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.

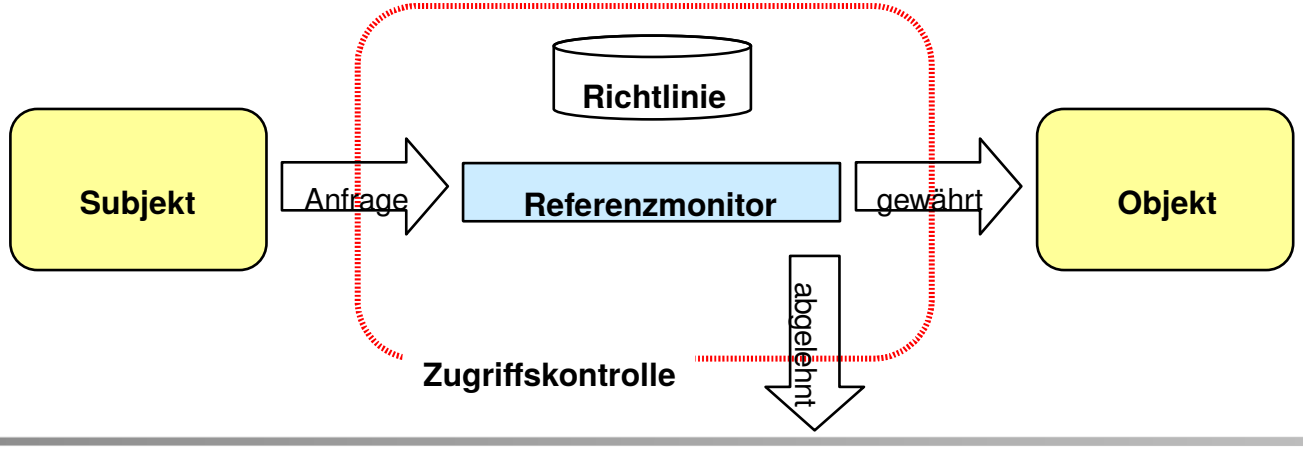
E-Book: http://www.ub.tu-dortmund.de/katalog/titel/1362263

Agenda

- Zugriffsschutz
  - Grundlagen
  - Rechtesysteme
  - Informationsflüsse

Zugriffsschutz

- Welches Subjekt darf auf welchem Objekt welche Aktionen ausführen?
  - Rechtesystem
  - Isolierung
  - Informationsflusskontrolle



Rechtesysteme

Zugriffsmatrix-Modell

Zugriffsmatrix-Modell (ZM) (Access-Matrix-Model)

Einfaches Modell zur Beschreibung der Zugriffsrechte

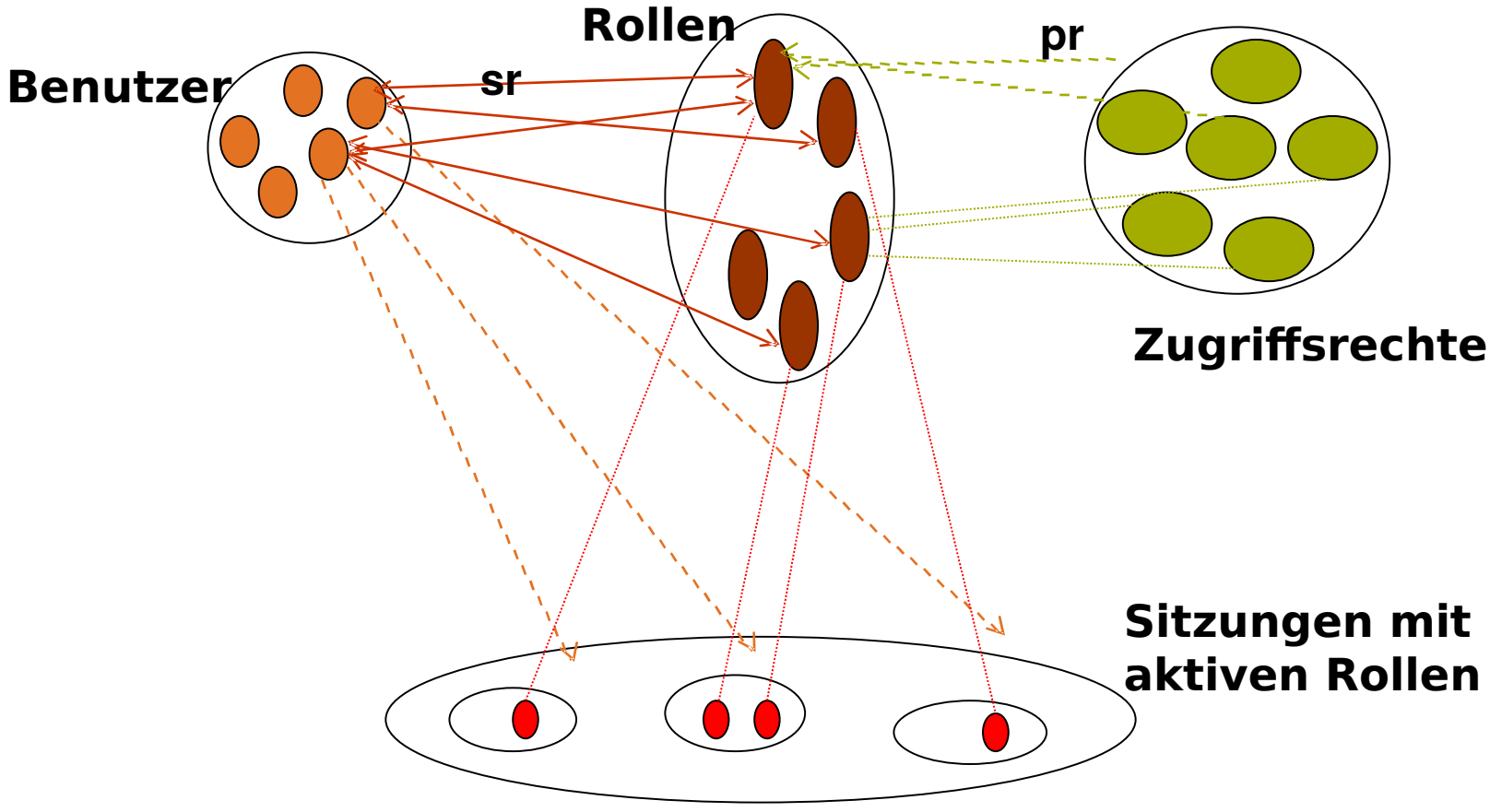
- (Dynamische) Menge von Objekten O_t
- (Dynamische) Menge von Subjekten S_t mit: $S_t \subseteq O_t$
- Menge von Rechten R
- Zugriffsmatrix $M_t : S_t \times O_t \rightarrow 2^R$, Schutz-Zustand zur Zeit t

