	interner Fehler, falscher Port oder Backend gestört	`ss`, `curl`, Anwendungslog
Dienst ist `failed`, Prozess läuft	Prozess wurde außerhalb von systemd gestartet	`ps`, `systemctl show`
Dienst startet nach Reboot nicht	nicht aktiviert oder Boot-Abhängigkeit fehlerhaft	`is-enabled`, Bootjournal
Dienst läuft als falscher Benutzer	Unit oder Drop-in geändert	`systemctl cat`, `systemctl show`
Konfigurationsänderung wirkt nicht	falsche Datei oder kein Reload/Restart	Prozessargumente und Unit prüfen
Port ist nicht geöffnet	Dienststart oder Bind-Konfiguration	`ss -ltnp`
Port ist nur lokal erreichbar	Bind-Adresse oder Firewall	`ss`, Firewall, Remote-Test
Port bereits belegt	zweite Instanz oder anderer Dienst	`ss`, `lsof`
Zugriff verweigert	Unix-Rechte, ACL, SELinux oder AppArmor	`namei`, `getfacl`, Auditlog
Datei nicht gefunden, obwohl sie existiert	falscher Namespace, Mount oder Pfadbestandteil	`namei`, `findmnt`, Prozessumgebung
Schreiben schlägt fehl	Rechte, `ro`, Speicher oder Inodes	`test -w`, `findmnt`, `df -h`, `df -i`
`df` zeigt voll, `du` nicht	gelöschte offene Datei oder reservierter Bereich	`lsof +L1`
`du` hängt	Netzwerk-Mount oder Storageproblem	`findmnt`, Prozesszustand, Kerneljournal
hohe Load bei geringer CPU	I/O-Wait oder Tasks im Zustand `D`	`vmstat`, `iostat`, `ps`
Prozess wurde unerwartet beendet	OOM, Signal oder Crash	Kernel- und Dienstjournal
Server nutzt Swap stark	früherer oder aktueller Speicherdruck	`vmstat`, `free`, Prozessspeicher
keine neuen Dateien trotz freiem Speicher	Inodes erschöpft	`df -ih`
Mount fehlt nach Neustart	fstab, Abhängigkeit oder Gerät fehlt	`findmnt --verify`, Bootjournal

Symptom	Mögliche Ursache	Nächster Test
Dateisystem plötzlich read-only	I/O- oder Dateisystemfehler	Kerneljournal
Cronjob läuft manuell, aber nicht automatisch	Umgebung, Benutzer, Pfad oder Shell	Cronlog und kontrollierter Umgebungstest
systemd-Timer löst aus, Aufgabe scheitert	zugehörige Service-Unit fehlerhaft	Journal beider Units
TLS-Zertifikat angeblich abgelaufen	falsche Zeit oder altes geladenes Zertifikat	`timedatectl`, `openssl s_client`
Paketupdate schlägt fehl	Repository, DNS, Signatur oder Lock	Paketmanager- und Prozessstatus
Server startet langsam	Mount-, Netzwerk- oder Unit-Timeout	`systemd-analyze critical-chain`
Netzwerk funktioniert nach IP, nicht nach Name	DNS oder NSS	`getent hosts`, `dig`

---

56. Vorgehensweise bei „Dienst startet nicht“

1. Host, Dienstname und Zeitpunkt dokumentieren.
2. `systemctl status` auswerten.
3. `systemctl show` auf Ergebnis und Exit-Status prüfen.
4. Dienstjournal seit dem letzten Startversuch lesen.
5. erste relevante Fehlermeldung bestimmen.
6. tatsächlich verwendete Unit und Drop-ins anzeigen.
7. Startbenutzer und Gruppen prüfen.
8. Anwendungskonfiguration mit dem vorgesehenen Syntaxprüfer testen.
9. benötigte Dateien und Pfade prüfen.
10. Unix-Rechte, ACL und alle Pfadbestandteile prüfen.
11. SELinux- oder AppArmor-Ablehnungen prüfen.
12. Portkonflikte prüfen.
13. Mounts und externe Abhängigkeiten prüfen.
14. Speicherplatz, Inodes, RAM und Limits kontrollieren.
15. genau eine nachgewiesene Ursache korrigieren.
16. Dienst kontrolliert starten.
17. Journal, Prozess, Port und Anwendungsfunktion nachkontrollieren.
18. Ursache, Maßnahme und Rückweg dokumentieren.

