

# Sicherheits- mechanismen

In Betriebssystemen

Hermann Härtig  
TU Dresden



# Wegweiser

## Wiederholung: Grundlagen

- Begriffe
- **Was** soll **wovor** geschützt werden (Schutzziele, Angriffsmodelle)

## Konstruktion sicherer Systeme

- Prinzipien
- Sicherheit und Systemarchitektur

# Begriffe

- Sicherheit (**Safety**):  
Schutz von Menschen vor Fehlern, Störungen, Ausfällen,  
bei Katastrophen
- Sicherheit (**Security**):  
Schutz von Menschen und Rechnern vor intendierten Fehlern:  
Angriffen
- Fehlertoleranz (Fault Tolerance):  
(Selbst-) Schutz von Systemen bei Ausfällen und Entwurfsfehlern
- Verlässlichkeit (Dependability, Robustness):  
Oberbegriff: Sicherheit und Fehlertoleranz
- Protection (mechanism):  
Schutz (-mechanismen)
- Security Policies („Sicherheitsstrategien“):  
definieren die Schutzziele für (eine Menge von) Anwendungen

# Schutzziele

Benutzer (Menschen/Organisationen/ ... )  
definieren Sicherheitsstrategien in Bezug auf die Schutzziele:

- Vertraulichkeit (confidentiality)  
Schutz vor unbefugter Weitergabe von Information
- Integrität (integrity)  
Schutz vor unbemerkt, unbefugter Manipulation
- Wiederherstellbarkeit (Recoverability)  
Schutz vor vollständigem Verlust
- Verfügbarkeit (availability)  
Zeitliche Zusagen für den Zugang zu einer Leistung  
quantitative Größe
- Verbindlichkeit
- Anonymität
- ...

# Schutzziele

## Integrität

- entweder Information ist aktuell, intakt und vollständig  
oder
- es ist möglich zu entdecken, dass diese Eigenschaften nicht erfüllt sind

→ wichtig:

Integrität vs. Wiederherstellbarkeit vs. Verfügbarkeit

z.B.: mittels kryptographischer Verfahren kann man die Integrität von Botschaften in einem offenen Netz gewährleisten, nicht aber vor deren Verlust schützen

# Horaz zu Vertraulichkeit

Et semel emissum volat irrevocabile verbum.

Horaz [Episteln I 18,71]

Frei übersetzt:

Und einmal hinausgeflogen ist das Wort nicht wieder einzufangen.

# Angriffe und Aufwand

Effektiver Schutz erfordert klare Vorstellung über Angriffe und deren Aufwand, die man abwehren können möchte.

Kosten und Nutzen für Schutzmaßnahmen müssen in sinnvoller Relation stehen.

Cormac Herley: skalierbare und nicht-skalierbare Angreifer

Schutzziele sind manchmal (zu Recht) anderen Erwägungen untergeordnet.

# Wegweiser

## Wiederholung: Grundlagen

- Begriffe
- **Was** soll **wovor** geschützt werden (Schutzziele, Angriffsmodelle)

## Konstruktion sicherer Systeme

- Prinzipien
- Sicherheit und Systemarchitektur
- Ein paar Mechanismen

# Konstruktion sicherer Systeme

## Entwurfsprinzipien – SALTZER und SCHRÖDER (schon 1975!)

- Prinzip der geringst-möglichen Privilegisierung  
nur die Rechte einräumen, die für die zu erbringende Funktionalität erforderlich sind
  - Verbot als Normalfall
  - Whitelisting vs. BlacklistingGegenbeispiel: ambiente Autorität in POSIX
- Sichere Standardeinstellungen („fail-safe defaults“)
- Separierung von Privilegien (Separation of duty)  
mehrfache Bedingungen für die Zulassung einer Operation

# Konstruktion sicherer Systeme

- So einfach wie möglich

reduziert Fehlermöglichkeiten

- bei deren Implementierung
- bei deren Einsatz

- Offenheit

Sicherheit darf nicht auf der Geheimhaltung von Entwurf und Implementierung basieren („Security by Obscurity“)

- Psychologisch und praktisch akzeptabel

# Konstruktion sicherer Systeme

- Vollständige Kontrolle (complete mediation):  
jeder Aktion, die potentiell ein Schutzziel verletzt, sollte kontrolliert werden  
  
Gegenbeispiel: Zugriff auf offene Unix Dateien
- Aktualität von Prüfungen  
TOCTTOU: time of check to time of use

# Sicherheit und Systemebenen

Umgebung:	physische Sicherung
HW:	Adressraumseparierung und User/Kernel-Modi
BS-Basis:	Kapselung von Prozessen
Mechanismen: (lokal)	Authentifikation, ACLs und Capabilities
Benutzer/Administration:	Einsatz der Mechanismen

# Authentifikation von Benutzern

## Überprüfung der Identität eines Benutzers

- Passwörter
  - ◊ unsicher
  - ◊ Standardweg für Einbrüche
- mögliche Verbesserungen
  - ◊ Zurückweisung bestimmter Passwörter
    - mindestens ein Sonderzeichen, keine Eigennamen, ...
  - ◊ System generiert Passwörter vor („access token“)
  - ◊ Festlegung und Durchsetzung einer maximalen Gültigkeitsdauer
  - ◊ nur einmal benutzbare Passwörter

# Weitere Authentifikationstechniken

- Chipkarten (Smart Cards)
  - ♦ Überprüfung der Identität der Chipkarte
  - ♦ Überprüfung der Identität des Besitzers durch „PIN“
- Überprüfung physischer Merkmale (Biometrie)
  - ♦ Fingerabdrücke
  - ♦ Stimm-Analyse
  - ♦ Unterschriftenprüfung  
(Druckschwankungen, Bewegungsablauf)

# Sicherheitsakteure

## Benutzer:

- erzeugen Prozesse und lassen sie für sich arbeiten
- speichern ihre längerfristigen Daten in Dateien/Dokumenten/Speicherobjekten
- steuern die Zugriffsrechte auf Objekte

## Entwickler:

- schreiben Anwendungen („Apps“), welche Benutzer installieren
- wissen welche Ressourcen ihre Anwendung im Normalbetrieb benötigt
- jede App ist ein potentiellles Sicherheitsrisiko

# Kapselung der Adressräume von Prozessen

- sprachbasiert:

alles in einer sicheren HLL

Beispiele:

Burroughs 1700, 5500, (1975 ff)

Microsoft Singularity

Oberon

- hardwarebasiert:

Kernel/User-Mode und Adressräume

Beispiele:

Windows, Linux, L4, ...

# Prozesse und ihre Interaktion

Grundsätzliche Annahme:

Prozesse versuchen nicht erlaubte Operation durchzuführen

- Dateien lesen/schreiben
- Ressourcen benutzen (z.B. Netzwerk)
- Informationen austauschen

Fragestellungen:

- Wie legt man Rechte fest?
- Wie setzt man deren Einhaltung durch?

