Vorlesung (WS 2014/15)

Sicherheit: Fragen und Lösungsansätze

Dr. Thomas P. Ruhroth

TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

**[mit freundlicher Genehmigung basierend
auf einem Foliensatz von
Prof. Dr. Claudia Eckert (TU München)]**

Literatur:

Claudia Eckert: IT-Sicherheit: Konzept - Verfahren - Protokolle, 7., überarb. und erw. Aufl., Oldenbourg, 2012.

E-Book: http://www.ub.tu-dortmund.de/katalog/titel/1362263

Agenda

- Umsetzungkonzepzte
- Anfang Sicherheitsprotokolle

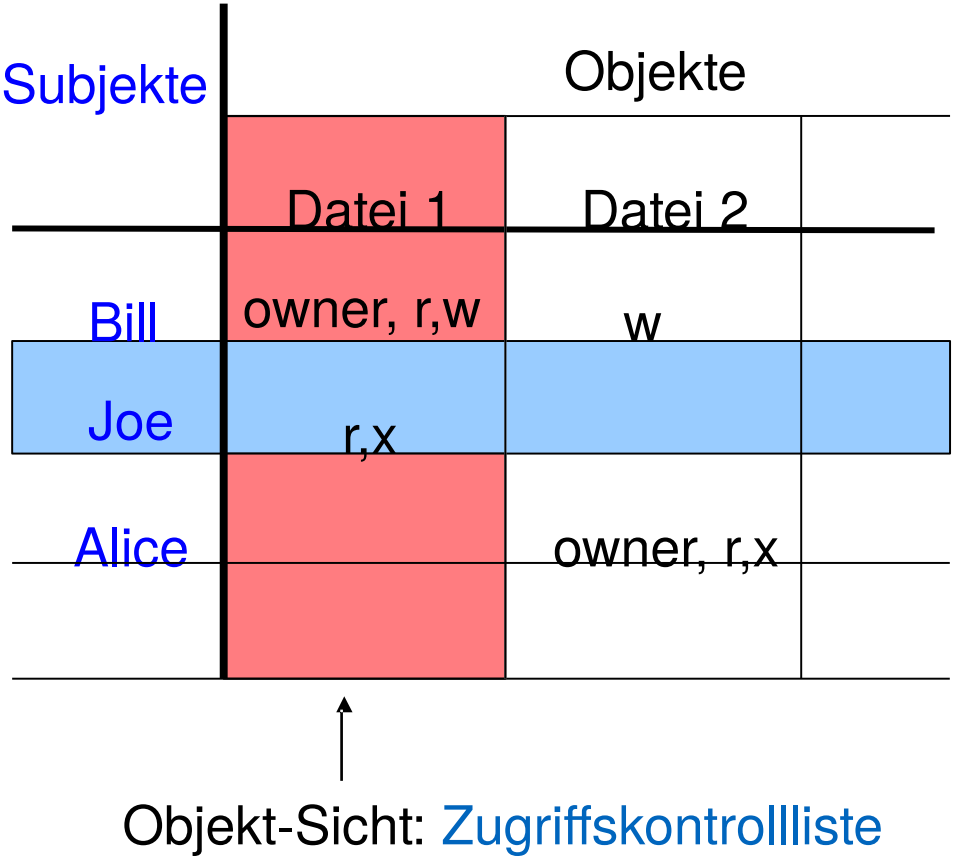
Umsetzungskonzepte

Konzepte zur Umsetzung der Rechteverwaltung

Zugriffsmatrix

- dünn besetzt (sparse)
- Implementierung als Tabelle ist ineffizient

Subjekt-Sicht: Capability-Listen →



Objekt-Sicht: Zugriffskontrollliste

