12. Sicherheit, Backup und Automatisierung

12.1 Sicherheit, Backup und Automatisierung

Linux-Systeme werden häufig als Server, Container-Hosts, Datenbanksysteme, Webserver, NAS-Systeme oder Administrationssysteme eingesetzt.

Deshalb sind Sicherheit, Backups und Automatisierung besonders wichtig.

Ein Linux-System soll nicht nur funktionieren, sondern auch:

```
- sicher betrieben werden
- nachvollziehbar administriert werden
- regelmäßig gesichert werden
- im Fehlerfall wiederherstellbar sein
- wiederkehrende Aufgaben automatisch erledigen
- unnötige Risiken vermeiden
```

Merksatz:

```
Administration bedeutet nicht nur einrichten,
sondern sicher betreiben,
sichern
und überwachen.
```

Lernziele

Nach dieser Seite solltest du erklären können:

```
- welche Grundprinzipien Linux-Sicherheit hat
- warum minimale Rechte wichtig sind
- warum Updates sicherheitsrelevant sind
- warum Backups und Restore-Tests zusammengehören
- was RPO und RTO bedeuten
- welche Backup-Arten es gibt
- wie Automatisierung mit cron,
  systemd Timern
  und Skripten funktioniert
```

- warum Automatisierung dokumentiert und überwacht werden muss
- welche typischen Sicherheits-
  und Backupfehler auftreten

Grundidee

Ein sicher betriebenes Linux-System braucht mehrere Schutzebenen.

Dazu gehören:

- Benutzer und Rechte
- sichere SSH-Konfiguration
- regelmäßige Updates
- Firewall-Regeln
- Diensthärtung
- Logauswertung
- Monitoring
- Backups
- Restore-Tests
- Automatisierung
- Dokumentation

Wichtig:

Eine einzelne Maßnahme reicht nicht aus.

Merksatz:

Sicherheit besteht aus mehreren Schichten.

Schutzziele der IT-Sicherheit

Die klassischen Schutzziele sind:

Schutzziel	Bedeutung
Vertraulichkeit	Daten dürfen nur von Berechtigten gelesen werden
Integrität	Daten dürfen nicht unbemerkt verändert werden
Verfügbarkeit	Systeme und Daten müssen bei Bedarf nutzbar sein

Beispiele:

Beispiel	Schutzziel
SSH-Schlüssel schützen	Vertraulichkeit
Prüfsummen nutzen	Integrität
Backups und Redundanz	Verfügbarkeit

Merksatz:

```
Vertraulichkeit,
Integrität
und Verfügbarkeit sind zentrale Schutzziele.
```

Prinzip der minimalen Rechte

Ein Benutzer, Dienst oder Prozess sollte nur die Rechte haben, die er wirklich benötigt.

Dieses Prinzip nennt man:

```
Least Privilege
```

Beispiele:

```
Ein Webserver braucht keine Root-Rechte,
wenn er nur Webseiten ausliefern soll.

Ein Backup-Benutzer braucht nicht automatisch Schreibrechte auf alle Produktivdaten.

Ein normaler Benutzer sollte keine systemweiten Konfigurationen ändern dürfen.
```

Vorteil:

```
Fehler oder Angriffe richten weniger Schaden an.
```

Merksatz:

```
So wenig Rechte wie möglich,
so viele wie nötig.
```

Root-Rechte bewusst nutzen

Root kann fast alles auf dem System verändern.

Deshalb:

```
- nicht dauerhaft als root arbeiten
- sudo gezielt verwenden
- Befehle vor Ausführung prüfen
- Wildcards mit Vorsicht verwenden
- keine unbekannten Skripte als root ausführen
- administrative Aktionen dokumentieren
```

Beispiel:

```
sudo systemctl restart nginx
```

ist besser nachvollziehbar als dauerhaft als root zu arbeiten.