---

57. Vorgehensweise bei „Server ist langsam“

1. Zeitraum und konkrete langsame Funktion erfassen.
2. Vergleichswert oder normalen Zustand bestimmen.
3. Load, CPU, RAM und Swap prüfen.
4. Prozesse nach CPU- und Speicherverbrauch sortieren.
5. Prozesszustände und blockierte Tasks prüfen.
6. `vmstat` auswerten.

7. I/O-Latenz und Datenträgerauslastung prüfen.
8. Kernelmeldungen auf Storage- oder Treiberfehler untersuchen.
9. Netzwerkfehler, Drops und Retransmissions berücksichtigen.
10. DNS-Antwortzeiten prüfen.
11. Anwendung und Backend getrennt testen.
12. cgroup-, Container- oder Unit-Limits prüfen.
13. geplante Jobs, Backups und Updates berücksichtigen.
14. Engpass anhand von Messwerten nachweisen.
15. eine kontrollierte Maßnahme durchführen.
16. dieselbe Funktion mit denselben Messwerten erneut testen.

---

58. Vorgehensweise bei „Kein Speicherplatz“

1. betroffenen Pfad dokumentieren.
2. zugehöriges Dateisystem mit `findmnt -T` bestimmen.
3. Speicherplatz mit `df -hT` prüfen.
4. Inodes mit `df -ih` prüfen.
5. Quotas und Storage-Limits berücksichtigen.
6. Verzeichnisgrößen innerhalb desselben Dateisystems untersuchen.
7. gelöschte offene Dateien mit `lsof +L1` prüfen.
8. Logs, Caches, temporäre Dateien und Containerdaten unterscheiden.
9. LVM-Thin-Pool oder Snapshotbereich prüfen.
10. Ursache des Wachstums bestimmen.
11. Aufbewahrungs- und Löschregeln prüfen.
12. keine unbekannten Dateien pauschal löschen.
13. eine kontrollierte Bereinigung oder Erweiterung durchführen.
14. Anwendung und Speicherzustand erneut prüfen.
15. Präventionsmaßnahme wie Rotation, Monitoring oder Quota dokumentieren.

---

59. Vorgehensweise bei „Dateisystem oder Mount nicht verfügbar“

1. betroffenen Mountpoint dokumentieren.
2. `findmnt` und `lsblk -f` auswerten.
3. Blockgerät, UUID und Dateisystemtyp bestimmen.
4. `/etc/fstab` prüfen.
5. `findmnt --verify` ausführen.
6. systemd-Mount-Unit und Journal prüfen.
7. Kernelmeldungen auf I/O-Fehler untersuchen.
8. bei Netzwerk-Mounts DNS, Route, Port und Server prüfen.
9. offene Prozesse und Abhängigkeiten berücksichtigen.
10. Storage-, RAID-, LVM- oder Multipath-Zustand prüfen.
11. Backup und Wartungsfenster klären.
12. nur die nachgewiesene Ursache korrigieren.
13. Mount kontrolliert herstellen.
14. Lese- und Schreibfunktion entsprechend dem Sollzustand testen.

15. Neustartverhalten gesondert verifizieren.

---

60. Maßnahmen und Rückwege

Ursache: Syntaxfehler in der Dienstkonfiguration

Nachweis:

- Dienstjournal nennt die Konfigurationsdatei und Zeile;
- anwendungseigener Syntaxprüfer reproduziert den Fehler;
- vorherige Konfiguration war funktionsfähig.