Gängige Konzepte

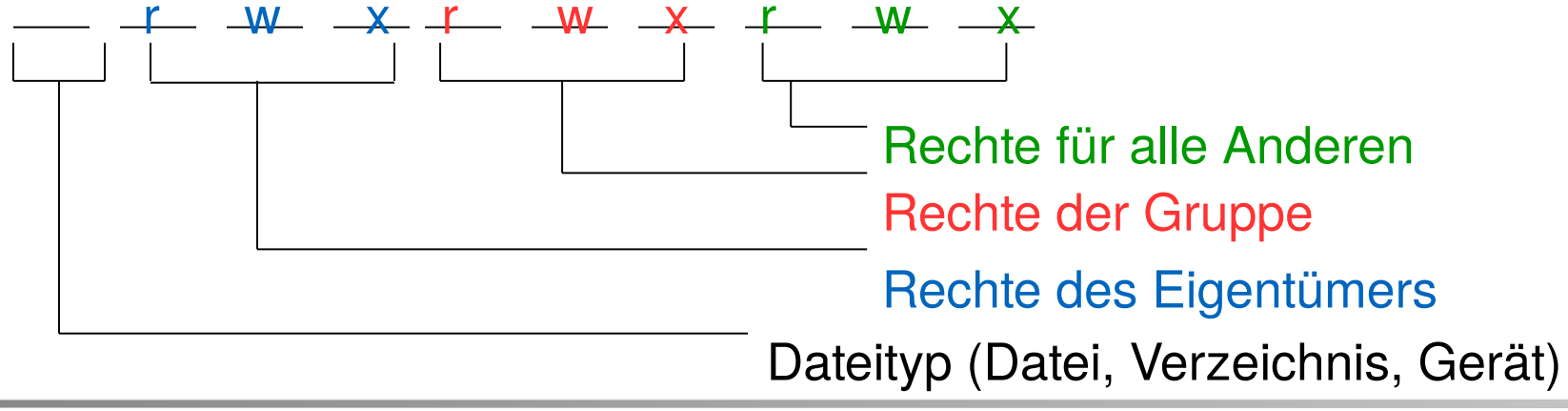
- Zugriffskontrolllisten
- Capability-Listen
- Domain-Type-Enforcement

Zugriffskontrollliste
Access-Control-List (ACL)

- Spaltenweise Realisierung der Matrix
- ACLs: am häufigsten eingesetztes Konzept in klassischen BS u.a. in UNIX/Linux, Windows 2000, XP, Microsoft Vista
- Eine ACL ist eine geschützte Datenstruktur des BS

Beispiel: ACL unter Unix/Linux (Android): vereinfachte ACL

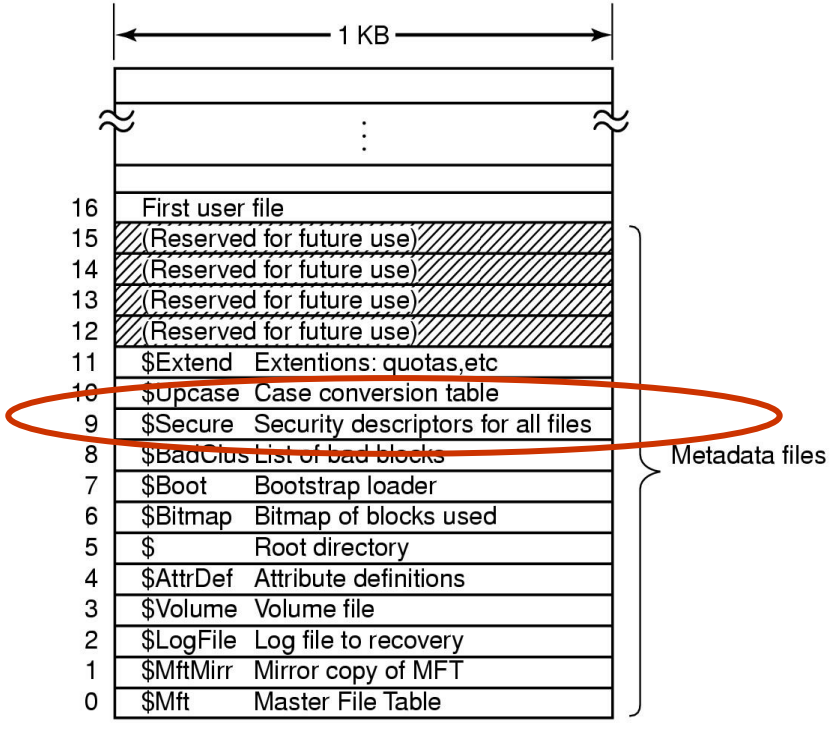
- Rechtevergabe nur an: Eigentümer, Gruppe, Rest der Welt



Beispiel: Windows Vista/Windows 7

- differenzierte ACL mit Berechtigungen bzw. Verboten für einzelne Benutzer, aber auch für Gruppen
- Berechtigungen/Verbote in Security-Descriptor gespeichert
- Security-Deskriptoren in

