10.9 Authentifizierung, Autorisierung und Anwendungssicherheit

Authentifizierung und Autorisierung gehören zu den wichtigsten Sicherheitsgrundlagen auf der Anwendungsschicht.

Die Anwendungsschicht ist OSI-Schicht 7.

Hier arbeiten viele Dienste direkt mit Benutzern, Anmeldungen, Rollen, Rechten und Anwendungen.

Wichtige Begriffe sind:

- Authentifizierung
- Autorisierung
- Identität
- Benutzerkonto
- Passwort
- Token
- Session
- Rolle
- Rechte
- Multi-Faktor-Authentifizierung
- Single Sign-on
- Least Privilege

Merksatz:

Authentifizierung prüft,
wer jemand ist.

Autorisierung prüft,
was jemand darf.

Warum ist das wichtig?

Viele Netzwerkdienste müssen entscheiden:

```
Wer greift zu?
```

und:

```
Was darf diese Person oder dieses System tun?
```

Beispiele:

```
Benutzer meldet sich an einer Webanwendung an.
Administrator verbindet sich per SSH.
Mitarbeiter öffnet eine SMB-Freigabe.
Anwendung ruft eine API mit Token auf.
Benutzer meldet sich an einem E-Mail-Postfach an.
Client greift auf einen Verzeichnisdienst zu.
```

Ohne saubere Authentifizierung und Autorisierung können unberechtigte Zugriffe entstehen.

Merksatz:

```
Anmeldung allein reicht nicht.
Rechte müssen ebenfalls geprüft werden.
```

Authentifizierung

Authentifizierung bedeutet:

```
Die Identität wird überprüft.
```

Frage:

```
Wer bist du?
```

Beispiele:

```
- Benutzername und Passwort
- SSH-Schlüssel
- Zertifikat
```

- Smartcard
- Fingerabdruck
- Einmalcode
- Hardware-Token
- App-Bestätigung

Merksatz:

Authentifizierung = Identität prüfen.

Autorisierung

Autorisierung bedeutet:

Es wird geprüft,
welche Rechte eine authentifizierte Identität hat.

Frage:

Was darfst du?

Beispiele:

- Datei lesen
- Datei ändern
- Benutzer anlegen
- Server verwalten
- Adminbereich öffnen
- API-Daten abrufen
- Bestellung freigeben
- Konfiguration ändern

Merksatz:

Autorisierung = Rechte prüfen.

Authentifizierung und Autorisierung unterscheiden

Begriff	Frage	Beispiel
Authentifizierung	Wer bist du?	Benutzer meldet sich an
Autorisierung	Was darfst du?	Benutzer darf nur bestimmte Ordner öffnen

Beispiel:

```
Felix meldet sich erfolgreich an.
```

Das ist:

```
Authentifizierung
```

Danach prüft das System:

```
Darf Felix diese Datei löschen?
```

Das ist:

```
Autorisierung
```

Merksatz:

```
Erst Identität,
dann Rechte.
```

Typischer Fehler

Ein häufiger Denkfehler ist:

```
Wenn ein Benutzer angemeldet ist,
darf er automatisch alles.
```

Das ist falsch.

Ein Benutzer kann korrekt angemeldet sein, aber trotzdem keine Berechtigung für eine bestimmte Aktion haben.

Beispiel:

```
Benutzer ist angemeldet,
darf aber keine Adminseite öffnen.
```

Merksatz:

```
Angemeldet bedeutet nicht automatisch berechtigt.
```

Identität

Eine Identität beschreibt, wer eine Person, ein Dienst oder ein System ist.

Beispiele:

```
- Benutzerkonto
- Dienstkonto
- Computerobjekt
- API-Client
- Zertifikat
- SSH-Schlüssel
- Anwendungskonto
```

Identitäten müssen eindeutig verwaltet werden.

Merksatz:

```
Identität beschreibt,
wer oder was zugreift.
```

Benutzerkonto

Ein Benutzerkonto ist eine digitale Identität für eine Person.

Es enthält oft:

```
- Benutzername
- Kennwort-Hash
```

```
- Gruppenmitgliedschaften
- Profilinformationen
- Berechtigungen
- Status des Kontos
- Anmelderichtlinien
```

Benutzerkonten können lokal oder zentral verwaltet werden.