Maßnahme:

- fehlerhafte Anweisung nach dokumentiertem Sollzustand korrigieren.

Rollback:

- gesicherte vorherige Konfiguration wiederherstellen.

Verifikation:

- Syntaxprüfung;
- Dienststart;
- Journal;
- Port;
- praktische Anwendungsfunktion.

Ursache: falsche Dateiberechtigung

Nachweis:

- Dienst läuft unter dem vorgesehenen Benutzer;
- benötigte Aktion schlägt für diesen Benutzer fehl;
- `namei`, `stat` oder `getfacl` zeigt das fehlende Recht;
- kein SELinux- oder AppArmor-Problem liegt vor.

Maßnahme:

- Eigentümer, Gruppe oder Mindestberechtigung nach dem Sollzustand korrigieren.

Rollback:

- vorherige Eigentümer-, Modus- und ACL-Werte dokumentiert wiederherstellen.

Verifikation:

- Zugriff als Dienstbenutzer;
- Dienststart;
- keine unnötigen zusätzlichen Rechte;
- Anwendungsfunktion.

Ursache: falscher SELinux-Kontext

Nachweis:

- AVC-Meldung passt zum Fehlerzeitpunkt;
- Pfad besitzt einen vom Richtlinien-Soll abweichenden Kontext;
- Unix-Rechte sind ausreichend.

Maßnahme:

- vorgesehenen persistenten Dateikontext konfigurieren und Kontext kontrolliert wiederherstellen.

Rollback:

- vorherige lokale Kontextregel dokumentiert wiederherstellen oder entfernen.

Verifikation:

- korrekter Kontext;
- keine neue AVC-Ablehnung;
- Dienst- und Benutzerfunktion;
- SELinux bleibt enforcing.

Ursache: Portkonflikt

Nachweis:

- Anwendung meldet „Address already in use“;
- `ss` oder `lsof` zeigt den belegenden Prozess;
- Prozess und Zuständigkeit wurden identifiziert.

Maßnahme:

- falsche Doppelinstanz kontrolliert entfernen oder den vorgesehenen Dienstport nach Sollkonfiguration zuordnen.

Rollback:

- ursprüngliche Port- und Dienstkonfiguration wiederherstellen.

Verifikation:

- vorgesehener Prozess besitzt den Port;
- Anwendung antwortet lokal und remote;
- keine zweite Anwendung wurde beeinträchtigt.

Ursache: gelöschte offene Logdatei belegt Speicher

Nachweis:

- `df` zeigt hohe Belegung;
- `du` erklärt die Belegung nicht;
- `lsof +L1` zeigt eine große gelöschte Datei;
- zuständiger Prozess wurde bestimmt.

Maßnahme:

- Prozess nach betrieblicher Freigabe kontrolliert zum erneuten Öffnen seiner Logs veranlassen oder neu starten.

Rollback:

- bei einem reinen Reopen normalerweise nicht erforderlich; bei Neustart gilt das dienstbezogene Rückfallverfahren.

Verifikation:

- Dateideskriptor geschlossen;
- Speicherplatz freigegeben;
- neue Protokolldatei wird korrekt beschrieben;
- Dienstfunktion bleibt erhalten.

---

61. Nachkontrolle

Nach einer Maßnahme sind mindestens folgende Punkte zu prüfen:

- Dienst besitzt den erwarteten Zustand;
- Prozess läuft unter dem vorgesehenen Benutzer;
- Prozess wird nicht fortlaufend neu gestartet;
- keine neuen Fehler erscheinen im Journal;
- erwarteter Port wird von der richtigen Anwendung verwendet;
- Bind-Adresse ist korrekt;
- lokale Anwendungsfunktion funktioniert;
- Remotezugriff funktioniert;
- DNS-Name zeigt auf das richtige System;