  Master-File-Table (MFT) verwaltet



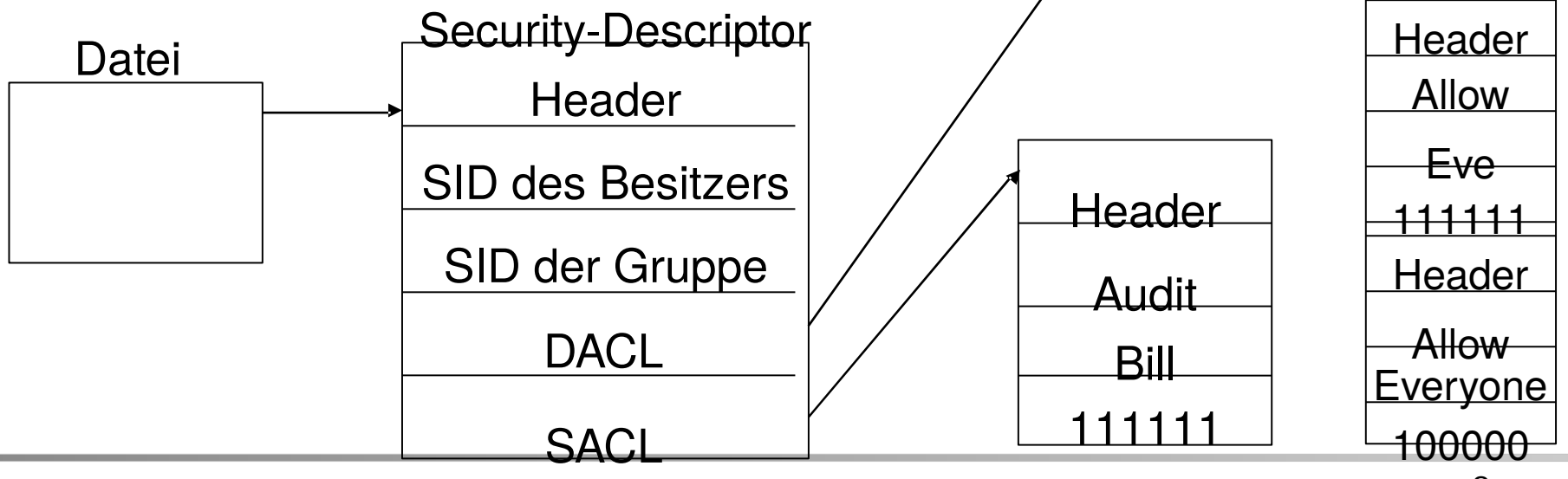
- Eintrag 0: Adressen der MFT-Blöcke
- Eintrag 2: Log-Datei, Z.B. neue Verzeichnisse.

  Strukturelle Änderungen
- Eintrag 9: Security-Deskriptoren

Security-Descriptor

Security-Descriptor enthält:

- SID = systemweit eindeutige Security-ID des Besitzers des Objekts und der Gruppe
- DACL = Discretionary ACL: Liste von ACEs
- ACE = Access-Control-Element mit allow/deny
- SACL = System-ACL, spezifiziert die Operationen, deren Nutzungen zu protokollieren sind



Capability-Konzept

Capability-Konzept Zeilenweise Realisierung der Matrix

- Capability: Zugriffsticket mit Objekt-UID und Rechte-Bits
- Capability-Besitz berechtigt zur Wahrnehmung der Rechte
- Für jedes Subjekt *s* wird eine Capability-Liste verwaltet

Beispiel Clist (Joe) = ((Datei1, {r,x}), (Datei2, {w}))

Vor- und Nachteile beider Konzepte?

Kombination aus beiden Ansätzen:

- ACL für ersten Zugriff, danach
- Ausstellen einer Capability: File-Handle (Unix) bzw. Object-Descriptor (Windows)

5.2.3 Domain-Type Enforcement

Domain-Type Enforcement, DTE

1995 vorgestellt, u.a. in Linux/Unix-Varianten umgesetzt, u.a.