# Subjekte und Objekte

Operation(Subjekt, Objekt) zulässig?

## **Subjekte:**

Prozesse

## **Objekte:**

Dateien, Geräte,  
Prozesse, Kommunikationskanäle,  
physischer Speicher, virtueller Speicher;  
Bankkonten, ...

## **Operationen:**

Lesen, Schreiben, Löschen, Ausführen;  
Einzahlen, Abheben, ...

# Schutzmatrix

- Subjekte und Objekt

	Objekte		
Subjekte			
		Rechte	

**Operation(Subjekt, Objekt) zulässig?**

# Ausprägungen der Schutzmatrix

spaltenweise: **ACL – Access Control List** (Zugriffssteuerliste)

- bei jedem Zugriff wird beim Objekt auf der Basis der Identität des Absenders dessen Berechtigung geprüft

zeilenweise: **Capabilities** (deutsch ???)

- bei jedem Zugriff wird etwas geprüft, was Subjekte besitzen und bei Bedarf weitergeben können

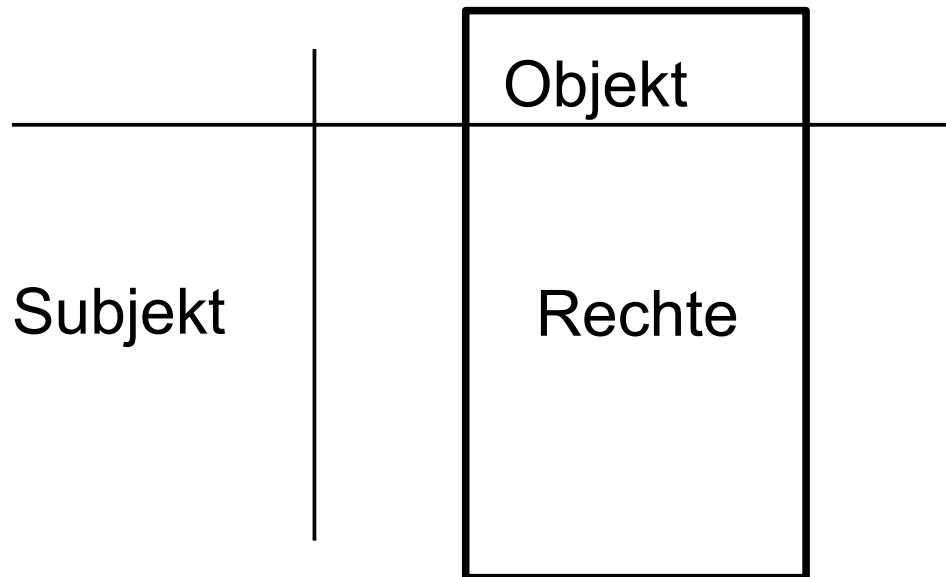
regelbasiert („mandatory access control“):

- bei jedem Zugriff werden Regeln ausgewertet

# Schutzmatrix spaltenweise: ACL

## Spaltenweise Darstellung: Access Control Lists

- Festlegung für jedes Objekt, „was welches Subjekt damit tun darf“



# Ein einfaches Modell: Datei- und Prozess-Attribute



- Festlegungen in Bezug auf Benutzer:
  - für welchen Benutzer arbeitet ein Prozess
  - welchem Benutzer gehört eine Datei (owner)
  - welche Rechte räumt ein Benutzer anderen und sich selbst an „seiner“ Datei ein
    - Rechte eines Prozesses an einer Datei
  - Attribute von Prozessen:        UserId
  - Attribute von Dateien:        OwnerId

Schutzmatrix :

	File 1	File 2	File 3	File 4	File 5	File 6	File 7	File 8
User1								
User2								
User3			read					
User4								

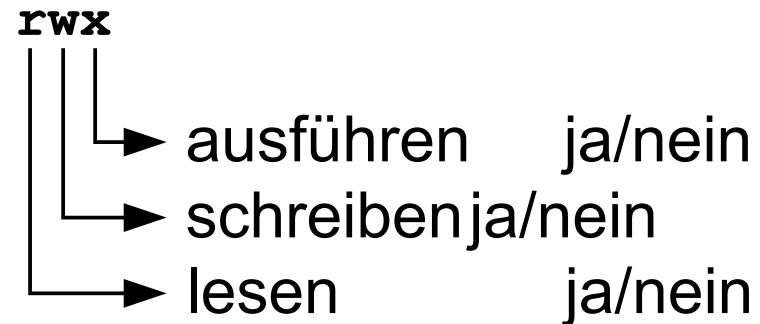
# ACLs in Unix

Unix: rudimentäre Zugriffssteuerlisten

- Prozess: UserId, GroupId
- Datei: Owner, Group
- Rechte in Bezug auf Owner, Group, Others

Rangliste.dat		
<b>rw-</b>	<b>r--</b>	<b>---</b>
		Others
	Group:	Schach
	Owner:	Heini

Dateiattribute:



# Rechte-Änderung per SETUID: Unix

## Beispiel

- Rangliste : */usr/henrike/spiele/schach/Rangliste*
- Programm: */usr/henrike/bin/spiele/Schach*
- jeder Spieler soll seine Ergebnisse selbst in die Rangliste eintragen können

### 1. Erster Versuch:

alle haben Schreibrecht → zu viele Rechte  
(funktioniert nicht)

### 2. SetUID:

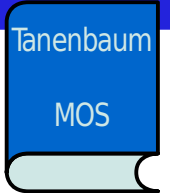
nur Henrike hat Schreibrecht;

Schachprogramm: „setuid“:

sobald ein Prozess das Schachprogramm aufruft,  
erhält es als UserId den Owner des Schachprogramms

→ Praktische Erfahrung: immer noch zu viele Rechte !!!

# Access Control Lists (Zugriffssteuerlisten)



Setzen der ACLs darf,

- wer einen ACL Eintrag für dieses Recht hat
- Erzeuger

## Multics

File 0 (Jens, \*, RWX)

File 1 (Jens, system, RWX)

File 2 (Jens, \*, RW-), (Else, staff, R --), (Meike, \*, RW-)

File 3 (\*, student, R--)

File 4 (Paul, \*, ---), (\*, student, R--)

## Windows NT

Objekt: allow, deny

full control, modify, read&execute, ...