Merksatz:

```
Root-Rechte sind Werkzeug,
keine Arbeitsumgebung.
```

Dienstkonten

Dienste sollten unter eigenen Dienstkonten laufen.

Beispiele:

Dienstkonto	Typischer Zweck
www-data	Webserver
nginx	Nginx-Webserver
postgres	PostgreSQL
mysql	MySQL/MariaDB
backup	Backup-Aufgaben

Vorteil:

```
Wenn ein Dienst kompromittiert wird,
hat der Angreifer nur die Rechte dieses Dienstkontos.
```

Merksatz:

```
Dienste sollten nicht unnötig als root laufen.
```

SSH absichern

SSH ist ein häufiger Administrationszugang.

Wichtige Schutzmaßnahmen:

```
- sichere Passwörter oder SSH-Schlüssel nutzen
- direkten Root-Login vermeiden
- nur benötigte Benutzer zulassen
- Firewall-Regeln setzen
- Logs prüfen
- fehlgeschlagene Logins überwachen
- private Schlüssel schützen
- bekannte Hosts prüfen
```

Wichtige Datei:

```
/etc/ssh/sshd_config
```

Merksatz:

```
SSH ist mächtig und muss besonders geschützt werden.
```

SSH-Schlüssel schützen

Ein SSH-Schlüsselpaar besteht aus:

```
privatem Schlüssel

öffentlichem Schlüssel
```

Der öffentliche Schlüssel darf auf Servern liegen.

Der private Schlüssel bleibt geheim.

Wichtig:

```
privaten Schlüssel nicht weitergeben

Passphrase verwenden

Dateirechte passend setzen

Schlüssel bei Verlust oder Verdacht entfernen

authorized_keys regelmäßig prüfen
```

Merksatz:

```
Der private SSH-Schlüssel ist wie ein Passwort
und muss geschützt werden.
```

Firewall-Grundlagen

Eine Firewall begrenzt, welcher Netzwerkverkehr erlaubt ist.

Eine gute Firewall-Regel beschreibt:

```
- Quelle
- Ziel
- Protokoll
- Port
- Richtung
- Aktion
- Zweck
```

Beispiel:

Feld	Beispiel
Quelle	Adminnetz

Feld	Beispiel
Ziel	Linux-Server
Protokoll	TCP
Port	22
Aktion	erlauben
Zweck	SSH-Administration

Merksatz:

```
Port allein ist keine vollständige Firewall-Regel.
```

Nur benötigte Dienste betreiben

Jeder laufende Dienst kann eine Angriffsfläche sein.

Deshalb prüfen:

```
Welche Dienste laufen?

Welche Ports sind offen?

Welche Dienste werden wirklich benötigt?

Sind Dienste aktuell?

Sind Dienste sicher konfiguriert?
```

Prüfen:

```
systemctl --type=service

ss -tulpen
```

Merksatz:

```
Nicht benötigte Dienste deaktivieren oder entfernen.
```

Updates und Patchmanagement

Sicherheitsupdates schließen bekannte Schwachstellen.

Patchmanagement bedeutet:

```
Updates geplant,
kontrolliert
und nachvollziehbar einspielen.
```

Dazu gehören:

```
- verfügbare Updates prüfen
- Sicherheitsrelevanz bewerten
- Backup prüfen
- Wartungsfenster planen
- Updates einspielen
- Dienste prüfen
- Logs prüfen
- Dokumentation aktualisieren
```

Merksatz:

```
Updates sind Teil der Sicherheit.
```

Warum Updates nicht blind einspielen?

Updates sind wichtig, können aber auch Probleme verursachen.

Mögliche Risiken:

```
Dienst startet nicht mehr

Konfiguration ist nicht kompatibel

Abhängigkeiten ändern sich

Kernel-Update braucht Neustart
```

Anwendung verhält sich anders

Drittanbieter-Paket verursacht Konflikt

Deshalb:

vorher sichern

Änderungen dokumentieren

Dienste nachher prüfen

Rollback-Möglichkeit einplanen

Merksatz:

Sicherheitsupdates sind wichtig,
aber produktive Systeme brauchen kontrollierte Änderung.