Lee Badger et. al *A Domain and Type Enforcement UNIX Prototype*,
Fifth USENIX UNIX Security Symposium Proceedings, 1995

Konzept:

- vereinfachtes Capability und Rollen-Konzept, spezielle Mandatory Access Policies
- jedem Subjekt wird genau eine Domäne als Attribut zugeordnet, z.B. *domain project*
- jedem zu schützenden Objekt wird ein Typ zugeordnet, z.B. *type budget*
- Die DTE Tabelle spezifiziert, auf welche Typen eine Domäne read und/oder write Zugriff besitzt

- die Domain-Transitions Tabelle (DTT) spezifiziert, welche Domänen von einer gegebenen Domain aufgerufen werden dürfen:
  - Domänenwechsel über die Ausführung einer speziellen Operation: enter
- Domänen-Attribut für ein Subjekt ist eine Art Capability, das Subjekt gehört zur Domäne (vereinfachte Rollenmit-gliedschaft) und besitzt die Rechte Domäne
- DTE definiert mandatorische Zugriffsregeln

Implementierung:

- Domänen-Attribut als Bestandteil des Prozess-Deskriptor
- Typ-Attribut Bestandteil des Objekt-Deskriptors (z.B. inode)

Beispiel - DTE Linux Implementierung

- Prozesse werden in Domänen aufgeteilt
  - Ausführung des „Init“ Prozess erfolgt in einer Default Domäne
  - Domänenwechsel durch Ausführen eines Prozesses

- Dateien werden verschiedenen Typen zugeordnet
  - 3 Typen pro Datei (Inode):
    - etype: Typ der Datei / des Verzeichnis
    - rtype: Typ des Verzeichnis und seiner Kinder
    - utype: Typ der Kinder des Verzeichnisses

Beispiel

- Domänen: common_d, log_d
  - Default Domäne: common_d (rwx → root_t, r → log_t)
  - log_d (r → root_t, rw → log_t, x → log_xt)
  - Transition von common_d → log_d durch Aufruf von /usr/bin/rsyslogd
- Typen: root_t, log_t, log_xt
  - Default Typ: root_t
  - utype von /var/log → log_t
  - etype von /usr/bin/rsyslogd → log_xt
- Auswirkungen:
  - Syslog Prozess darf nur Logdateien schreiben.
  - Syslog darf keine Datei ausführen, die er selbst schreiben darf
  - Syslog darf keine Programme mit Typ root_t ausführen
  - Insbesondere kein /bin/bash für eine mögliche Reverse Shell

Fallbeispiel: SE Linux

Fallbeispiel: SE Linux (Security Enhanced)

- Ende 2000 als NSA Forschungsprojekt gestartet
- Nutzt Linux Security Module im Kernel (LSM)
- Ist in Version 2.6 des Linux Kernels integriert

Ziele:

- Kernel soll die Ausführung von Applikationen anhand von Regeln überwachen können
- Unterstützung von MAC, RBAC und DTE
- Subjekte (process) und Objekte (file) besitzen einen eigenen Sicherheitskontext zur Laufzeit
  - Für Objekte über Attribut beschrieben: ls – Z <filename>
  - Für Prozesse: ps axZ
  - Sicherheitskontext: user_r:role_r:type_t:mls-component