# ACL und Schutz vor trojanischen Pferden

## App-Sandbox in Android

- ♦ erzeuge eine neue Benutzerkennung
- ♦ füge neuen Benutzer in genau die ACLs ein, die für das Programm wirklich gebraucht werden
- ♦ starte das Programm mit der Nutzerkennung des neuen Nutzers

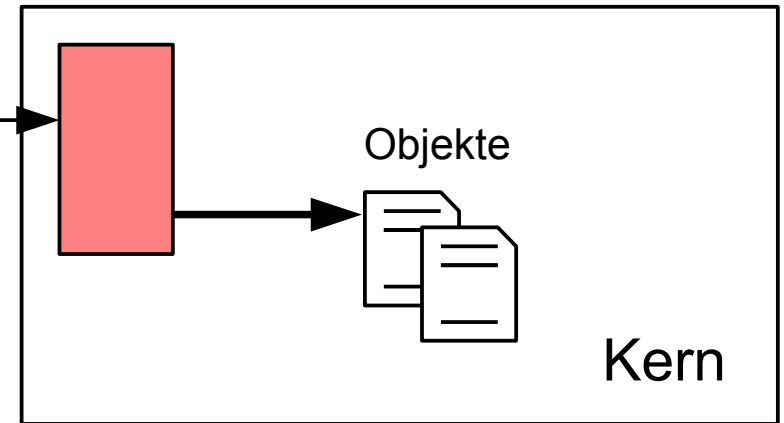
# Durchsetzung von ACLs

## Kapselung der ACLs

1. durch eine vertrauenswürdige Einheit, an der man nicht vorbeigehen kann:

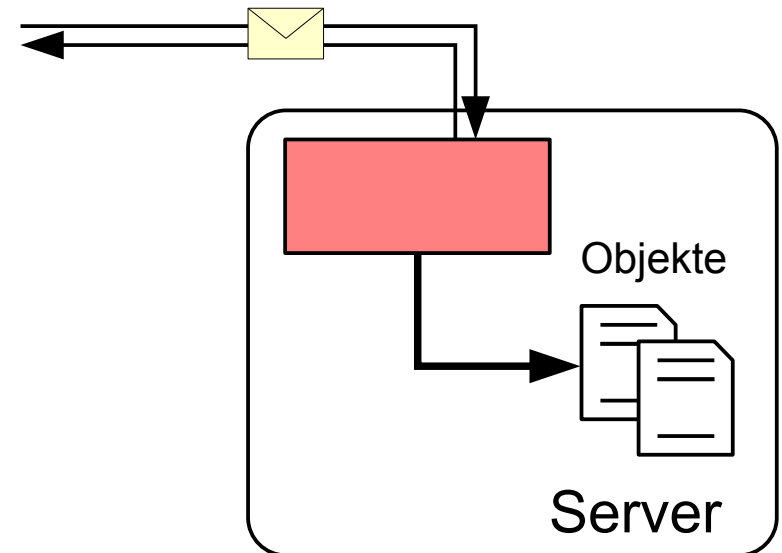
- Unix: der Kern
- WinNT: „Security Manager“

`open ( )`



2. durch die Server, die Objekte implementieren

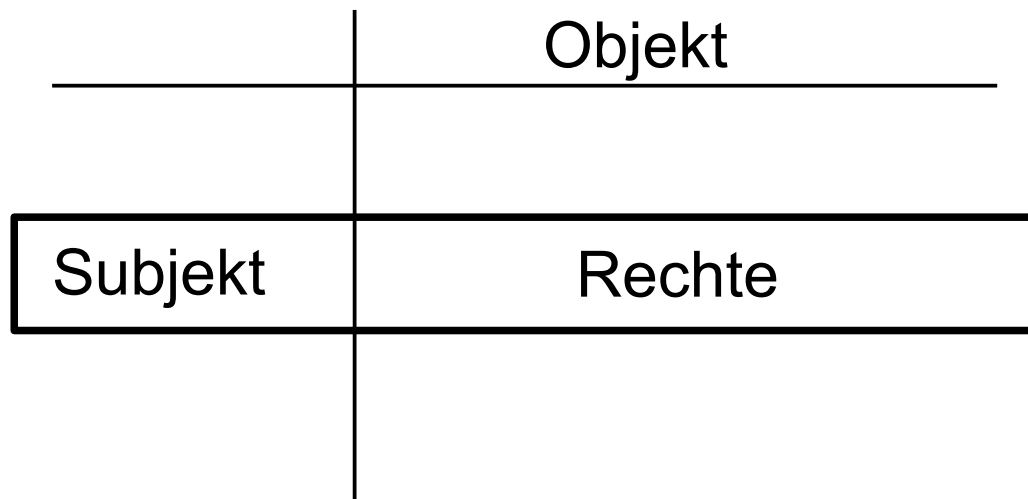
Botschaften



# Schutzmatrix zeilenweise: Capability

## Zeilenweise Darstellung: Capability

- Festlegung für jedes Subjekt, „wie es auf welche Objekte zugreifen darf“



# Capabilities

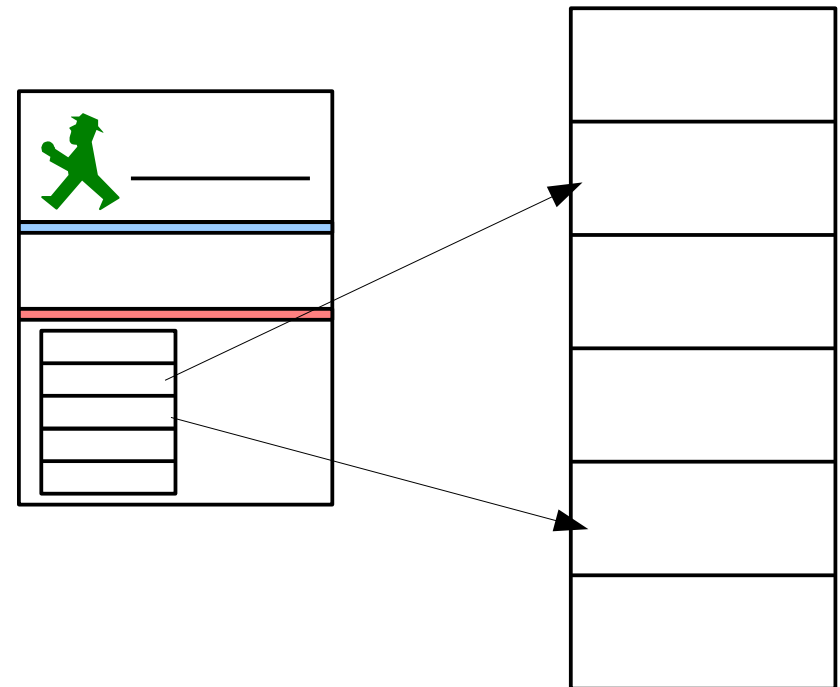
Prozesse:

- array von Capabilities
- Zugriffs auf Objekte durch Index in array

Capability:

- Pointer auf Objekt
- Rechte

Prozesse können capabilities weitergeben



# Beispiel für Capabilities

- Rudimentäre Form: Unix Dateideskriptoren
- Weitergabe nur durch „FORK“ Operation

Prozessleitblock

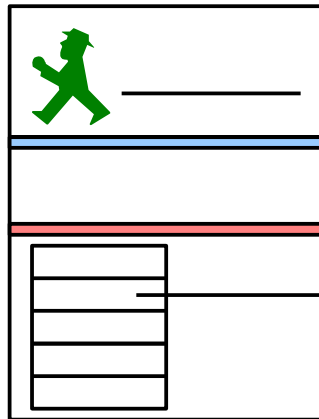
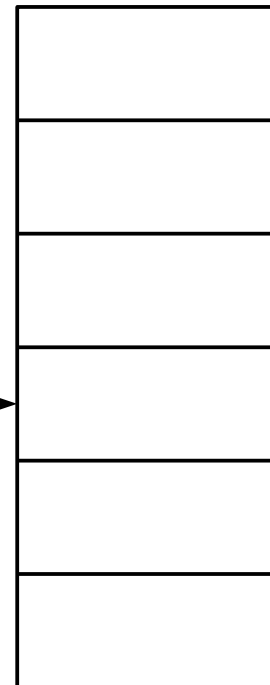


Tabelle offener  
Dateien



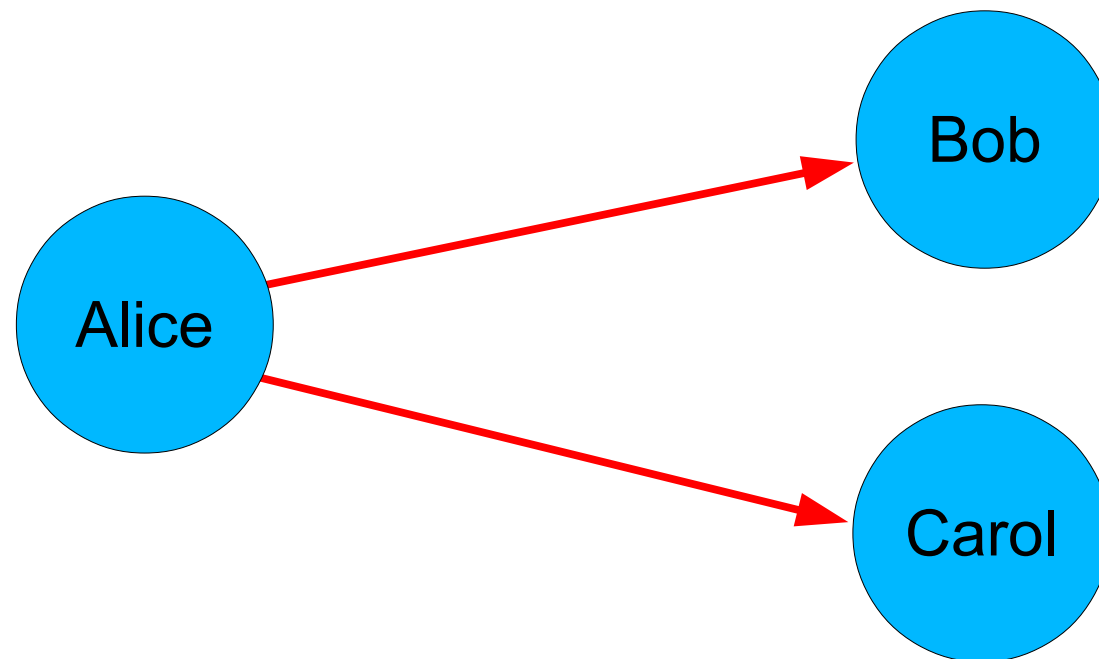
# Rechte-Änderungen bei Capabilities

Weitergabe durch Botschaften

Entzug ??

z.B.: Aufgabe von Namensdiensten

Granovetter-Diagramm nach Mark Miller



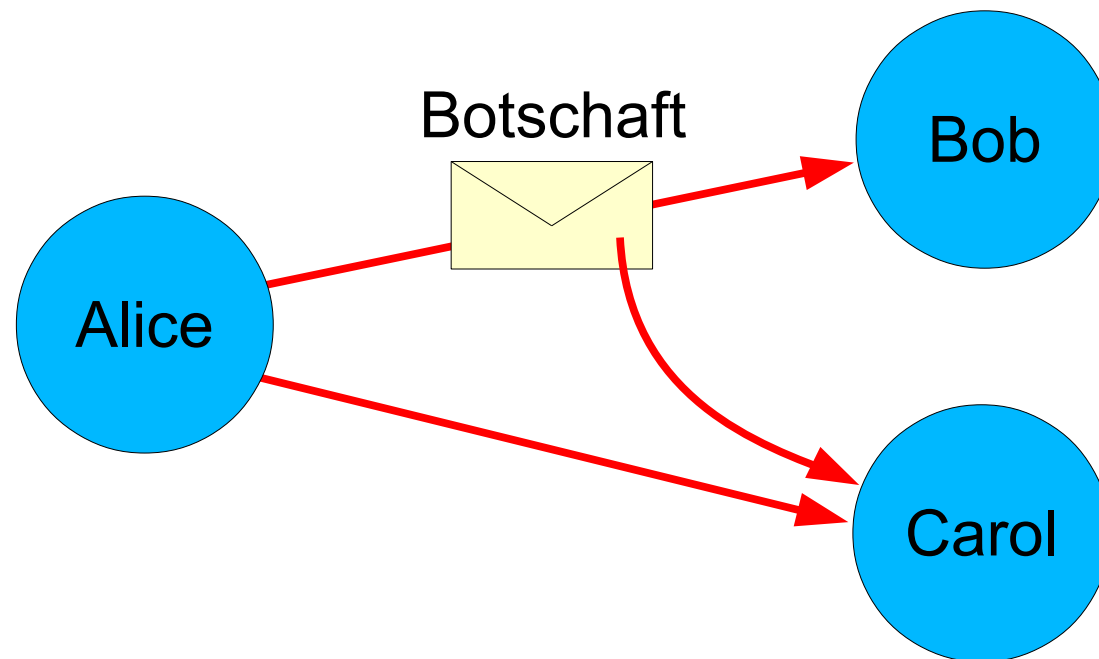
# Rechte-Änderungen bei Capabilities

Weitergabe durch Botschaften

Entzug ??