Dateirechte und Sicherheit

Falsche Dateirechte sind ein häufiger Sicherheitsfehler.

Beispiele:

Problem	Risiko
777 auf Verzeichnissen	jeder darf schreiben
private Schlüssel zu offen	SSH verweigert Zugriff oder Schlüssel kann gelesen werden
Konfigurationsdateien für alle lesbar	Geheimnisse können sichtbar sein
Dienstdateien falscher Besitzer	Dienst startet nicht oder Sicherheitsrisiko
Backups für alle lesbar	Datenabfluss

Merksatz:

Rechteprobleme sind Sicherheits-
und Betriebsprobleme.

Sensible Dateien

Sensible Dateien müssen besonders geschützt werden.

Beispiele:

```
SSH-Schlüssel

Datenbank-Passwörter

API-Tokens

Zertifikatsschlüssel

Backup-Dateien

Konfigurationsdateien mit Zugangsdaten

Passwort-Hashes
```

Beispiele für Pfade:

```
~/.ssh/id_rsa

~/.ssh/authorized_keys

/etc/shadow

/etc/ssh/sshd_config

/etc/sudoers
```

Merksatz:

```
Geheimnisse gehören nicht in offen lesbare Dateien.
```

---

sudo sicher nutzen

sudo sollte gezielt vergeben werden.

Wichtige Punkte:

```
- nicht jeder Benutzer braucht sudo
- sudo-Regeln dokumentieren
- Gruppenmitgliedschaften prüfen
- keine unnötigen NOPASSWD-Regeln
- administrative Aktionen nachvollziehbar halten
```

Prüfen:

```
id benutzer

groups benutzer

sudo -l
```

Merksatz:

```
sudo ist kontrollierte Rechteerweiterung.
```

Logs und Sicherheitsereignisse

Sicherheitsrelevante Ereignisse findet man häufig in Authentifizierungs- und Dienstlogs.

Beispiele:

```
fehlgeschlagene Logins

erfolgreiche SSH-Logins

sudo-Nutzung

neue Benutzer

Dienststarts

Firewall-Blockierungen
```

```
ungewöhnliche Prozesse
```

Beispiele:

```
journalctl -u ssh

grep -i "failed" /var/log/auth.log

grep -i "sudo" /var/log/auth.log
```

Merksatz:

```
Sicherheit ohne Logprüfung ist blind.
```

Backups

Ein Backup ist eine Sicherung von Daten, Konfigurationen oder Systemzuständen.

Backups schützen vor:

```
- versehentlichem Löschen
- Hardwaredefekt
- Dateisystemfehler
- Ransomware
- fehlgeschlagenem Update
- Fehlkonfiguration
- Diebstahl
- Brand oder Wasserschaden
- Benutzerfehler
```

Merksatz:

```
Kein Backup bedeutet:
keine sichere Wiederherstellung.
```

Backup ist nicht gleich Restore

Ein Backup ist nur dann wertvoll, wenn es auch wiederhergestellt werden kann.

Deshalb wichtig:

- Restore testen
- Wiederherstellungszeit messen
- Datenintegrität prüfen
- Dokumentation erstellen
- Zuständigkeiten klären
- Notfallzugriff prüfen

Merksatz:

> Ein ungetestetes Backup ist nur eine Hoffnung.

Restore

Restore bedeutet:

> Wiederherstellung aus einem Backup.

Beispiele:

- einzelne Datei wiederherstellen
- Datenbank zurückspielen
- VM wiederherstellen
- Serverkonfiguration wiederherstellen
- komplettes System neu aufbauen

Wichtig:

Restore muss geübt und dokumentiert sein.

Merksatz:

Backup ist die Sicherung.
Restore ist die Wiederherstellung.

