Lernzettel

Sicherheit, Datenschutz und gesellschaftliche Verantwortung in OS: Zugriffskontrollen, Privilege, Schutz gegen Daten- und Identitätsdiebstahl

Universität: Technische Universität Berlin
Kurs/Modul: Systemprogrammierung

Erstellungsdatum: September 6, 2025

Lernzettel: Sicherheit, Datenschutz und gesellschaftliche Verantwortung in OS: Zugriffskontrollen, Privilege, Schutz gegen Daten- und Identitätsdiebstahl

(1) Überblick und Zielsetzung. Moderne Betriebssysteme schützen Ressourcen, Vertraulichkeit, Integrität und Verfügbarkeit. Zugriffskontrollen, Privilege-Management und Gegenmaßnahmen gegen Daten- und Identitätsdiebstahl sind zentrale Bausteine. Darüber hinaus trägt ein verantwortungsvoll gestaltetes OS zur gesellschaftlichen Sicherheit (Kritische Infrastrukturen) und zur Nachhaltigkeit bei, z. B. durch effiziente Ressourcennutzung.

(2) Zugriffskontrollen – Grundbegriffe. Zugriffskontrollen steuern, wer was im System tun darf. Wichtige Konzepte:

- Authentifizierung vs. Autorisierung
- Zugriffskontrollmodelle: DAC (Discretionary Access Control) vs. MAC (Mandatory Access Control)
- ACLs (Access Control Lists) und Berechtigungsbits
- Capabilities-basierte Sicherheit (CAPs) vs. User IDs (UID) / Group IDs (GID)
- Prinzip der minimalen Rechte (Least Privilege) und Privilege Separation

(3) Sicherheits- und Privilege-Modelle. Sichere Systeme verwenden klare Modelle zur Autorisierung:

- RBAC (Role-Based Access Control): Rollen definieren Berechtigungen
- ABAC (Attribute-Based Access Control): Berechtigungen via Attributen von Subjekten, Objekten und Kontext
- MAC-basierte Modelle mit Sicherheitslabels (z. B. SELinux/AppArmor/SMACK) zur strikten Trennung
- Privilege Escalation vermeiden: ordentliche Trennung von Kernel- und Benutzer-Modus; `setuid`-Programme kritisch prüfen

(4) Privilege und Privilege-Escalation – Konzepte. Wichtige Mechanismen:

- Kernelmodus vs. Benutzermodus
- Privilege-Eskalation verhindern durch Drop von Rechten nach Ausführung (setuid-root vermeiden oder minimieren)
- Capabilities statt Voll-Root-Berechtigungen: $\texttt{cap}_s et$,

(5) Schutz gegen Daten- und Identitätsdiebstahl. Kernprinzipien und konkrete Maßnahmen:

- Starke Authentifizierung: Passwörter sicher speichern (Salz, Hashing mit Argon2/bcrypt/scrypt)

- Mehrfaktor-Authentifizierung (MFA) und Token-basierte Verfahren
- Verschlüsselung: TLS im Transport, Verschlüsselung im Ruhezustand (z. B. LUKS), sicheres Key-Management
- Integrität und Authentizität: Prüfsummen, digitale Signaturen
- Auditing, Logging und Anomalieerkennung; Least-Privilege-Prinzip auch in Logging
- Sichere Standards wie OAuth/OpenID Connect, OAuth 2.0-Flow sicher implementieren

(6) Isolation, Sandbox und Schutzmechanismen. Zur Minimierung von Privilege-Verletzungen:

- Namespaces, Chroot-Umgebungen, Containerisierung (Docker, OCI)
- Seccomp, AppArmor/SELinux, CAP_SETPCAP u. ä.
- Capabilities Bounding Set, Privilege dropping, Least Privilege
- Verschiedene Formen der Isolation: User Namespace, PID Namespace, Network Namespace

(7) Datenschutzaspekte in OS. Datenschutz-präventive Merkmale im Betriebssystem:

- Privacy by Design: Minimierung der erhobenen Daten, Zweckbindung
- Logging mit Datenschutzprinzipien (Zweckbindung, Zugriffskontrollen, Anonymisierung)
- Pseudonymisierung und Data Minimization
- Sichere Speicher- und Kommunikationspfade (Verschlüsselung, sichere Zufallszahlen)
- Transparenz über Datennutzung und Retention Policy

(8) Gesellschaftliche Verantwortung und Nachhaltigkeit. OS-Sicherheit hat Auswirkungen auf die Gesellschaft und Umwelt:

- Schutz kritischer Infrastrukturen: Verfügbarkeit, Integrität, Geheimhaltung
- Verantwortungsvolle Offenlegung von Sicherheitslücken (Disclosure)
- Nachhaltigkeit durch effiziente Systeme, Energiereduktion, ressourcenschonende Virtualisierung
- Barrierefreiheit und faire Nutzung von Technologien

(9) Praktische Hinweise für die Programmierung in der Systempraxis. Hinweise für die Praxis in C bzw. Systemprogrammierung:

- Einsatz sicherer Standard-APIs zur Zugriffskontrolle, z. B. `chmod`, `chown`, `setuid`, `capset`
- Verwende minimal privilegierte Prozesse, droppe Rechte nach Start

- Nutze Containerisierung mit sicheren Standards, Seccomp-Filter, AppArmor/SELinux-Richtlinien
- Berücksichtige Datenschutzanforderungen bei Logging und Speicherzugriff

Hinweise zur Umsetzung. Dieser Lernzettel fasst zentrale Konzepte zusammen und dient der Orientierung. Nutze ihn als Basis für vertiefende Übungen zu Zugriffskontrollen, Privilege-Management, Schutz gegen Daten- und Identitätsdiebstahl sowie gesellschaftliche Verantwortung im OS-Kontext.