z.B.: Aufgabe von Namensdiensten

Granovetter-Diagramm nach Mark Miller



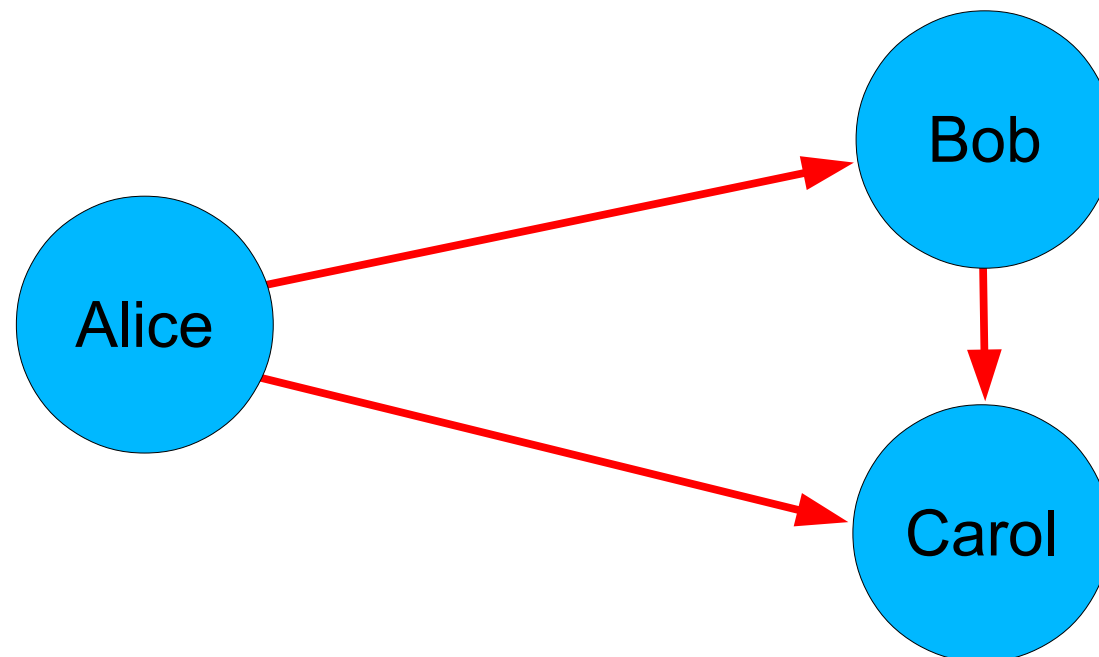
# Rechte-Änderungen bei Capabilities

Weitergabe durch Botschaften

Entzug ??

z.B.: Aufgabe von Namensdiensten

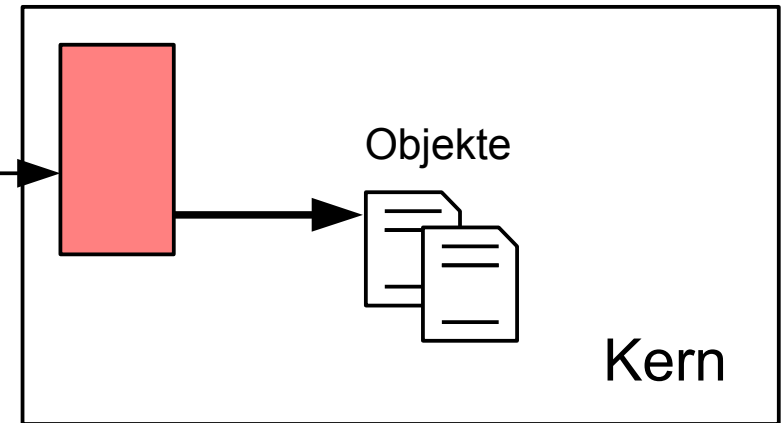
Granovetter-Diagramm nach Mark Miller



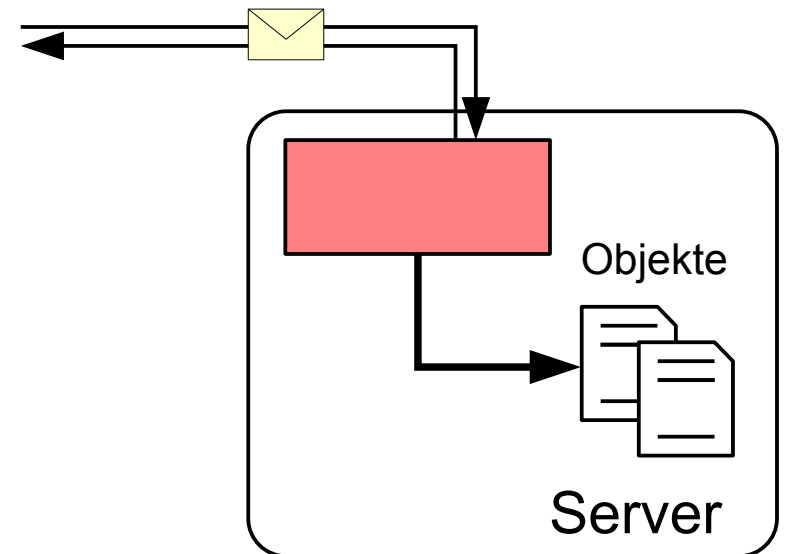
# Durchsetzung von Capabilities

## Kapselung der Capabilities

1. durch eine vertrauenswürdige Einheit, an der man nicht vorbeigehen kann:
  - Betriebssystem-Kern  
z.B. Keykos, Eros, L4
  - Hardware (Tags)  
z.B. Burroughs 5500, rudimentär
2. durch die Server, die Objekte implementieren, und kryptographische Verfahren  
z.B. Amoeba