- Firewallzustand entspricht dem Soll;
- benötigte Dateien sind erreichbar;
- Eigentümer und Berechtigungen sind minimal und korrekt;
- SELinux oder AppArmor bleibt wirksam;
- erforderliche Mounts sind vorhanden;
- Dateisystem ist im vorgesehenen Modus eingehängt;
- Speicherplatz und Inodes sind ausreichend;
- keine gelöschten großen Dateien bleiben geöffnet;
- CPU, RAM, Swap und I/O sind plausibel;
- keine neuen Kernel- oder Storagefehler erscheinen;
- Timer oder Cronjob arbeitet auch automatisch;
- Verhalten nach einem Neustart wurde berücksichtigt;
- temporäre Diagnoseänderungen wurden zurückgenommen;
- Ursache, Maßnahme, Rückweg und Ergebnis wurden dokumentiert.

---

62. Dokumentationsbeispiel

Symptom:
Der Dienst example-api ließ sich nach einem Konfigurationsupdate nicht mehr starten. Clients erhielten „Connection refused“ auf TCP-Port 8443.

Zeitpunkt:
02.08.2026, 11:42 Uhr MESZ

Server:
linux-app-01.example.local

Dienst:
example-api.service

Nachweis:
systemctl status zeigte den Zustand „failed“.
Im Journal war unmittelbar nach dem Startversuch ein Berechtigungsfehler für /etc/example-api/server.key dokumentiert.
Der Dienst lief als Benutzer example-api.
Ein kontrollierter Lesetest als example-api schlug fehl.
Die übergeordneten Verzeichnisse waren zugänglich.
Es gab keine passende SELinux- oder AppArmor-Ablehnung.
Die Schlüsseldatei war nach dem Austausch irrtümlich root:root mit Modus 0600 zugeordnet worden.

Ursache:
Die neue private Schlüsseldatei besaß nicht den vorgesehenen Gruppeneigentümer.
Der Dienstbenutzer konnte sie deshalb nicht lesen.

Maßnahme:
Nach Dokumentation des Ausgangszustands wurde die Datei der vorgesehenen Dienstgruppe zugeordnet. Der Modus blieb auf das erforderliche Minimum beschränkt.

Rollback:
Der ursprüngliche Eigentümer und die ursprüngliche Gruppe wurden dokumentiert und können wiederhergestellt werden.

Verifikation:
Der Konfigurationstest war erfolgreich.
Der Dienst startete ohne neue Fehlermeldung.
TCP-Port 8443 wurde durch den vorgesehenen Prozess geöffnet.
Die lokale und externe HTTPS-Prüfung waren erfolgreich.
Der private Schlüssel ist weiterhin nicht für andere Benutzer lesbar.

Prävention:
Der Zertifikatsaustausch wird künftig durch einen dokumentierten Ablauf mit Kontrolle von Eigentümer, Gruppe, Modus, Sicherheitskontext und anschließendem Konfigurationstest durchgeführt.

---

63. Entscheidungsbaum

```
Linux-Dienst oder Anwendung gestört
        ↓
Ist das System erreichbar und die Zeit plausibel?
        ├── Nein
        │   ↓
        │   Boot, Netzwerk, Routing, DNS und Zeit prüfen
        │
        └── Ja
            ↓
Ist die Unit aktiv?
        ├── Nein
```

```
    │  ↓
    │  Status, Exit-Code und Dienstjournal prüfen
    │
    └── Ja
      ↓
Läuft der erwartete Prozess unter der richtigen Identität?
    ├── Nein
    │  ↓
    │  Unit, Drop-ins, Benutzer und Startmethode prüfen
    │
    └── Ja
      ↓
Lauscht die Anwendung auf der vorgesehenen Adresse und dem Port?
    ├── Nein
    │  ↓
    │  Konfiguration, Portkonflikt und Socket-Aktivierung prüfen
    │
    └── Ja
      ↓
Funktioniert die Anwendung lokal?
    ├── Nein
    │  ↓
    │  Anwendung, Backend, Rechte, Mounts und Ressourcen prüfen
    │
    └── Ja
      ↓
Funktioniert der Zugriff über das Netzwerk?
    ├── Nein
    │  ↓
    │  Bind-Adresse, Firewall, Route, DNS und Paketfluss prüfen
    │
    └── Ja
      ↓
Benutzerfunktion, Lastverhalten und abhängige Systeme prüfen
```

---

64. Typische Prüfungsfragen

Warum reicht der Zustand `active` eines systemd-Dienstes nicht als vollständiger Funktionsnachweis?