RPO

RPO steht für:

Recovery Point Objective

Es beschreibt, wie viel Datenverlust maximal akzeptabel ist.

Beispiel:

RPO 24 Stunden

bedeutet:

Im schlimmsten Fall dürfen Daten seit dem letzten täglichen Backup verloren gehen.

Beispiel:

RPO 1 Stunde

bedeutet:

Maximal eine Stunde Datenverlust ist akzeptabel.

Merksatz:

RPO beantwortet:
Wie viele Daten dürfen maximal verloren gehen?

RTO

RTO steht für:

```
Recovery Time Objective
```

Es beschreibt, wie lange die Wiederherstellung maximal dauern darf.

Beispiel:

```
RTO 4 Stunden
```

bedeutet:

```
Der Dienst soll innerhalb von 4 Stunden wieder laufen.
```

Merksatz:

```
RTO beantwortet:
Wie schnell muss es wieder funktionieren?
```

RPO und RTO unterscheiden

Begriff	Frage	Beispiel
RPO	Wie viele Daten dürfen verloren gehen?	maximal 1 Stunde
RTO	Wie lange darf die Wiederherstellung dauern?	maximal 4 Stunden

Merksatz:

```
RPO = Datenverlust.
RTO = Wiederherstellungszeit.
```

Backup-Arten

Backup-Art	Bedeutung
Vollbackup	vollständige Sicherung

Backup-Art	Bedeutung
inkrementelles Backup	sichert Änderungen seit letztem Backup
differentielles Backup	sichert Änderungen seit letztem Vollbackup
Snapshot	Zustand zu einem Zeitpunkt
Image-Backup	vollständiges Abbild eines Systems
Datei-Backup	Sicherung einzelner Dateien und Ordner
Datenbankdump	logische Sicherung einer Datenbank

Merksatz:

> Backup-Art muss zum Wiederherstellungsziel passen.

Vollbackup

Ein Vollbackup sichert alle ausgewählten Daten vollständig.

Vorteile:

> einfach zu verstehen
>
> Wiederherstellung oft einfacher
>
> vollständiger Stand

Nachteile:

> benötigt viel Speicher
>
> dauert länger
>
> erzeugt mehr Last

Merksatz:

> Vollbackup ist einfach,
> aber speicherintensiv.

Inkrementelles Backup

Ein inkrementelles Backup sichert nur Änderungen seit dem letzten Backup.

Vorteile:

spart Speicher

schneller als Vollbackup

Nachteile:

Wiederherstellung kann mehrere Sicherungen benötigen

Kette muss vollständig sein

Merksatz:

Inkrementell spart Speicher,
macht Restore aber abhängiger von der Backup-Kette.

Differentielles Backup

Ein differentielles Backup sichert Änderungen seit dem letzten Vollbackup.

Vorteile:

Restore einfacher als bei langer inkrementeller Kette

weniger Speicher als tägliches Vollbackup

Nachteile:

wächst bis zum nächsten Vollbackup

Merksatz:

Differentiell bezieht sich auf das letzte Vollbackup.

Snapshot

Ein Snapshot ist ein Zustand zu einem bestimmten Zeitpunkt.

Snapshots sind nützlich vor:

```
Updates

Konfigurationsänderungen

Tests

größeren Wartungen
```

Wichtig:

```
Snapshots ersetzen nicht immer ein echtes externes Backup.
```

Risiko:

```
Wenn das gesamte Storage-System ausfällt,
kann auch der Snapshot verloren sein.
```

Merksatz:

```
Snapshot ist praktisch,
aber nicht automatisch ein vollständiges Backup-Konzept.
```

3-2-1-Regel

Eine bekannte Backup-Regel ist:

```
3 Kopien der Daten

2 unterschiedliche Medientypen oder Speicherorte

1 Kopie extern oder offline
```

Ziel:

Schutz vor Hardwareausfall,
Bedienfehlern,
Ransomware
und Standortschäden.