Beispiele:

```
- lokales Windows-Konto
- Active-Directory-Konto
- LDAP-Konto
- Cloud-Konto
- Anwendungskonto
```

Merksatz:

```
Benutzerkonto = digitale Identität eines Benutzers.
```

Dienstkonto

Ein Dienstkonto wird nicht direkt von einem normalen Benutzer verwendet, sondern von einem Dienst oder einer Anwendung.

Beispiele:

```
- Datenbankzugriff einer Webanwendung
- Backup-Dienst
- Monitoring-Dienst
- LDAP-Abfragekonto
- API-Integration
```

Wichtig:

```
Dienstkonten sollten nur die Rechte haben,
die sie wirklich benötigen.
```

Merksatz:

```
Dienstkonto = Konto für einen Dienst,
nicht für normale Benutzerarbeit.
```

Passwort

Ein Passwort ist ein geheimer Nachweis, mit dem sich ein Benutzer authentifizieren kann.

Sichere Passwörter sollten:

```
- ausreichend lang sein
- nicht leicht zu erraten sein
- nicht mehrfach verwendet werden
- nicht im Klartext gespeichert werden
- bei Verdacht geändert werden
```

Wichtig:

```
Länge ist oft wichtiger als komplizierte Sonderzeichen-Regeln.
```

Merksatz:

```
Passwort = geheimer Identitätsnachweis.
```

Passwort-Hash

Passwörter sollten nicht im Klartext gespeichert werden.

Stattdessen speichert man einen Passwort-Hash.

Ein Hash ist ein Einweg-Prüfwert.

Bei der Anmeldung wird das eingegebene Passwort erneut gehasht und mit dem gespeicherten Hash verglichen.

Merksatz:

```
Passwörter nicht im Klartext speichern,
sondern sicher hashen.
```

Salt

Ein Salt ist ein zusätzlicher zufälliger Wert beim Passwort-Hashing.

Ziel:

```
gleiche Passwörter sollen nicht automatisch gleiche Hashes erzeugen.
```

Beispiel:

```
Zwei Benutzer haben dasselbe Passwort.
```

Durch unterschiedliche Salts entstehen trotzdem unterschiedliche Hashes.

Merksatz:

```
Salt schützt gegen einfache Hash-Vergleiche und vorberechnete Tabellen.
```

Multi-Faktor-Authentifizierung

Multi-Faktor-Authentifizierung wird kurz genannt:

```
MFA
```

Dabei werden mehrere Faktoren kombiniert.

Typische Faktoren:

Faktor	Beispiel
Wissen	Passwort
Besitz	Smartphone, Token, Smartcard
Sein	Fingerabdruck, Gesichtserkennung

Beispiel:

```
Passwort
plus
Einmalcode aus App
```

Merksatz:

```
MFA kombiniert mehrere Nachweise.
```

Warum ist MFA wichtig?

Wenn ein Passwort gestohlen wird, kann ein Angreifer sich ohne zweiten Faktor nicht so einfach anmelden.

MFA schützt besonders bei:

```
- Cloud-Konten
- VPN-Zugängen
- Admin-Konten
- E-Mail-Konten
- Remotezugriff
- kritischen Anwendungen
```

Merksatz:

```
MFA reduziert das Risiko durch gestohlene Passwörter.
```

Einmalpasswort

Ein Einmalpasswort wird auch genannt:

```
OTP
```

OTP steht für:

```
One-Time Password
```

Es ist nur einmal oder nur kurzzeitig gültig.

Beispiele:

```
- Code aus Authenticator-App
- Hardware-Token
```

```
- SMS-Code
- E-Mail-Code
```

Merksatz:

```
OTP = zeitlich oder einmalig gültiger Code.
```

Token

Ein Token ist ein digitaler Nachweis für Zugriff.

Ein Token kann enthalten:

```
- Benutzerinformationen
- Ablaufzeit
- Berechtigungen
- Aussteller
- Signatur
- Sitzungsbezug
```

Tokens werden häufig bei Webanwendungen und APIs verwendet.

Merksatz:

```
Token = digitaler Zugriffsnachweis.
```

Bearer Token

Ein Bearer Token wird häufig bei APIs genutzt.

Er wird oft im HTTP-Header übertragen:

```
Authorization: Bearer <token>
```

Wichtig:

```
Wer den Bearer Token besitzt,
kann ihn oft verwenden.
```

Deshalb muss er wie ein Passwort geschützt werden.

Merksatz:

```
Bearer Token wie ein Passwort behandeln.
```

Session

Eine Session ist eine Sitzung zwischen Client und Anwendung.