Antwort anzeigen

Der Zustand zeigt, dass systemd die Unit als aktiv betrachtet. Er beweist nicht, dass die Anwendung auf dem richtigen Port lauscht, ihre Backends erreicht oder korrekte Antworten auf Benutzeranfragen liefert.

Worin unterscheiden sich `enabled` und `active`?

Antwort anzeigen

`enabled` beschreibt die Einbindung für einen automatischen Start. `active` beschreibt den aktuellen Laufzeitzustand. Ein Dienst kann aktiv und gleichzeitig nicht aktiviert sein oder aktiviert, aber aktuell fehlgeschlagen sein.

Warum sollte zuerst `SIGTERM` und nicht sofort `SIGKILL` verwendet werden?

Antwort anzeigen

`SIGTERM` ermöglicht dem Prozess eine kontrollierte Beendigung, beispielsweise das Schreiben von Daten und Schließen von Dateien. `SIGKILL` kann nicht behandelt werden und verhindert eine geordnete Bereinigung.

Warum können `df` und `du` unterschiedliche Speicherbelegungen anzeigen?

Antwort anzeigen

Eine häufige Ursache sind gelöschte Dateien, die von einem laufenden Prozess weiterhin geöffnet gehalten werden. `du` findet sie nicht mehr im Verzeichnisbaum, während das Dateisystem den belegten Speicher weiterhin berücksichtigt.

Warum kann ein Dateisystem trotz freiem Speicherplatz keine neue Datei aufnehmen?

Antwort anzeigen

Die verfügbaren Inodes können erschöpft sein. In diesem Fall ist noch Datenkapazität frei, aber es können keine weiteren Dateisystemobjekte angelegt werden.

Warum beweist ein erfolgreicher manueller Skriptstart nicht, dass ein Cronjob funktioniert?

Antwort anzeigen

Cron verwendet häufig eine reduzierte Umgebung, einen anderen Benutzer, ein anderes Arbeitsverzeichnis und eine eingeschränkte `PATH`-Variable. Das Skript kann deshalb manuell funktionieren und automatisch scheitern.

Warum sollte SELinux nicht pauschal deaktiviert werden?

Antwort anzeigen

Dadurch wird eine wichtige Sicherheitsschicht für das gesamte System entfernt, ohne die eigentliche Ursache zu beheben. Stattdessen müssen Ablehnung, Pfad, Dateikontext und notwendige Mindestberechtigung untersucht werden.

Was bedeutet ein Prozess im Zustand `D`?

Antwort anzeigen

Der Prozess befindet sich in einem nicht unterbrechbaren Schlafzustand und wartet häufig auf eine I/O- oder Kerneloperation. Die zugrunde liegende Storage-, Mount- oder Treiberstörung muss untersucht werden.

Warum kann ein fehlender Mount zu Daten am falschen Ort führen?

Antwort anzeigen

Wenn der vorgesehene Mount fehlt, bleibt das lokale Mountpoint-Verzeichnis sichtbar. Eine Anwendung kann dann unbemerkt in das lokale Root-Dateisystem statt auf das vorgesehene Volume schreiben.