Merksatz:

3-2-1 reduziert das Risiko,
dass alle Kopien gleichzeitig verloren gehen.

Offline-Backup

Ein Offline-Backup ist nach der Sicherung nicht dauerhaft erreichbar.

Vorteile:

Schutz vor Ransomware

Schutz vor versehentlichem Löschen

Schutz vor kompromittierten Zugangsdaten

Beispiele:

externe Festplatte getrennt lagern

offline geschalteter Backup-Speicher

unveränderbare Sicherung

Merksatz:

Was dauerhaft verbunden ist,
kann auch dauerhaft angegriffen oder gelöscht werden.

Backup-Verschlüsselung

Backups enthalten oft sensible Daten.

Deshalb sollten sie geschützt werden durch:

- Verschlüsselung
- Zugriffskontrolle
- sichere Aufbewahrung
- getrennte Zugangsdaten
- Protokollierung
- regelmäßige Prüfung

Wichtig:

Schlüssel und Passwörter müssen sicher aufbewahrt werden.
Ohne Schlüssel ist ein verschlüsseltes Backup nicht wiederherstellbar.

Merksatz:

Verschlüsseltes Backup schützt Daten,
aber der Schlüssel muss verfügbar bleiben.

Backup-Integrität

Backups müssen vollständig und unverändert sein.

Mögliche Prüfungen:

Prüfsummen

Test-Restore

Backup-Logs

Vergleich von Dateianzahlen

Stichproben

automatische Prüfberichte

Merksatz:

```
Backup erfolgreich gemeldet heißt noch nicht:
Restore funktioniert.
```

Backup von Konfigurationen

Neben Daten sollten auch Konfigurationen gesichert werden.

Beispiele:

```
/etc

systemd-Units

Firewall-Regeln

Webserver-Konfiguration

Datenbank-Konfiguration

SSH-Konfiguration

Cronjobs

Skripte

Docker Compose Dateien

Dokumentation
```

Warum?

```
Ohne Konfiguration dauert Wiederherstellung deutlich länger.
```

Merksatz:

```
Daten ohne Konfiguration reichen oft nicht für schnellen Restore.
```

Backup von Datenbanken

Datenbanken sollte man nicht einfach im laufenden Betrieb als Dateien kopieren, wenn dadurch inkonsistente Daten entstehen können.

Sicherer sind oft:

```
Datenbankdump

datenbankspezifische Backup-Werkzeuge

konsistente Snapshots

Replikation plus Backup
```

Beispiele:

```
PostgreSQL:
pg_dump

MySQL/MariaDB:
mysqldump
```

Merksatz:

```
Datenbanken brauchen konsistente Backups.
```

Automatisierung

Automatisierung bedeutet, wiederkehrende Aufgaben automatisch auszuführen.

Beispiele:

```
Backups

Logrotation

Updates prüfen
```

```
Speicherplatz prüfen

Dienste überwachen

Reports erstellen

Dateien aufräumen

Zertifikatsprüfung

Synchronisation
```

Merksatz:

```
Automatisierung spart Zeit,
muss aber überwacht werden.
```

---

cron

cron ist ein klassischer Zeitplaner unter Linux.

Cronjobs können Befehle oder Skripte regelmäßig ausführen.