## Botschaften

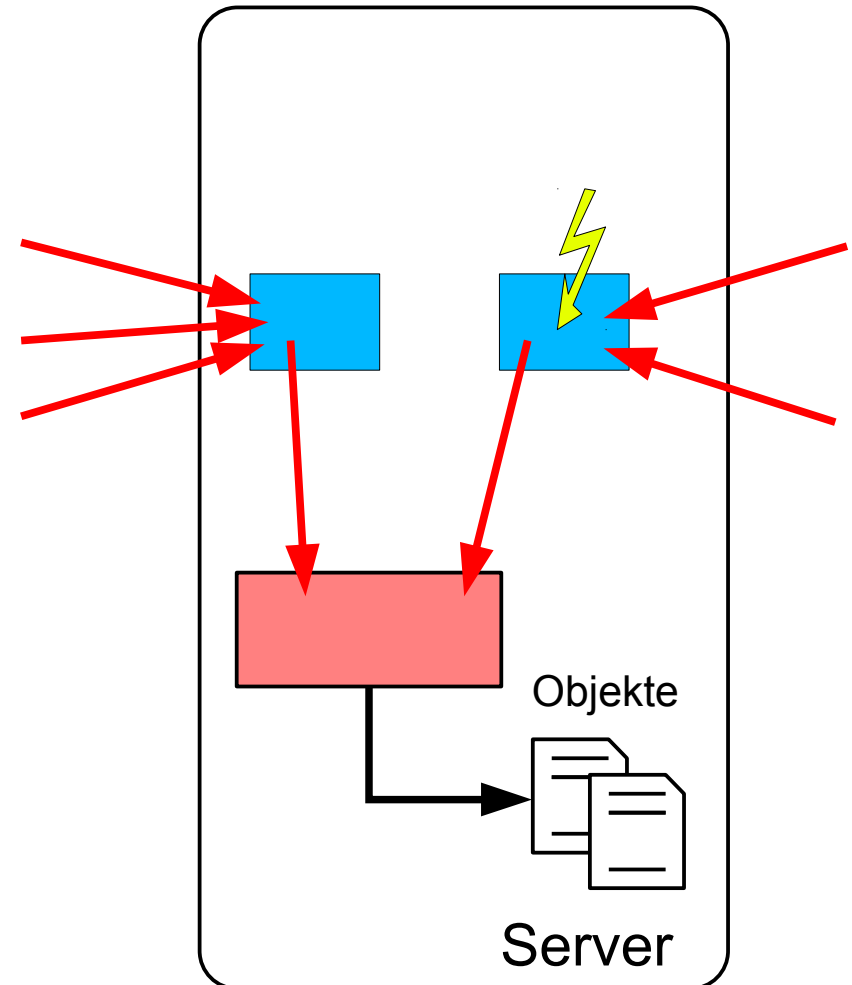


# Entzug von Capabilities

- durch Buchführung
- nur durch Indirektion  
EROS, Keykos, Amoeba

viele weitere Fragestellungen  
zu Capabilities

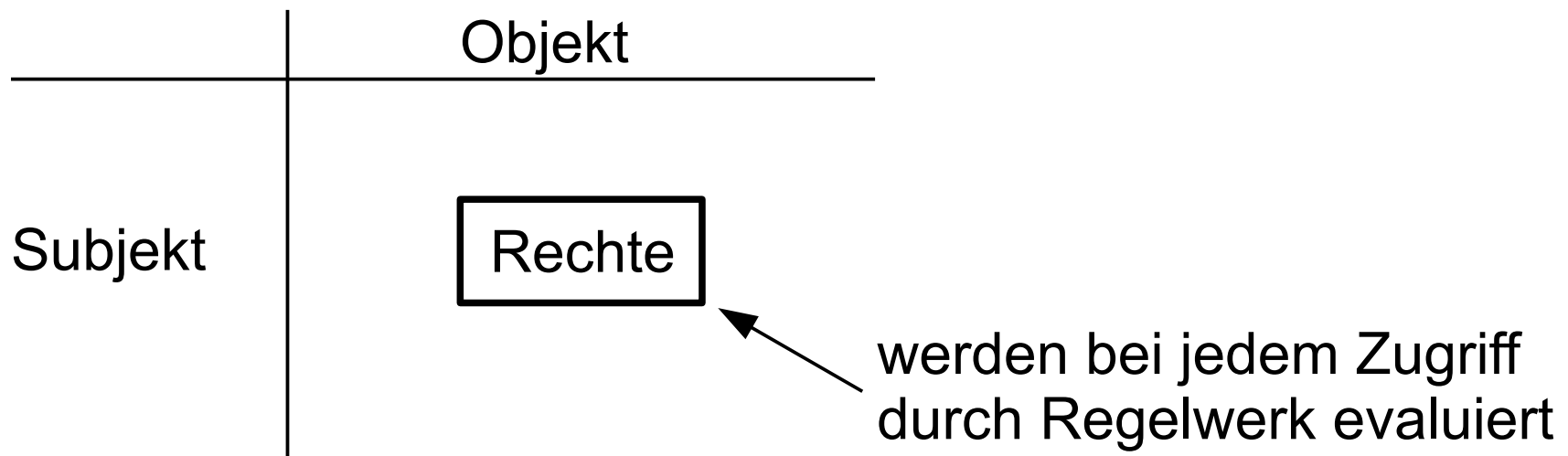
→ Distributed OS (Sommersem.)



# Schutzmatrix regelbasiert

## Mandatory Access Control (Regelbas. Zugriffssteuerung)

- Konzept:  
Subjekte und Objekte haben Attribute („labels“)  
Entscheidung über Zugriff anhand von Regeln
- Implementierung:  
sog. Sicherheitskerne



# Beispiel „Multilevel Security“ (Militär)



## Einstufung von Personen und Dokumenten

- in eine Sicherheitsebene  
z. B. {normal, vertraulich, geheim, streng geheim}
  - in eine oder mehrere Kategorien  
z.B. {Nato, Atom, Crypto}
- z. B. (geheim; Nato, Atom, Crypto)

## Begriffe

- Personen – clearance
- Dokumente – classification

# Multilevel Security



## Vorschrift 1: Sicherheitsebene (X,Y)

- Sicherheitsebenen sind geordnet
- X mindestens auf gleicher Sicherheitsebene-Ebene wie Y

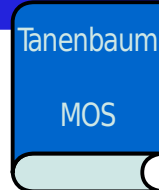
## Vorschrift 2: Kategorien (X,Y)

- Kategorien sind unabhängig voneinander
- X muss alle Kategorien haben, die auch Y hat

## Kombination von Sicherheitsebene und Kategorie:

Vorschrift 1(X,Y) und Vorschrift 2(X,Y)  $\rightarrow$  X „dominiert“ Y

# Beispiel



Dokument (geheim; Nato, Atom)

---

Person 1: (geheim; Nato, Atom, Crypto)

Person 1 dominiert Dokument, da

geheim  $\geq$  geheim

UND  $\{\text{Nato, Atom, Crypto}\} \supset \{\text{Nato, Atom}\}$

---

Person 2: (streng geheim; Nato, Crypto)