---

65. Prüfungsfallen

- einen Dienst sofort neu starten, bevor Status und Protokolle gesichert wurden;
- `active` mit vollständiger Anwendungsfunktion gleichsetzen;
- `enabled` mit aktuell laufend gleichsetzen;
- nur die letzte statt der ersten ursächlichen Fehlermeldung betrachten;
- Unit-Drop-ins nicht berücksichtigen;
- `After=` mit einer Startabhängigkeit gleichsetzen;
- Prozess außerhalb von systemd starten und dadurch den Sollzustand verfälschen;
- sofort `SIGKILL` verwenden;

- Prozess im Zustand `D` wie einen normalen hängenden Prozess behandeln;
- Zombieprozess ungeprüft als Hauptursache hoher Last ansehen;
- Anwendung als `root` starten, um Berechtigungsfehler zu umgehen;
- pauschal Modus `777` vergeben;
- nur die Dateirechte, aber nicht die übergeordneten Verzeichnisse prüfen;
- ACL-Maske nicht berücksichtigen;
- SELinux oder AppArmor pauschal deaktivieren;
- Auditmeldungen ohne zeitlichen Zusammenhang bewerten;
- Portöffnung mit funktionierender Anwendung gleichsetzen;
- nur localhost testen;
- nur Remotezugriff testen, ohne die lokale Anwendung zu prüfen;
- nur IPv4 und nicht IPv6 berücksichtigen;
- `/etc/hosts`, NSS und DNS verwechseln;
- Firewallregel ändern, ohne Zone und Interface zu prüfen;
- Paketmitschnitte ungeschützt speichern;
- Load Average mit reiner CPU-Auslastung gleichsetzen;
- niedrigen Wert bei `free` automatisch als Speichermangel bewerten;
- hohen Swap-Verbrauch ohne aktuelle Swap-Aktivität als Fehler ansehen;
- OOM-Kill übersehen, weil systemd den Dienst neu gestartet hat;
- nur Speicherplatz, aber keine Inodes prüfen;
- Dateien allein nach Größe ungeprüft löschen;
- Logdatei löschen, obwohl der Prozess sie weiterhin geöffnet hält;
- `df` oder `du` auf hängenden Netzwerk-Mounts ungeprüft ausführen;
- fehlenden Mount übersehen und in das lokale Mountpoint-Verzeichnis schreiben;
- ein read-only-Dateisystem sofort als `rw` remounten;
- `fsck` auf einem schreibend eingehängten Dateisystem ausführen;
- LVM-Volume vergrößern, aber das Dateisystem nicht berücksichtigen;
- RAID-Rebuild mit Backup gleichsetzen;
- Paketmanager-Lockdatei ungeprüft löschen;
- Drittanbieter-Repository und Versionsabhängigkeiten nicht berücksichtigen;
- Cronjob mit der interaktiven Benutzerumgebung testen;
- Timerstatus prüfen, aber die zugehörige Service-Unit nicht;
- Systemzeit ungeprüft sprunghaft ändern;
- private Schlüssel in Protokolle oder Tickets kopieren;
- Kernelmodule ungeprüft entladen;
- mehrere Änderungen gleichzeitig durchführen;
- nur den Dienststatus, aber nicht die ursprüngliche Benutzerfunktion nachtesten.

---

66. Checkliste

- [ ] Hostname und Betriebssystem wurden dokumentiert.
- [ ] Kernelversion und Architektur wurden erfasst.
- [ ] Zeitpunkt und Zeitzone wurden geprüft.
- [ ] vollständiger Fehlertext wurde dokumentiert.