Crontab bearbeiten:

```
crontab -e
```

Crontab anzeigen:

```
crontab -l
```

Beispiel:

```
0 2 * * * /home/felix/backup.sh
```

Bedeutung:

```
Jeden Tag um 02:00 Uhr backup.sh ausführen.
```

Merksatz:

```
cron startet Aufgaben nach Zeitplan.
```

Crontab-Zeitfelder

Ein Cronjob hat fünf Zeitfelder.

```
Minute Stunde Tag-des-Monats Monat Wochentag Befehl
```

Beispiel:

```
30 3 * * 1 /home/felix/report.sh
```

Bedeutung:

```
Jeden Montag um 03:30 Uhr.
```

Feld	Bedeutung
Minute	0 bis 59
Stunde	0 bis 23
Tag des Monats	1 bis 31
Monat	1 bis 12
Wochentag	0 bis 7, häufig 0 und 7 = Sonntag

Merksatz:

```
Cron hat fünf Zeitfelder vor dem Befehl.
```

Typische Cron-Beispiele

Cron-Eintrag	Bedeutung
0 2 * * *	täglich um 02:00 Uhr
*/15 * * * *	alle 15 Minuten
0 * * * *	jede volle Stunde

Cron-Eintrag	Bedeutung
30 3 * * 1	montags um 03:30 Uhr
0 1 1 * *	am 1. jedes Monats um 01:00 Uhr

Merksatz:

```
Stern bedeutet:
jeder mögliche Wert.
```

---

cron und Umgebung

Cron hat oft eine andere Umgebung als die normale Shell.

Typische Probleme:

```
PATH ist anders

Variablen fehlen

Arbeitsverzeichnis ist anders

Ausgaben gehen verloren

Berechtigungen unterscheiden sich
```

Deshalb in Cron-Skripten:

```
absolute Pfade verwenden

PATH bewusst setzen

Logs schreiben

Fehlerausgabe umleiten

Skripte ausführbar machen
```

Merksatz:

```
Ein Skript,
das manuell funktioniert,
funktioniert nicht automatisch auch in cron.
```

---

cron mit Logging

Beispiel:

```
0 2 * * * /home/felix/backup.sh >> /var/log/backup.log 2>&1
```

Bedeutung:

```
Standardausgabe und Fehlerausgabe werden in backup.log geschrieben.
```

Merksatz:

```
Automatisierung braucht Logausgaben.
```

---

systemd Timer

systemd Timer sind eine moderne Alternative zu cron.

Sie bestehen meist aus:

```
.service

.timer
```

Beispielhafte Einsatzbereiche:

```
regelmäßige Backups

Prüfskripte

Aufräumaufgaben

Monitoring-Jobs
```

Vorteile:

```
Integration in systemd

Logs über journalctl

Abhängigkeiten möglich

Status über systemctl
```

Merksatz:

```
systemd Timer sind zeitgesteuerte systemd-Units.
```

---

cron und systemd Timer vergleichen

Merkmal	cron	systemd Timer
Klassiker	ja	moderner
einfache Zeitpläne	gut	gut
systemd-Integration	gering	hoch
Logs	oft manuell	journalctl
Abhängigkeiten	begrenzt	besser
Statusprüfung	weniger einheitlich	systemctl

Merksatz:

```
cron ist einfach.
systemd Timer sind stärker in systemd integriert.
```

Automatisierung überwachen

Automatisierte Aufgaben können fehlschlagen.

Mögliche Ursachen:

```
Ziel nicht erreichbar
```

Speicher voll

Rechteproblem

Passwort oder Schlüssel ungültig

Netzwerkproblem

Skriptfehler

Dienst nicht verfügbar

Pfad geändert

Deshalb wichtig:

Logs

Exit-Codes

Benachrichtigung bei Fehler

regelmäßige Kontrolle

Monitoring

Merksatz:

Eine Automatisierung ohne Kontrolle kann unbemerkt ausfallen.

Exit-Codes in Automatisierung

Exit-Codes zeigen, ob ein Befehl erfolgreich war.

Exit-Code	Bedeutung
0	Erfolg
ungleich 0	Fehler oder besonderer Zustand

Wichtig für:

```
cron

systemd Timer

Monitoring

Skripte

CI/CD
```

Merksatz:

```
Automatisierung muss Fehler über Exit-Codes erkennbar machen.
```

Dokumentation

Sicherheits- Backup- und Automatisierungsmaßnahmen müssen dokumentiert werden.