Policy Decision

Policy Decision

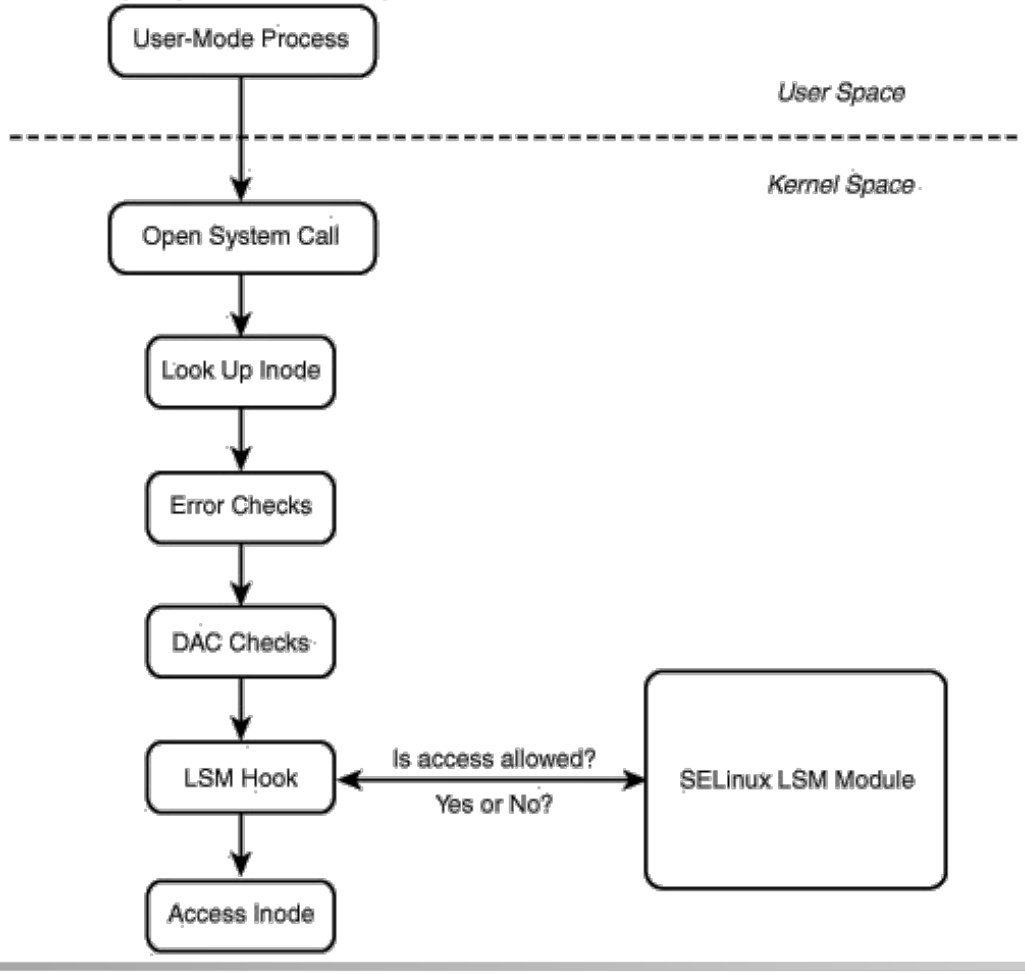
Struktur einer Regel: allow type1_t type2_t:class { operation };

Beispiel:

- allow httpd_t httpd_log_t:dir create;
  D.h. Prozesse mit Kontext-Typ httpd_t dürfen neue Dateien in Verzeichnissen mit Kontext-Typ httpd_log_dir erzeugen

Policy Enforcement

- Sicherheitsmodule erweitern den Kernel
- LSM Sicherheitschecks werden nach den Standard Linux Checks aufgerufen



Erläuterung: LSM

Linux Security Modules (LSM)

- Schnittstelle im Linux Kernel
- Jeder Syscall im Kernel ruft eine Callback Funktion aus dem Sicherheitsframework auf
  - Prüfung und evtl. Einschränkungen der Berechtigungen für den aktuellen Aufruf
  - Aktivierung des Frameworks durch Laden eines entsprechenden Moduls
- Aber: Schnittstelle könnte auch durch Angreifer genutzt werden
  - Erleichterter Einstieg in Kernel-interne Datenstrukturen
- SELinux als umfangreichste, komplexeste Lösung
  - Alternativ: Apparmor, GRSecurity, Tomoyo Linux, Smack, …

5.3 Zugriffskontrolle

Allgemeines Prinzip

- idR. Aufteilung in Berechtigungs- und Zulässigkeitskontrolle

Berechtigungskontrolle: PDP: Policy Decision Point

- Prüfung beim erstmaligem Zugriff auf ein Objekt
- von vertrauenswürdigen Systemdiensten (z.B. Dateisystem)
- Ausstellung einer Bescheinigung, z.B. File-Handle, Ticket