Beispiel:

```
Benutzer meldet sich an.
Server erstellt Session.
Browser erhält Session-Cookie.
Weitere Anfragen werden dieser Session zugeordnet.
```

Sessions helfen, einen Benutzer über mehrere Anfragen hinweg wiederzuerkennen.

Merksatz:

```
Session verbindet mehrere Anfragen zu einer Sitzung.
```

Session-Cookie

Ein Session-Cookie enthält häufig eine Session-ID.

Der Browser sendet dieses Cookie bei weiteren Anfragen mit.

Dadurch weiß der Server:

```
Diese Anfrage gehört zu dieser Sitzung.
```

Wichtig:

```
Session-Cookies müssen geschützt werden.
```

Merksatz:

Session-Cookie hält die Anmeldung über mehrere Anfragen fest.

Sichere Cookie-Eigenschaften

Wichtige Cookie-Sicherheitsattribute:

Attribut	Bedeutung
Secure	Cookie nur über HTTPS senden
HttpOnly	Cookie nicht per JavaScript auslesbar
SameSite	Schutz gegen bestimmte Cross-Site-Angriffe
Expires / Max-Age	Ablaufzeit des Cookies

Merksatz:

Sichere Cookies schützen Sitzungen.

Session Hijacking

Session Hijacking bedeutet:

Ein Angreifer übernimmt eine gültige Sitzung.

Das kann passieren, wenn ein Session-Cookie gestohlen wird.

Mögliche Ursachen:

- unsichere Verbindung
- XSS-Schwachstelle
- unsichere Cookies
- gestohlener Token
- Malware auf Client
- unsichere Speicherung

Merksatz:

Wer eine Session übernimmt,
kann oft wie der Benutzer handeln.

Session Timeout

Ein Session Timeout beendet eine Sitzung nach einer bestimmten Zeit.

Gründe:

```
- Schutz bei vergessener Abmeldung
- Begrenzung gestohlener Sessions
- Ressourcen sparen
- Sicherheitsrichtlinie erfüllen
```

Beispiel:

```
Benutzer ist 30 Minuten inaktiv.
Anwendung meldet ihn ab.
```

Merksatz:

```
Timeout begrenzt die Lebensdauer einer Sitzung.
```

Single Sign-on

Single Sign-on wird kurz genannt:

```
SSO
```

SSO bedeutet:

```
Ein Benutzer meldet sich einmal an
und kann danach mehrere Dienste nutzen.
```

Beispiele:

```
- Anmeldung über Microsoft Entra ID
- Anmeldung über Active Directory
- Anmeldung über Google Workspace
- Anmeldung über zentralen Identitätsanbieter
```

Vorteil:

```
weniger separate Passwörter
zentrale Verwaltung
bessere Benutzerfreundlichkeit
```

Merksatz:

```
SSO = einmal anmelden,
mehrere Dienste nutzen.
```

Identity Provider

Ein Identity Provider wird kurz genannt:

```
IdP
```

Ein IdP stellt Identitäten bereit und bestätigt Anmeldungen.

Beispiele:

```
- Active Directory
- Microsoft Entra ID
- Keycloak
- Okta
- Google Workspace
```

Eine Anwendung kann dem IdP vertrauen, statt eigene Benutzer vollständig selbst zu verwalten.

Merksatz:

```
IdP bestätigt die Identität für Anwendungen.
```

Service Provider

Ein Service Provider ist die Anwendung oder der Dienst, der Zugriff gewährt.

Beispiel:

```
Benutzer meldet sich über IdP an.
Webanwendung vertraut dem IdP.
Webanwendung lässt Benutzer hinein.
```

Merksatz:

```
Service Provider nutzt Identitätsinformationen vom IdP.
```

OAuth 2.0

OAuth 2.0 ist ein Autorisierungsframework.

Es wird häufig verwendet, damit Anwendungen Zugriff auf bestimmte Ressourcen erhalten können, ohne direkt das Passwort des Benutzers zu kennen.

Beispiel:

```
Eine App darf auf Kalenderdaten zugreifen,
aber nicht auf das Passwort des Benutzers.
```

Merksatz:

```
OAuth 2.0 regelt delegierten Zugriff.
```

OpenID Connect

OpenID Connect baut auf OAuth 2.0 auf.

Es wird für Authentifizierung verwendet.