Dokumentieren:

```
- Zweck
- Pfade
- Zeitpläne
- Benutzer
- Rechte
- Abhängigkeiten
- Restore-Schritte
- Verantwortliche
- Prüftermine
- bekannte Risiken
- Änderungen
```

Merksatz:

```
Was nicht dokumentiert ist,
ist im Notfall schwer wiederherzustellen.
```

Typische Sicherheitsfehler

Fehler	Risiko
direkter Root-Login per SSH	leichteres Angriffsziel
schwache Passwörter	unberechtigter Zugriff
777-Rechte	jeder darf alles
keine Updates	bekannte Schwachstellen bleiben offen
unnötige Dienste	größere Angriffsfläche
private Schlüssel ungeschützt	Identitätsdiebstahl
Geheimnisse in Skripten	Zugangsdaten werden sichtbar
keine Logs	Vorfälle schwer nachvollziehbar
keine Firewall	unnötige Dienste erreichbar
sudo für alle	Rechteausweitung

Typische Backupfehler

Fehler	Risiko
kein Restore-Test	Backup eventuell unbrauchbar
Backup liegt nur lokal	Verlust bei Hardwaredefekt
Backup dauerhaft beschreibbar	Ransomware kann es verschlüsseln
Datenbankdateien inkonsistent kopiert	Restore fehlerhaft
Backup nicht verschlüsselt	Datenabfluss
Backup-Logs nicht geprüft	Fehler bleiben unbemerkt
Konfiguration nicht gesichert	Wiederherstellung dauert länger
RPO/RTO nicht definiert	unklare Anforderungen
Speicherziel voll	Backups schlagen fehl
alte Backups nie gelöscht	Speicher läuft voll

Typische Automatisierungsfehler

Fehler	Risiko
keine Logs	Fehler bleiben unsichtbar
keine Exit-Code-Prüfung	Fehler werden nicht erkannt
relative Pfade in cron	Skript findet Dateien nicht
PATH nicht gesetzt	Befehl wird nicht gefunden
keine Benachrichtigung	Ausfall bleibt unbemerkt

Fehler	Risiko
Skript nicht getestet	Fehler wird automatisiert
Geheimnisse im Klartext	Sicherheitsrisiko
keine Dokumentation	Wartung schwierig
keine Rechteprüfung	Skript scheitert oder ist unsicher
keine Sperre gegen parallele Läufe	Datenkonflikte

Sichere Arbeitsweise

Bei Sicherheit:

```
minimale Rechte

Dienste prüfen

Updates planen

SSH absichern

Firewall setzen

Logs kontrollieren
```

Bei Backups:

```
Backupziel prüfen

Backup automatisieren

Restore testen

RPO/RTO definieren

Backup verschlüsseln

externe Kopie vorhalten
```

Bei Automatisierung:

Skripte testen

Logs schreiben

Exit-Codes nutzen

Monitoring einrichten

Dokumentation pflegen

Merksatz:

Sicherer Betrieb entsteht durch regelmäßige Kontrolle,
nicht durch einmalige Einrichtung.

Typische Prüfungsfragen

Frage	Kurzantwort
Was bedeutet Least Privilege?	nur notwendige Rechte vergeben
Warum sollte SSH abgesichert werden?	zentraler Administrationszugang
Warum sind Updates wichtig?	schließen bekannte Schwachstellen
Was ist ein Backup?	Sicherung von Daten oder Systemzuständen
Was ist Restore?	Wiederherstellung aus Backup
Was bedeutet RPO?	maximal akzeptabler Datenverlust
Was bedeutet RTO?	maximal akzeptable Wiederherstellungszeit
Was ist ein Vollbackup?	vollständige Sicherung
Was ist ein inkrementelles Backup?	Änderungen seit letztem Backup
Was ist ein differentielles Backup?	Änderungen seit letztem Vollbackup
Was ist ein Snapshot?	Zustand zu einem Zeitpunkt
Was macht cron?	zeitgesteuerte Aufgaben ausführen
Was ist ein systemd Timer?	zeitgesteuerte systemd-Unit
Warum Restore-Test?	prüft, ob Backup wirklich nutzbar ist