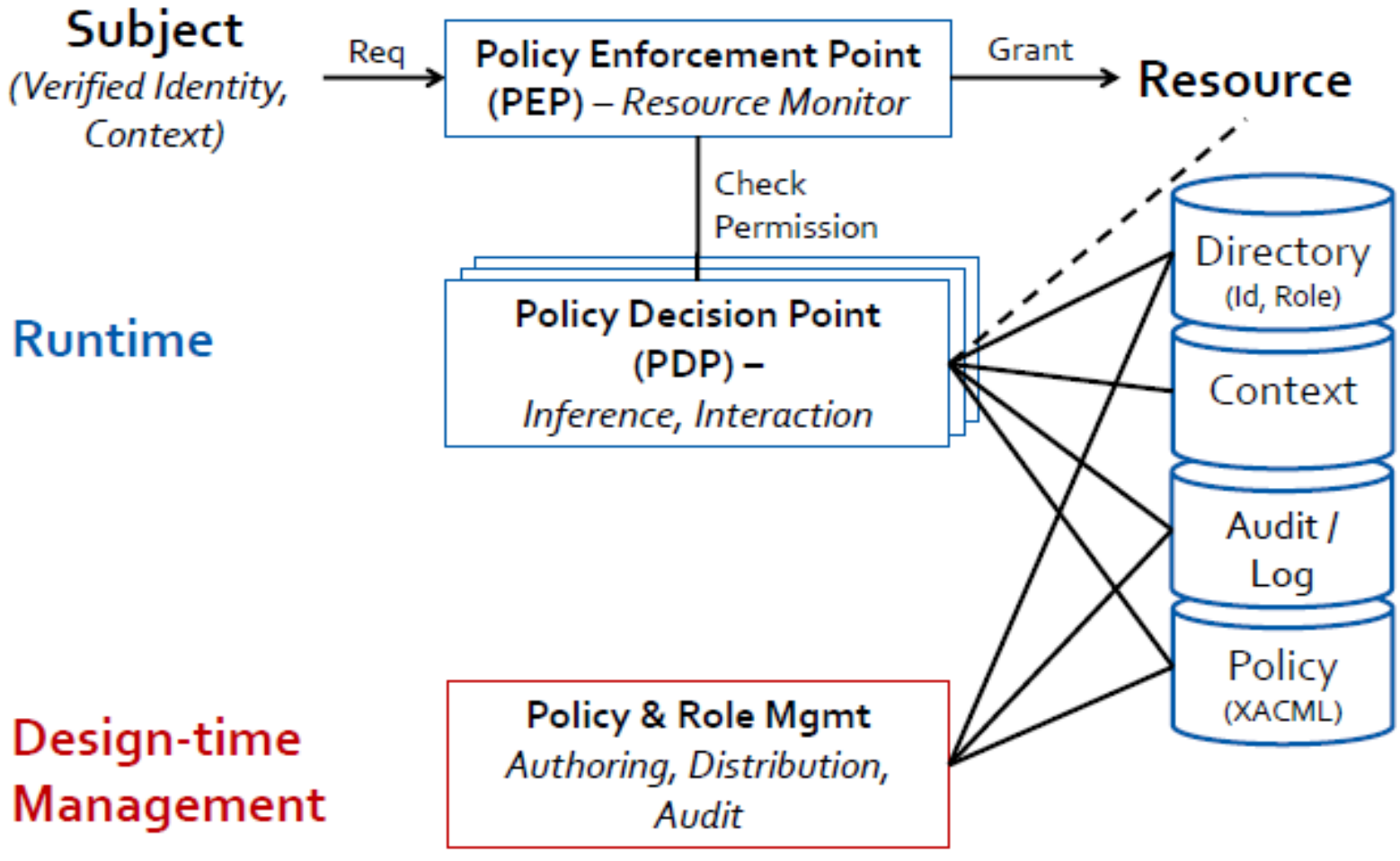
Zulässigkeitskontrolle: PEP: Policy Enforcement Point

- durch Objektverwalter (z.B. user-level Server)
- bei Objektzugriff: Prüfen der Gültigkeit der Bescheinigung
- kein Zugriff auf die Rechteinformation notwendig

Konsequenzen?

Allgemeines Schema für Zusammenspiel: PDP und PEP

LEHRSTUHL 14
SOFTWARE ENGINEERING

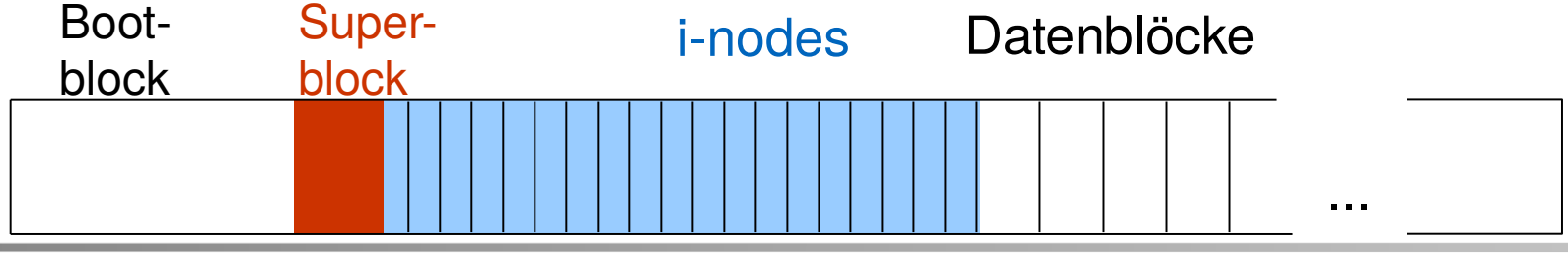


Beispiel Unix/Linux

- Subjekte/Prozesse identifiziert über UID, GUID
- Zu schützende Objekte: Dateien, Verzeichnisse
- Dateien/Verzeichnisse werden BS-intern über einen Datei-Deskriptor, die i-node (index-node), beschrieben