- [ ] Umfang der Störung wurde bestimmt.
- [ ] letzte funktionierende Nutzung wurde erfasst.
- [ ] letzte Änderungen wurden geprüft.
- [ ] systemd-Gesamtzustand wurde geprüft.
- [ ] Unit-Status und Exit-Code wurden geprüft.
- [ ] Dienstjournal wurde ausgewertet.
- [ ] vorheriger Boot wurde bei Bedarf berücksichtigt.
- [ ] tatsächlich verwendete Unit-Datei wurde geprüft.
- [ ] Drop-ins wurden berücksichtigt.
- [ ] Abhängigkeiten und Startreihenfolge wurden geprüft.
- [ ] Anwendungsidentität wurde bestimmt.
- [ ] Prozess und Prozesszustand wurden geprüft.
- [ ] wiederholte Prozessneustarts wurden berücksichtigt.
- [ ] offene Dateien wurden bei Bedarf geprüft.
- [ ] tatsächlich verwendete Konfiguration wurde bestimmt.
- [ ] Konfigurationssyntax wurde geprüft.
- [ ] Eigentümer, Gruppe und Modus wurden geprüft.
- [ ] alle Bestandteile des Pfades wurden geprüft.
- [ ] ACL wurde berücksichtigt.
- [ ] Zugriff wurde als Dienstbenutzer getestet.
- [ ] SELinux wurde geprüft.
- [ ] AppArmor wurde geprüft.
- [ ] Listener, Port und Bind-Adresse wurden geprüft.
- [ ] Portkonflikt wurde ausgeschlossen.
- [ ] lokale Anwendungsfunktion wurde getestet.
- [ ] Remotezugriff wurde getestet.
- [ ] IP-Adressen und Routen wurden geprüft.
- [ ] DNS und NSS wurden berücksichtigt.
- [ ] Firewallregeln wurden geprüft.
- [ ] IPv4 und IPv6 wurden unterschieden.
- [ ] CPU und Load wurden geprüft.
- [ ] RAM und Swap wurden geprüft.
- [ ] OOM-Ereignisse wurden geprüft.
- [ ] Speicherplatz wurde geprüft.

- [ ] Inodes wurden geprüft.
- [ ] gelöschte offene Dateien wurden berücksichtigt.
- [ ] Mounts wurden geprüft.
- [ ] `/etc/fstab` wurde bei Mountproblemen geprüft.
- [ ] read-only-Zustand wurde berücksichtigt.
- [ ] Kernel- und I/O-Fehler wurden geprüft.
- [ ] LVM oder RAID wurde bei Bedarf geprüft.
- [ ] Netzwerk-Mounts wurden berücksichtigt.
- [ ] Paketstatus und letzte Updates wurden geprüft.
- [ ] Cronjob oder systemd-Timer wurde in seiner tatsächlichen Umgebung geprüft.
- [ ] Zeit- und Zertifikatsprobleme wurden berücksichtigt.
- [ ] Boot- und Kernelmeldungen wurden geprüft.
- [ ] genau eine kontrollierte Maßnahme wurde durchgeführt.
- [ ] Rückweg wurde vor der Änderung festgelegt.
- [ ] Dienst, Port und Benutzerfunktion wurden nachgetestet.
- [ ] temporäre Diagnoseänderungen wurden zurückgenommen.
- [ ] Ursache, Maßnahme und Ergebnis wurden dokumentiert.

---

67. Schnellreferenz

Beobachtung	Nächstes Werkzeug
Betriebssystemversion unklar	`cat /etc/os-release`
Kernelversion unklar	`uname -r`
Systemzustand unklar	`systemctl is-system-running`
fehlgeschlagene Units gesucht	`systemctl --failed`
Dienststatus unklar	`systemctl status`
Exit-Code unklar	`systemctl show`
Dienstprotokoll gesucht	`journalctl -u`
vorheriger Boot relevant	`journalctl -b -1`
tatsächliche Unit unklar	`systemctl cat`
Drop-ins vermutet	`systemctl show -p DropInPaths`
Unit-Syntax unklar	`systemd-analyze verify`
Prozess gesucht	`pgrep -a`, `ps`
Prozessbaum benötigt	`pstree -ap`