Person 2 dominiert Dokument nicht, da

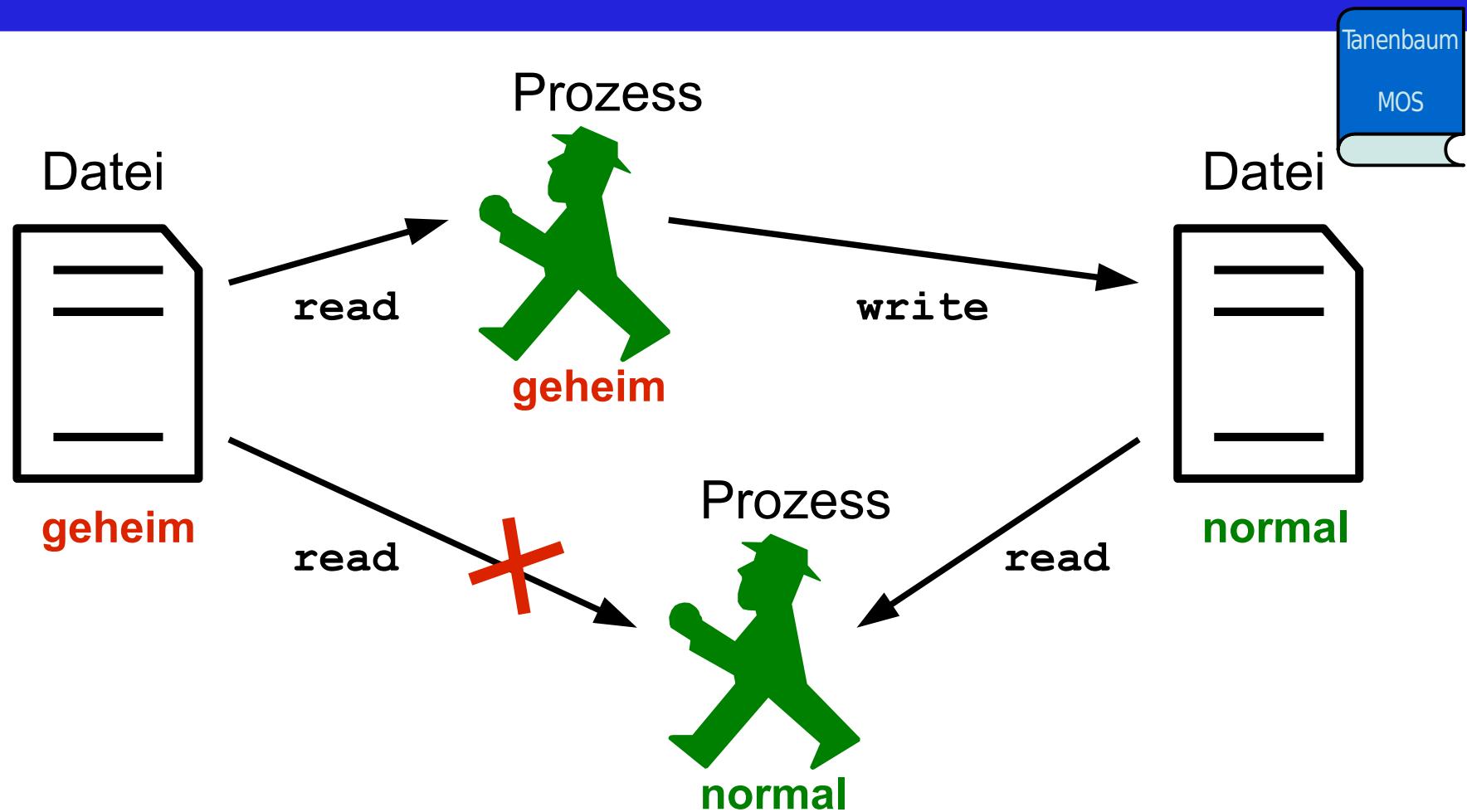
$\neg (\{\text{Nato, Crypto}\} \supset \{\text{Nato, Atom}\})$

# Regeln der Multilevel Security

## Regel 1: „Simple Security“

Ein Subjekt darf *lesend* auf ein Objekt nur dann zugreifen, wenn die Einstufung des Subjekts die des Objekts dominiert.

# Problem



aber: als geheim eingestufter Prozess kann Informationen an Datei weitergeben, die als normal eingestuft ist

→ zusätzliche Regel (\*-property)

# Regeln der Multilevel Security

Tanenbaum

MOS

## Regel 1: „Simple Security“

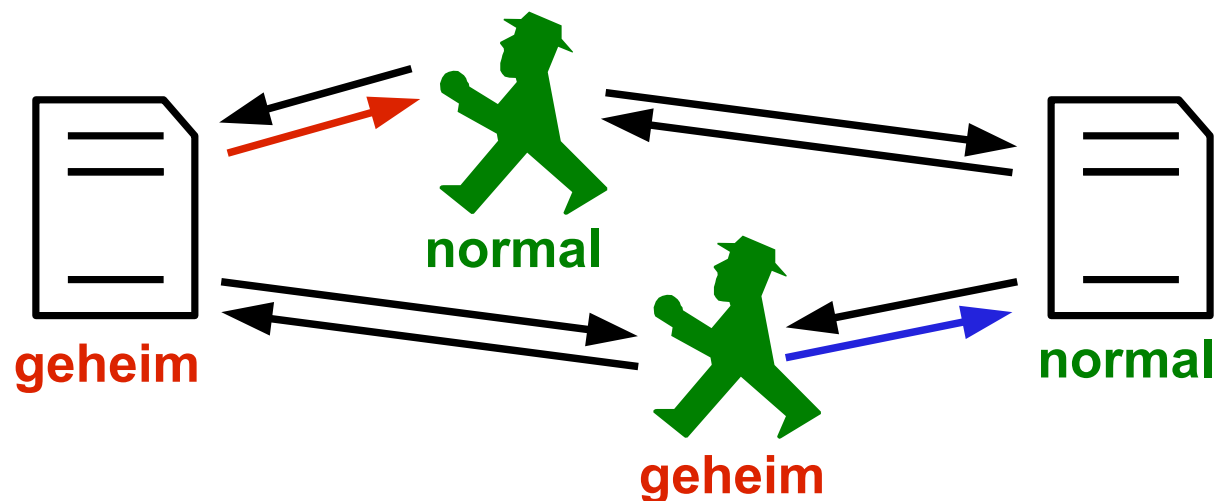
Ein Subjekt darf **lesend** auf ein Objekt nur dann zugreifen, wenn die Einstufung des Subjekts die des Objekts dominiert.

## Regel 2: „\*-Property, Confinement Property“

Ein Subjekt darf **schreibend** auf ein Objekt nur dann zugreifen, wenn die Einstufung des Subjekts von der des Objekts dominiert wird.