Kurz:

```
OAuth 2.0:
Zugriff erlauben

OpenID Connect:
Identität bestätigen
```

Merksatz:

OpenID Connect ergänzt OAuth um Anmeldung und Identität.

SAML

SAML steht für:

Security Assertion Markup Language

SAML wird häufig für Single Sign-on in Unternehmen verwendet.

Dabei tauschen Identitätsanbieter und Anwendung Anmeldeinformationen aus.

Typisch bei:

- Unternehmensanwendungen
- Cloud-Diensten
- älteren SSO-Umgebungen

Merksatz:

SAML ist ein verbreiteter SSO-Standard.

Rollen

Rollen bündeln Berechtigungen.

Beispiele:

- Benutzer
- Administrator
- Support
- Prüfer
- Projektleitung
- Gast

Ein Benutzer erhält eine Rolle, und die Rolle enthält Rechte.

Merksatz:

```
Rolle = Sammlung von Berechtigungen.
```

Rechte

Rechte beschreiben konkrete erlaubte Aktionen.

Beispiele:

```
- lesen
- schreiben
- ändern
- löschen
- ausführen
- freigeben
- verwalten
- Benutzer anlegen
```

Merksatz:

```
Recht = konkrete erlaubte Aktion.
```

Gruppen

Gruppen bündeln Benutzer.

Beispiel:

```
Gruppe:
IT-Support
```

Mitglieder:

```
Felix
Maria
Jonas
```

Rechte werden häufig Gruppen zugewiesen, nicht einzelnen Benutzern.

Vorteil:

```
einfacher zu verwalten
übersichtlicher
weniger Fehler
```

Merksatz:

```
Benutzer in Gruppen,
Rechte an Gruppen.
```

RBAC

RBAC steht für:

```
Role-Based Access Control
```

Das bedeutet:

```
Zugriff wird über Rollen gesteuert.
```

Beispiel:

```
Rolle Administrator:
darf Benutzer verwalten

Rolle Leser:
darf Inhalte lesen

Rolle Bearbeiter:
darf Inhalte ändern
```

Merksatz:

```
RBAC = Rechte über Rollen verwalten.
```

Least Privilege

Least Privilege bedeutet:

```
Benutzer und Dienste erhalten nur die Rechte,
die sie wirklich benötigen.
```

Beispiel:

```
Ein Dienst braucht nur Leserechte.
Dann bekommt er keine Schreibrechte.
```

Ziel:

```
Schaden begrenzen,
falls ein Konto missbraucht wird.
```

Merksatz:

```
Nur so viele Rechte wie nötig,
so wenige wie möglich.
```

Need-to-know-Prinzip

Need-to-know bedeutet:

```
Zugriff nur auf Informationen,
die für die Aufgabe benötigt werden.
```

Beispiel:

```
Personalabteilung darf Personaldaten sehen.
Andere Abteilungen nicht.
```

Merksatz:

```
Zugriff nur,
wenn er fachlich notwendig ist.
```

Privileged Access

Privileged Access bedeutet:

```
besonders mächtiger Zugriff.
```

Beispiele:

```
- Administratorrechte
- Domain-Admin
- Root-Zugriff
- Datenbankadministrator
- Cloud-Administrator
- Firewall-Administrator
```

Solche Zugriffe müssen besonders geschützt werden.

Merksatz:

```
Adminrechte sind besonders kritisch.
```

Schutz für Admin-Konten

Admin-Konten sollten besonders abgesichert werden.

Maßnahmen:

```
- MFA
- getrennte Admin-Konten
- kein tägliches Arbeiten mit Admin-Konto
- starke Protokollierung
- Zugriffsbeschränkung
- Least Privilege
- regelmäßige Prüfung
- keine geteilten Konten
```

Merksatz:

```
Admin-Konten besonders schützen und überwachen.
```

Geteilte Konten

Geteilte Konten sind Konten, die von mehreren Personen genutzt werden.

Problem:

```
Aktionen sind schwer einer Person zuzuordnen.
```

Beispiel:

```
Alle nutzen admin.
```

Besser:

```
persönliche Konten
plus
gezielte Adminrechte
```

Merksatz:

```
Geteilte Konten vermeiden,
weil Nachvollziehbarkeit fehlt.
```

Protokollierung

Sicherheitsrelevante Anmeldungen und Zugriffe sollten protokolliert werden.