Beobachtung	Nächstes Werkzeug
offene Datei unklar	`lsof`, `fuser`
gelöschte Datei belegt Speicher	`lsof +L1`
Dienstbenutzer unklar	`systemctl show -p User -p Group`
Benutzergruppen unklar	`id`, `getent`
Pfadberechtigung unklar	`namei -l`, `stat`
ACL unklar	`getfacl`
SELinux-Status unklar	`getenforce`, `sestatus`
SELinux-Ablehnung vermutet	`ausearch`
AppArmor-Ablehnung vermutet	`aa-status`, Kerneljournal
Listener unklar	`ss -ltnp`
Portkonflikt vermutet	`ss`, `lsof`
lokale HTTP-Funktion unklar	`curl -v`
TLS-Fehler	`openssl s_client`
IP-Konfiguration unklar	`ip -br address`
Route unklar	`ip route get`
DNS-Auflösung unklar	`getent hosts`, `dig`
Firewallzustand unklar	`firewall-cmd`, `nft`
Paketfluss unklar	`tcpdump`
CPU-Last unklar	`top`, `mpstat`, `pidstat`
RAM unklar	`free -h`, `vmstat`
OOM vermutet	`journalctl -k`
Swap unklar	`swapon --show`, `vmstat`
Speicherplatz unklar	`df -hT`
Verzeichnisgröße unklar	`du -xhd1`
Inodes unklar	`df -ih`
Mount unklar	`findmnt`
Blockgerät unklar	`lsblk -f`
fstab fehlerhaft vermutet	`findmnt --verify`
I/O-Fehler vermutet	Kerneljournal, `iostat`
LVM-Zustand unklar	`pvs`, `vgs`, `lvs`
Software-RAID unklar	`/proc/mdstat`, `mdadm --detail`

Beobachtung	Nächstes Werkzeug
Paketdatei verändert	`rpm -V`, `debsums`
Cronjob unklar	Crontab und Cronjournal
systemd-Timer unklar	`systemctl list-timers --all`
Zeitsynchronisation unklar	`timedatectl`, `chronyc`
Boot langsam	`systemd-analyze critical-chain`
Systemaufruf scheitert	`strace`

Merksatz

> Auf einem Linux-Server wird vom objektiven Symptom über Unit, Prozess, Protokoll, Identität, Port und Abhängigkeiten bis zu Berechtigungen, Sicherheitsrichtlinien, Ressourcen, Mounts und Kernel geprüft. Ein Dienst wird erst verändert oder neu gestartet, wenn der Fehlerzustand gesichert, die Auswirkung bewertet und ein Rückweg festgelegt wurde. `active`, ein offener Port und ein erfolgreicher lokaler Test sind jeweils nur Teilnachweise – entscheidend ist die ursprüngliche Benutzerfunktion.

Quellen und weiterführende Dokumentation

Offizielle systemd-Dokumentation

- systemd – systemctl
- systemd – journalctl
- systemd – systemd.service
- systemd – systemd.unit
- systemd – systemd.timer
- systemd – systemd-analyze
- systemd – systemd.resource-control

Offizielle Linux-Kernel-Dokumentation

- Linux Kernel – Administration Guide
- Linux Kernel – Magic SysRq
- Linux Kernel – Kernel Parameters
- Linux Kernel – Out Of Memory Handling

- Linux Kernel – Software RAID

Offizielle Red-Hat-Dokumentation

- Red Hat Enterprise Linux 9 – Systemadministration und Monitoring
- Red Hat Enterprise Linux 9 – SELinux verwenden
- Red Hat – SELinux-Probleme untersuchen
- Red Hat Enterprise Linux 9 – Dateisysteme verwalten
- Red Hat Enterprise Linux 9 – Systemstatus und Leistung überwachen
- Red Hat Enterprise Linux 9 – Logische Volumes verwalten

Offizielle Ubuntu-Dokumentation

- Ubuntu Server – How to manage systemd services
- Ubuntu Server – Security
- Ubuntu Server – Networking

Linux-Handbuchseiten

- Linux man-pages – Übersicht
- man7 – proc(5)
- man7 – signal(7)
- man7 – capabilities(7)