Datei-Deskriptor

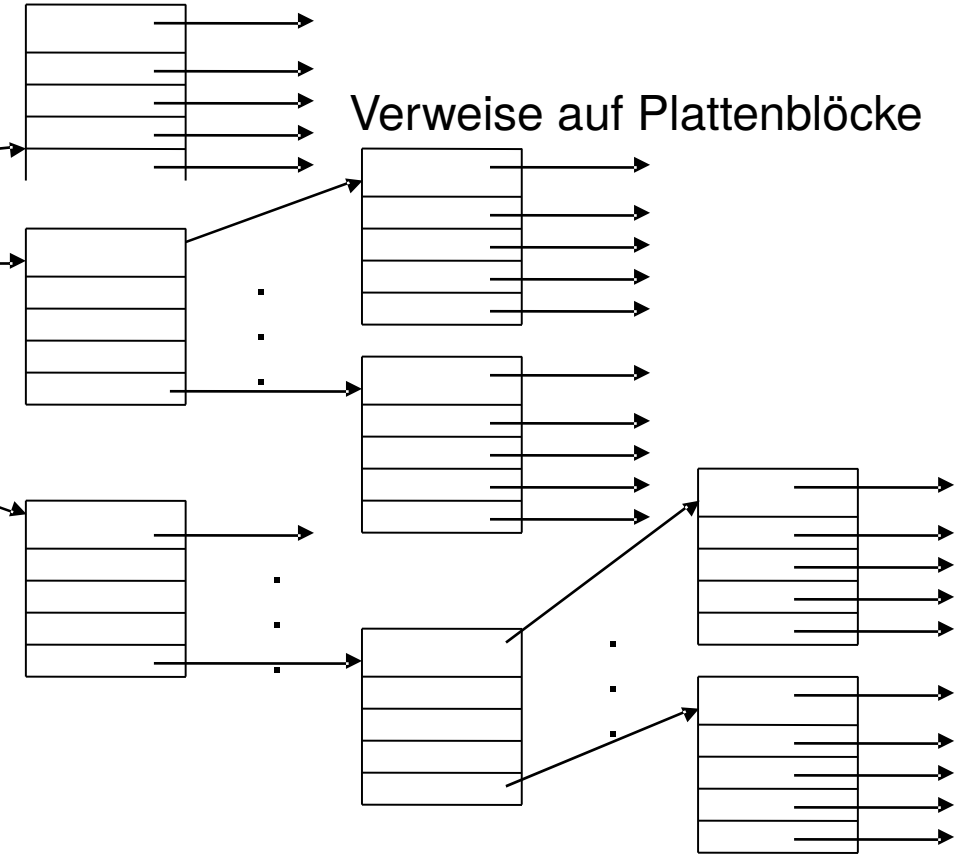
- die i-node enthält u.a. Name des Datei-Owners und die ACL
- die i-nodes werden auf der Festplatte verwaltet,
- beim Öffnen der Datei (open-call) wird i-node eingelagert



Unix-i-node

owner joe, uid
group student, guid
type regular file
perms rwxr-xr-x
accessed Feb 12 1999 3:00 P.M.
modified Feb 11 1999 10:16 A.M.
Adressen der 10 ersten Plattenblöcke
einfach indirekt
zweifach indirekt
dreifach indirekt

ACL als Bestandteil der i-node unter Unix



Ablauf bei der Zugriffskontrolle unter Unix/Linux

Open-System-Call: Angabe des Zwecks r, w, x

Aktionen des Unix Kerns (siehe nächste Folie)

(1) Laden der i-node der zu öffnenden Datei <datei_i> in i-node Tabelle des Kernels