→ Simple Security Verletzung

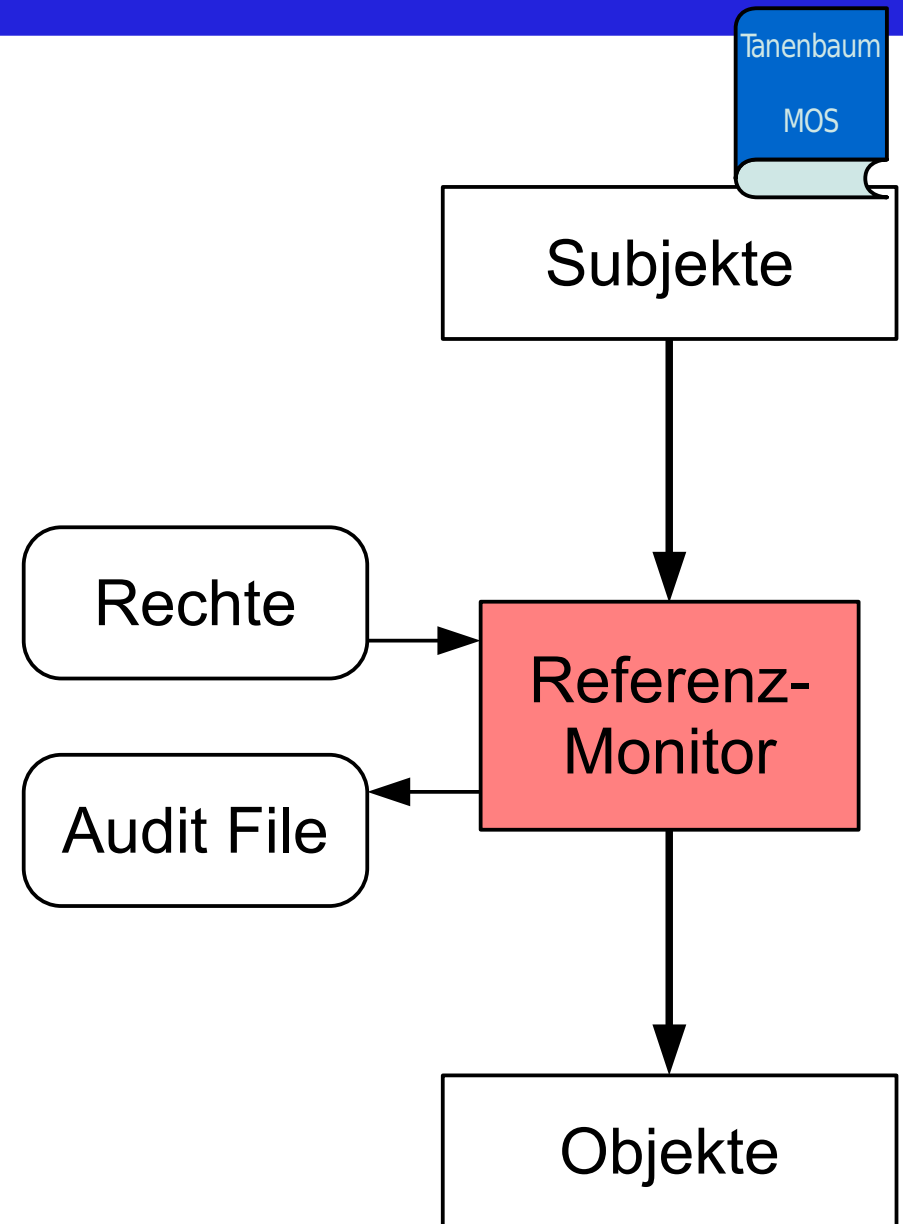
→ Confinement Property Verletzung



# Durchsetzung: Referenzmonitor/Sicherheitskern

## Prinzipien

- Vollständigkeit
  - kein Subjekt darf auf Objekt unter Umgehung des Referenzmonitors zugreifen
- Isolation
  - keine Modifikation des Referenzmonitors
- Verifizierbarkeit
  - Code Inspektion
  - Test
  - formale Beweise



# Moderne MAC-Implementierungen

- SELinux, AppArmor
  - komplexe Regelsätze, schwer zu überblicken
- TrustedBSD, Apple Seatbelt
  - Grundlage für die **App Sandbox** in iOS und OS X
- Windows 8 App Container

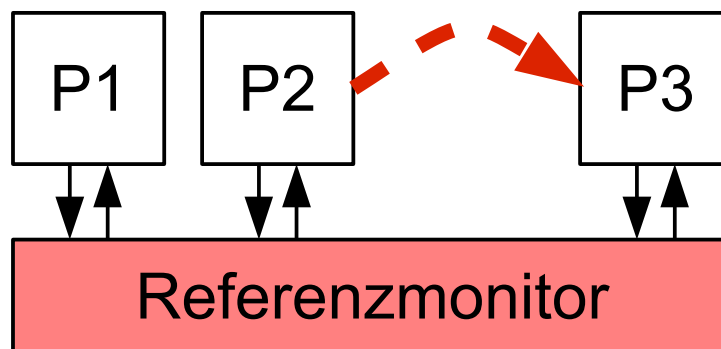
## Kombination von **Entwicklerwissen und Nutzerinteressen:**

- Entwickler beschreibt über **Manifest**, was die Anwendung im Normalfall benötigt
- System setzt diese Beschränkungen durch
- auch bei einem Angriff werden nicht mehr Rechte erlangt
- Nutzer kann durch **ACLs** seine Dokumente verwalten

# Aber: Verdeckte Kanäle (Covert channels)

Informationsfluss an veröffentlichten, durch Schutzmechanismen überwachten Schnittstellen vorbei

- also etwa Info von P2 an P3, obwohl von P2 nach P3 keine Info fließen darf



# Speicherbasierte verdeckte Kanäle



→ “Covert Storage Channels”

Notation für Beispiele im Folgenden

→ ein Prozess sendet,

← ein anderer Prozess empfängt

## Beispiele

- Dateinamen und Attribute
  - neuer Dateiname mit n Zeichen
  - ← Lesen des Verzeichnisses
- Bandbreite: n Zeichen (pro Op)
- einfach zu eliminieren (keine Leserechte für Verzeichnis)

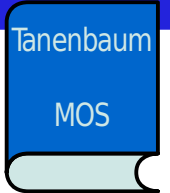
# Speicherbasierte verdeckte Kanäle



- Dateiexistenz
  - erzeuge Datei
  - ← Test ob vorhanden
- Bandbreite: hoch
- Eliminierung: keine gemeinsamen Verzeichnisse

z. B.: /tmp/...

bla1i bla2c bla2h bla3b bla3i bla3n bla3v bla4e bla4r  
bla5d bla5e bla6c bla6k bla6t ...



→ “Covert Timing Channels”

## Beispiele

- Anteil an CPU für einen Prozess
  - rechenintensiv/nicht rechenintensiv im Sekundenabstand
  - ← Beobachtung, wie hoch der eigene Anteil
- Bandbreite: ca. 1 Bit pro Sec
  - Voraussetzung: Zeitmessung
- Gegenmaßnahme:
  - Ausschaltung des Zugriffs auf Uhren inklusive timeouts !
  - absichtliche Ungenauigkeit von Uhren

# Weitere Beispiele

- Steganografie nutzt verdeckte Kanäle
- Stromverbrauch
- Sensoren und physikalische Kanäle

# Zusammenfassung

- Schutzziele
- Konstruktionsprinzipien für sichere Systeme
- Schutzmatrix
- ACLs
- Capabilities
- Regelbasierte Zugriffssteuerung
- Verdeckte Kanäle