Wichtige Informationen:

```
- Benutzer
- Zeitpunkt
- Quelle
- Zielsystem
- erfolgreiche Anmeldung
- fehlgeschlagene Anmeldung
- Rechteänderung
- kritische Aktion
```

Merksatz:

```
Ohne Logs keine saubere Nachvollziehbarkeit.
```

Brute-Force-Angriff

Brute Force bedeutet:

```
viele Passwörter werden ausprobiert.
```

Ziel:

```
ein gültiges Passwort finden.
```

Gefährdet sind besonders:

```
- Weblogin
- SSH
- RDP
- VPN
- E-Mail
- Cloud-Konten
```

Schutzmaßnahmen:

```
- MFA
- Kontosperrung
- Rate Limiting
- starke Passwörter
- Monitoring
- IP-Beschränkung
- Fail2ban
```

Merksatz:

```
Brute Force wird durch Begrenzung und MFA erschwert.
```

Phishing

Phishing bedeutet:

> Benutzer werden getäuscht,
> um Zugangsdaten oder Tokens preiszugeben.

Beispiele:

> - gefälschte Login-Seite
> - angebliche Passwortänderung
> - falsche Paketbenachrichtigung
> - gefälschte Rechnung
> - angeblicher IT-Support

Schutzmaßnahmen:

> - Schulung
> - MFA
> - sichere Mailfilter
> - Domainprüfung
> - Passwortmanager
> - Meldewege für verdächtige Mails

Merksatz:

> Phishing greift den Menschen und die Anmeldung an.

Credential Stuffing

Credential Stuffing bedeutet:

> Angreifer verwenden bekannte Zugangsdaten aus Datenlecks
> bei anderen Diensten erneut.

Problem:

```
Viele Benutzer verwenden gleiche Passwörter mehrfach.
```

Schutzmaßnahmen:

```
- keine Passwortwiederverwendung
- MFA
- Passwortmanager
- Überwachung verdächtiger Logins
```

Merksatz:

```
Wiederverwendete Passwörter sind ein großes Risiko.
```

Passwortspraying

Beim Passwortspraying probiert ein Angreifer ein häufiges Passwort gegen viele Benutzerkonten.

Beispiel:

```
Sommer2026!
```

gegen viele Konten.

Ziel:

```
Kontosperrungen vermeiden,
aber trotzdem Treffer finden.
```

Merksatz:

```
Passwortspraying testet wenige Passwörter gegen viele Konten.
```

Account Lockout

Account Lockout bedeutet:

```
Ein Konto wird nach mehreren Fehlversuchen gesperrt.
```

Ziel:

```
Brute-Force-Angriffe erschweren.
```

Wichtig:

```
Sperrregeln müssen sinnvoll eingestellt sein,
damit Angriffe erschwert werden,
aber Benutzer nicht unnötig blockiert werden.
```

Merksatz:

```
Kontosperrung begrenzt Passwortversuche.
```

Rate Limiting

Rate Limiting begrenzt, wie viele Anfragen in einer bestimmten Zeit erlaubt sind.

Beispiel:

```
maximal 5 Loginversuche pro Minute
```

Ziel:

```
automatisierte Angriffe erschweren
```

Merksatz:

```
Rate Limiting bremst massenhafte Versuche.
```

CAPTCHA

CAPTCHA soll prüfen, ob ein Benutzer wahrscheinlich ein Mensch ist.

Typische Nutzung:

- Login
- Registrierung
- Formular
- Passwortzurücksetzung

CAPTCHA kann automatisierte Angriffe erschweren, ersetzt aber keine starke Authentifizierung.

Merksatz:

CAPTCHA erschwert Bots,
ersetzt aber keine Sicherheit.

Passwortmanager

Ein Passwortmanager speichert Passwörter verschlüsselt.

Vorteile:

- lange zufällige Passwörter
- weniger Passwortwiederverwendung
- einfacheres Verwalten vieler Konten
- Schutz vor Tippfehlern
- teilweise Schutz gegen Phishing durch Domainprüfung

Merksatz:

Passwortmanager helfen bei starken eindeutigen Passwörtern.