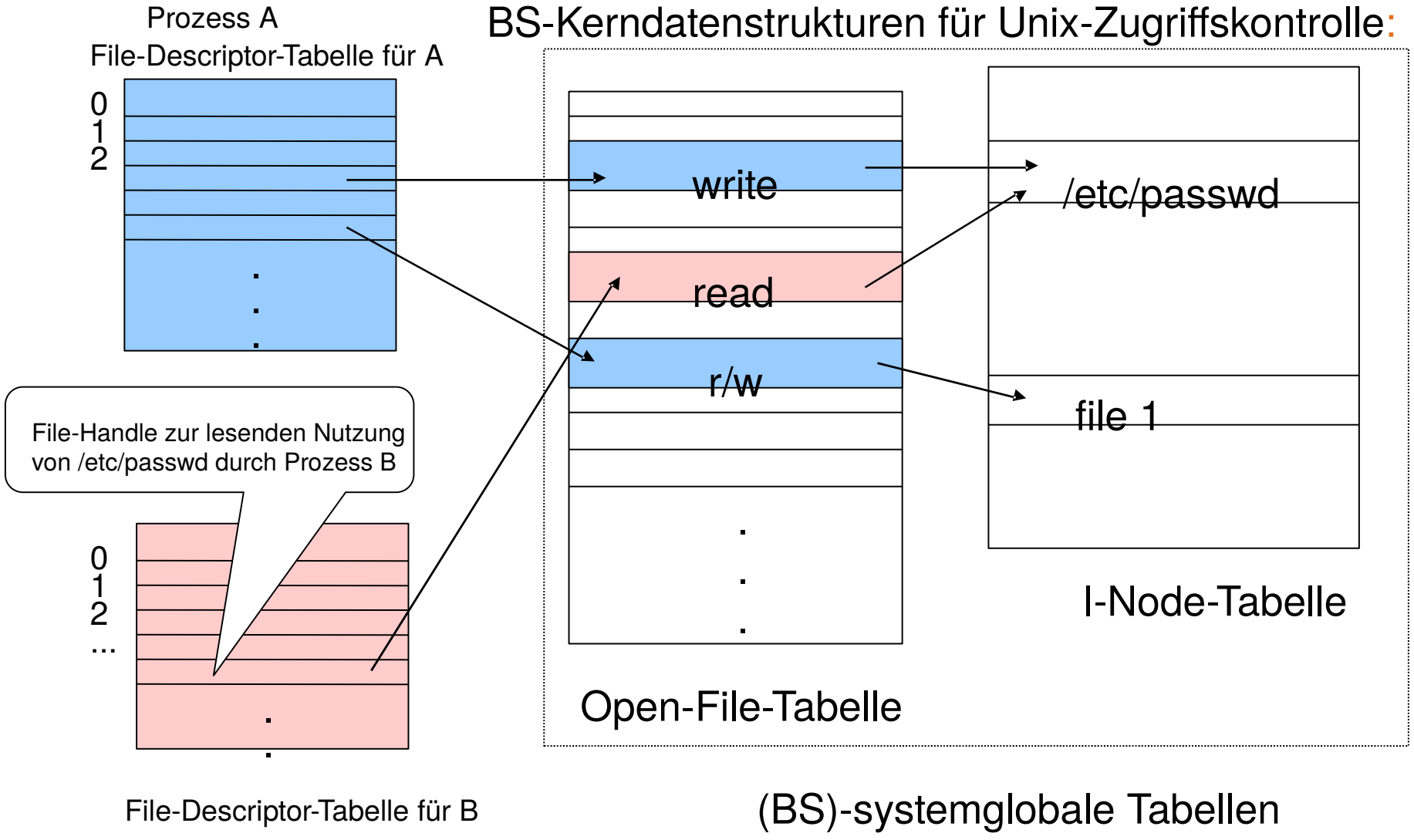
(2) Prüfen, ob zugreifender Prozess gemäß der ACL der Datei zum gewünschten Zugriff r, w, x berechtigt ist

(3) Falls o.k., return File-Handle:enthält Information über zulässige Zugriffsrecht r,w,x

- Eintrag mit Rechten in Open File Tabelle des Kernels
- Verweis in File-Descriptor-Tabelle des Prozesses auf Recht

(4) Zugriffe auf geöffnete Datei <datei_i> mit File-Handle Dateisystem führt Zulässigkeitskontrolle durch.

Vorteile/Nachteile dieser Vorgehensweise?



Sicherheitsprotokolle