Typische Prüfungsfallen

Falle	Richtig denken
Backup mit Restore gleichsetzen	Backup sichern, Restore wiederherstellen
Backup nie testen	ungetestetes Backup ist unsicher
RPO und RTO verwechseln	RPO Datenverlust, RTO Zeit
Snapshot als vollständiges Backup sehen	nicht automatisch extern oder unabhängig
cron ohne Logging	Fehler bleiben unsichtbar
manuelles Skript läuft, cron nicht	andere Umgebung beachten
777 als schnelle Lösung	Sicherheitsrisiko
Root dauerhaft nutzen	unnötig gefährlich
Updates ohne Prüfung	produktive Dienste können betroffen sein
keine Dokumentation	Notfallwiederherstellung erschwert
Geheimnisse in Skripten speichern	Sicherheitsrisiko

IHK-sichere Kurzformulierung

Linux-Sicherheit basiert auf mehreren Maßnahmen wie minimalen Rechten, sicheren SSH-Zugängen, regelmäßigen Updates, Firewall-Regeln, Diensthärtung, Logging und Monitoring. Backups schützen vor Datenverlust, sind aber nur zuverlässig, wenn auch die Wiederherstellung getestet wurde. RPO beschreibt, wie viel Datenverlust maximal akzeptabel ist, während RTO beschreibt, wie lange die Wiederherstellung maximal dauern darf. Backup-Arten sind unter anderem Vollbackup, inkrementelles Backup, differentielles Backup, Snapshot, Image-Backup und Datenbankdump. Wiederkehrende Aufgaben können mit cron, systemd Timern und Skripten automatisiert werden. Automatisierung muss protokolliert, überwacht und dokumentiert werden, damit Fehler nicht unbemerkt bleiben.

Merksätze

Sicherheit besteht aus mehreren Schichten.

Vertraulichkeit,
Integrität
und Verfügbarkeit sind zentrale Schutzziele.

Least Privilege bedeutet:
nur notwendige Rechte.

Root-Rechte bewusst verwenden.

Dienste nicht unnötig als root betreiben.

SSH besonders absichern.

Privater SSH-Schlüssel bleibt geheim.

Firewall-Regeln brauchen Quelle,
Ziel,
Protokoll,
Port,
Aktion
und Zweck.

Nicht benötigte Dienste entfernen oder deaktivieren.

Updates schließen bekannte Schwachstellen.

Patchmanagement ist ein Prozess.

Backup schützt vor Datenverlust.

Restore ist die Wiederherstellung.

Ein ungetestetes Backup ist nur eine Hoffnung.

RPO = maximal akzeptabler Datenverlust.

RTO = maximal akzeptable Wiederherstellungszeit.

Vollbackup sichert alles.

Inkrementell sichert Änderungen seit letztem Backup.

Differentiell sichert Änderungen seit letztem Vollbackup.

Snapshot ist ein Zeitpunktzustand.

Snapshot ersetzt nicht automatisch ein externes Backup.

3-2-1-Regel reduziert Backup-Risiken.

Backup-Speicher besonders schützen.

Datenbanken brauchen konsistente Backups.

cron startet Aufgaben nach Zeitplan.

systemd Timer sind zeitgesteuerte systemd-Units.

Cron hat oft andere Umgebung als die normale Shell.

Automatisierung braucht Logs.

Exit-Codes zeigen Erfolg oder Fehler.

Automatisierung ohne Kontrolle kann unbemerkt ausfallen.

Dokumentation ist Teil des Betriebs.

Sicherer Betrieb ist regelmäßige Kontrolle,
nicht einmalige Einrichtung.