Sichere Passwortregeln

Sinnvolle Regeln:

- lange Passwörter erlauben
- Passwortmanager unterstützen
- bekannte kompromittierte Passwörter blockieren
- MFA nutzen
- keine unnötig kurzen Ablaufintervalle ohne Grund
- Passwortwiederverwendung vermeiden

Merksatz:

```
Lange eindeutige Passwörter plus MFA sind sehr stark.
```

Anwendungssicherheit

Anwendungssicherheit bedeutet:

```
Anwendungen so entwickeln,
konfigurieren und betreiben,
dass Missbrauch erschwert wird.
```

Typische Themen:

```
- sichere Anmeldung
- Rechteprüfung
- Eingabevalidierung
- Schutz vor SQL-Injection
- Schutz vor XSS
- sichere Sessions
- sichere APIs
- Logging
- Updates
- sichere Konfiguration
```

Merksatz:

```
Anwendungssicherheit schützt Dienste auf Schicht 7.
```

SQL-Injection

SQL-Injection ist ein Angriff, bei dem schädliche Daten in eine Datenbankabfrage eingeschleust werden.

Beispiel sinngemäß:

```
Eingabefeld wird nicht geprüft.
Angreifer fügt SQL-Befehl ein.
```

Datenbank führt unerwünschte Abfrage aus.

Schutzmaßnahmen:

- parametrisierte Abfragen
- Eingabevalidierung
- minimale Datenbankrechte
- Fehlermeldungen nicht offenlegen

Merksatz:

SQL-Injection nutzt unsichere Datenbankeingaben aus.

Cross-Site Scripting

Cross-Site Scripting wird kurz genannt:

XSS

Dabei wird schädlicher Code in eine Webseite eingeschleust.

Ziel:

im Browser anderer Benutzer ausgeführt werden.

Mögliche Folgen:

- Session-Cookie stehlen
- Inhalte manipulieren
- Benutzeraktionen auslösen
- Phishing innerhalb der Anwendung

Schutzmaßnahmen:

- Ausgaben korrekt escapen
- Eingaben prüfen
- Content Security Policy

- HttpOnly-Cookies
- sichere Frameworks nutzen

Merksatz:

XSS greift Benutzer über die Webseite an.

CSRF

CSRF steht für:

Cross-Site Request Forgery

Dabei wird ein angemeldeter Benutzer dazu gebracht, unbeabsichtigt eine Aktion in einer Webanwendung auszuführen.

Beispiel:

Benutzer ist angemeldet.
Angreifer bringt Browser dazu,
eine unerwünschte Anfrage zu senden.

Schutzmaßnahmen:

- CSRF-Token
- SameSite-Cookies
- Prüfung von Origin oder Referer
- kritische Aktionen bestätigen lassen

Merksatz:

CSRF missbraucht eine bestehende Anmeldung.

Unsichere direkte Objektreferenz

Eine unsichere direkte Objektreferenz entsteht, wenn eine Anwendung nur eine ID prüft, aber nicht die Berechtigung.

Beispiel:

```
/rechnung/1001
```

Benutzer ändert URL zu:

```
/rechnung/1002
```

Wenn die Anwendung nicht prüft, ob Benutzer Rechnung 1002 sehen darf, entsteht ein Sicherheitsproblem.

Merksatz:

```
Jede Ressource braucht Berechtigungsprüfung.
```

Sichere APIs

APIs müssen besonders sauber geschützt werden.

Wichtige Punkte:

```
- Authentifizierung
- Autorisierung
- sichere Tokens
- HTTPS
- Rate Limiting
- Eingabevalidierung
- saubere Fehlercodes
- keine geheimen Daten in Antworten
- Logging
- Versionierung
```

Merksatz:

```
API-Sicherheit ist Anwendungssicherheit auf Schnittstellenebene.
```

Fehlermeldungen

Fehlermeldungen sollten hilfreich, aber nicht zu verräterisch sein.

Schlecht:

```
Datenbankfehler mit Tabellenname und SQL-Abfrage wird angezeigt.
```

Besser:

```
Allgemeine Fehlermeldung für Benutzer.
Details nur im geschützten Serverlog.
```

Merksatz:

```
Interne Details gehören in Logs,
nicht in öffentliche Fehlermeldungen.
```

Updates und Patches

Anwendungen und Abhängigkeiten müssen aktuell gehalten werden.

Risiken bei veralteter Software:

```
- bekannte Schwachstellen
- unsichere Bibliotheken
- veraltete Authentifizierungsverfahren
- fehlende Sicherheitsupdates
- Angriffe über bekannte Exploits
```

Merksatz:

```
Veraltete Anwendungen sind ein Sicherheitsrisiko.
```

Sichere Konfiguration

Auch sichere Software kann unsicher betrieben werden, wenn sie falsch konfiguriert ist.

Beispiele:

- Standardpasswörter
- Debug-Modus aktiv
- offene Adminbereiche
- zu viele Rechte
- unsichere CORS-Regeln
- veraltete TLS-Versionen
- unnötige Dienste aktiv
- Verzeichnislisting aktiv

Merksatz:

Falsche Konfiguration kann sichere Software unsicher machen.

Fehlersuche bei Anmeldung und Rechten

Eine sinnvolle Reihenfolge:

1. Benutzer existiert?
2. Passwort oder Schlüssel korrekt?
3. Konto aktiv?
4. MFA korrekt?
5. Gruppe oder Rolle vorhanden?
6. Rechte korrekt?
7. Session oder Token gültig?
8. Zeit korrekt?
9. Dienst erreichbar?
10. Logs prüfen.

Merksatz:

Bei Loginproblemen Identität,
Faktor,
Rolle
und Logs prüfen.

Fehlerbild: Anmeldung schlägt fehl

Mögliche Ursachen:

- falsches Passwort
- Benutzername falsch
- Konto gesperrt
- Konto deaktiviert
- Passwort abgelaufen
- MFA fehlt
- falscher Identity Provider
- LDAP oder AD nicht erreichbar
- Zeitabweichung
- Token ungültig

Merksatz:

Loginfehler sind nicht immer nur Passwortfehler.

Fehlerbild: Zugriff verweigert

Mögliche Ursachen:

- Benutzer ist nicht in richtiger Gruppe
- Rolle fehlt
- Berechtigung fehlt
- Ressource gehört anderem Benutzer
- Richtlinie blockiert
- Token enthält falsche Rechte
- Anwendung prüft Rechte fehlerhaft
- alte Session enthält alte Rechte

Merksatz:

Zugriff verweigert = Autorisierung prüfen.

Fehlerbild: Benutzer ist angemeldet, sieht aber nichts

Mögliche Ursachen:

- keine Gruppenmitgliedschaft
- falsche Rolle
- fehlende Lizenz
- falscher Mandant
- Berechtigungen nicht synchronisiert
- Anwendung filtert Daten nach Rechten
- Session muss erneuert werden

Merksatz:

Erfolgreiche Anmeldung bedeutet noch keine sichtbaren Berechtigungen.

Fehlerbild: MFA funktioniert nicht

Mögliche Ursachen:

- falsches Gerät registriert
- Uhrzeit falsch
- Token abgelaufen
- Benutzer hat neues Smartphone
- Push-Benachrichtigung blockiert
- Konto nicht korrekt für MFA registriert
- Netzwerkverbindung fehlt

Merksatz:

MFA-Probleme mit Gerät,
Zeit,
Registrierung
und Konto prüfen.

Fehlerbild: Token abgelaufen

Mögliche Ursachen:

- Ablaufzeit erreicht
- Benutzer wurde abgemeldet

- Refresh Token ungültig
- Systemzeit falsch
- Token wurde widerrufen
- Session wurde beendet

Merksatz:

> Tokens haben Lebensdauer und können widerrufen werden.

Einordnung in das OSI-Modell

Thema	Schicht
Loginformular	7
API-Token	7
Session-Cookie	7 mit Schicht-5-Bezug
LDAP-Anmeldung	7
OAuth / OpenID Connect	7
SAML	7
TLS-Schutz	6 mit Schicht-7-Bezug
TCP-Port	4
IP-Adresse	3
Benutzerrechte	7 / Anwendungsebene

Merksatz:

> Anmeldung und Rechte sind typische Schicht-7-Themen.

Typische IHK-Fragen

In AP1 und AP2 kann zum Beispiel gefragt werden:

- Was bedeutet Authentifizierung?
- Was bedeutet Autorisierung?
- Was ist der Unterschied zwischen Authentifizierung und Autorisierung?
- Was ist MFA?
- Warum ist MFA sinnvoll?

- Was ist ein Token?
- Was ist eine Session?
- Was ist ein Session-Cookie?
- Was bedeutet Single Sign-on?
- Was ist ein Identity Provider?
- Was ist Least Privilege?
- Warum sollten Rechte über Gruppen vergeben werden?
- Warum sind geteilte Admin-Konten problematisch?
- Was ist Brute Force?
- Was ist Phishing?
- Was ist SQL-Injection?
- Was ist XSS?
- Warum sind sichere Fehlermeldungen wichtig?

Typische Prüfungsfallen

Authentifizierung ist nicht Autorisierung.

Anmeldung bedeutet nicht automatisch Zugriff.

Rechte müssen pro Ressource geprüft werden.

MFA erhöht die Sicherheit deutlich.

Token müssen wie Passwörter geschützt werden.

Session-Cookies müssen sicher gesetzt werden.

HTTPS ist wichtig für Login und Tokens.

Bearer Token kann von jedem genutzt werden,
der ihn besitzt.

Geteilte Konten erschweren Nachvollziehbarkeit.

Adminrechte besonders schützen.

Least Privilege immer beachten.

Gruppen erleichtern Rechteverwaltung.

Phishing greift oft Zugangsdaten an.

SQL-Injection betrifft unsichere Datenbankeingaben.

XSS betrifft unsichere Ausgaben im Browser.

CSRF missbraucht bestehende Sitzungen.

Fehlermeldungen dürfen keine internen Details verraten.

Wichtige Begriffe kurz erklärt

Begriff	Kurze Erklärung
Authentifizierung	Identität prüfen
Autorisierung	Rechte prüfen
Identität	Benutzer, Dienst oder System
Benutzerkonto	digitale Identität eines Benutzers
Dienstkonto	Konto für Anwendungen oder Dienste
Passwort	geheimer Identitätsnachweis
Passwort-Hash	Einweg-Prüfwert eines Passworts
Salt	zusätzlicher Zufallswert beim Hashing
MFA	Multi-Faktor-Authentifizierung
OTP	Einmalpasswort
Token	digitaler Zugriffsnachweis
Bearer Token	nutzbarer Zugriffstoken
Session	Sitzung zwischen Client und Anwendung
Session-Cookie	Cookie zur Sitzungszuordnung
SSO	Single Sign-on
IdP	Identity Provider
RBAC	rollenbasierte Zugriffskontrolle
Least Privilege	nur notwendige Rechte
Brute Force	massenhaftes Passwortprobieren
Phishing	Täuschung zur Datenerbeutung

Begriff	Kurze Erklärung
SQL-Injection	Einschleusen von SQL-Befehlen
XSS	Einschleusen von Code in Webseiten
CSRF	Missbrauch bestehender Anmeldung

IHK-sichere Kurzformulierung

Authentifizierung und Autorisierung sind zentrale Sicherheitskonzepte auf der Anwendungsschicht. Authentifizierung prüft die Identität eines Benutzers, Dienstes oder Systems. Autorisierung prüft anschließend, welche Rechte diese Identität besitzt. Eine erfolgreiche Anmeldung bedeutet daher nicht automatisch, dass ein Benutzer alle Aktionen durchführen darf. Multi-Faktor-Authentifizierung erhöht die Sicherheit, weil neben dem Passwort ein weiterer Faktor benötigt wird. Rechte sollten möglichst über Gruppen oder Rollen vergeben werden und dem Least-Privilege-Prinzip folgen. Tokens, Session-Cookies und Passwörter müssen geschützt werden, da sie Zugriff auf Anwendungen und Dienste ermöglichen können.

Merksätze

```
Authentifizierung = Wer bist du?

Autorisierung = Was darfst du?

Erst Identität,
dann Rechte.

Anmeldung ist nicht gleich Berechtigung.

Benutzerkonto = digitale Identität.

Dienstkonto = Konto für Dienste.

Passwörter nicht im Klartext speichern.

Passwort-Hash schützt gespeicherte Passwörter.

Salt macht Hashes robuster.

MFA kombiniert mehrere Faktoren.

OTP = Einmalpasswort.
```

Token = digitaler Zugriffsnachweis.

Bearer Token wie Passwort schützen.

Session verbindet mehrere Anfragen.

Session-Cookie schützt Anmeldung.

Secure,
HttpOnly
und SameSite schützen Cookies.

SSO = einmal anmelden,
mehrere Dienste nutzen.

IdP bestätigt Identität.

OAuth regelt delegierten Zugriff.

OpenID Connect ergänzt Anmeldung.

RBAC verwaltet Rechte über Rollen.

Least Privilege = nur notwendige Rechte.

Admin-Konten besonders schützen.

Geteilte Konten vermeiden.

Brute Force probiert Passwörter.

Phishing täuscht Benutzer.

SQL-Injection nutzt unsichere Eingaben.

XSS greift Benutzer im Browser an.

CSRF missbraucht bestehende Sitzungen.

Anwendungssicherheit ist Schicht-7-Sicherheit.