M_t	Datei1	Datei2	Datei3	Prozess1	Prozess2
Prozess1	{ read, write }		{ read, write }		{ send, receive }
Prozess2				{ send, receive }	
Prozess3		{ owner, execute }		{ signal }	

Role-based Access-Control

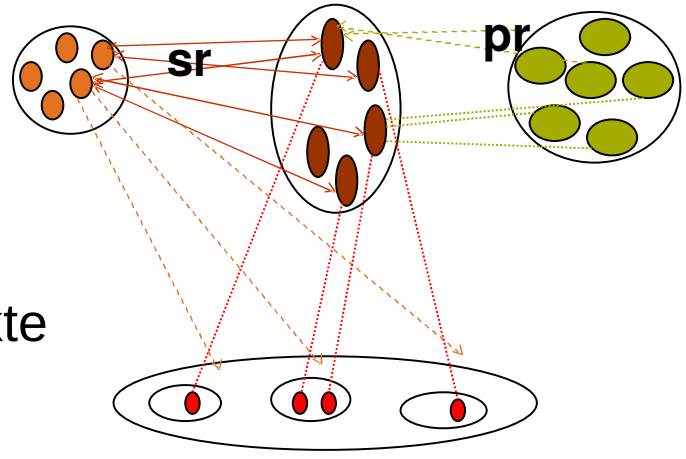
Role-based Access-Control (RBAC)

- 1992 Ferraiolo u. Kuhn, NIST RBAC Standard zusammen mit R. Sandhu http://csrc.nist.gov/rbac/sandhu-ferraiolo-kuhn-00.pdf
- **Ziel:** effektive, skalierende, effiziente Rechteverwaltung
- **Lösung:** Aufgabenorientierte Rechtevergabe durch Rollen
- Rolle: beschreibt eine Aufgabe bzw. mit damit verbundenen
  - Verantwortlichkeiten, Pflichten und Berechtigungen
- Nachbilden von Organisationsstrukturen
  - Rechte und Verantwortlichkeiten sind häufig direkt aus den Organigrammen der HR Abteilungen ableitbar
- Erfüllen der Prinzipien: need-to-know, separation-of-duty
- Weit verbreitet: u.a. Betriebssysteme, ERP, CMS, …


Benutzer
sr
Rollen
pr
Zugriffsrechte
Sitzungen mit
aktiven Rollen

Komponenten eines RBAC-Modells

Komponenten eines (einfachen) RBAC-Modells



- Menge von Subjekten = Benutzer
- Menge von Rollen *Role*, Rolle $r \in Role$
- Menge von Zugriffsrechten *P* (permission) für Objekte
- Zwei Abbildungen:

  (1) Benutzer-Rollenzuordnung $sr : S \rightarrow 2^{Role}$

  (2) Rechte-Rollenzuordnung $pr : Role \rightarrow 2^{P}$

- Sitzung $session \subseteq S \times 2^{Role}$, $(s,RL) \in session$, dann ist *RL* die Menge der aktiven Rollen des Benutzers *s*, $RL \subseteq sr(s)$
- $R_i \in session(s)$, falls $(s,RL) \in session \wedge R_i \in RL$

  D.h. *s* agiert in Rolle R_i , falls *s* Mitglied in der Rolle R_i ist u. diese Rolle in einer Sitzung aktiviert hat

Beispiel: Rollen

Beispiel: **Rollen** und deren Berechtigungen im Krankenhausszenario:

- Behandelnde Ärzte (auch AiP, PJ, zeitweise zugeordnete Ärzte):
  - ganze Patientenakte im Behandlungszusammenhang (außer besonders sensible Daten), (lesend, schreibend)
  - abteilungsinterne Daten aller Aufenthalte
- Pflegekräfte:
  - Zugriff auf Krankenakte; Umfang durch Abteilungsleiter festgelegt
- Famulanten, Studenten, Auszubildende:
  - erforderlicher Umfang durch verantwortlichen Lehrenden festgelegt (im Rahmen seiner eigenen Befugnisse).
- Verwaltungsmitarbeiter:
  - Stammdaten, (lesend, schreibend)
  - abrechnungsrelevante Daten (u. U. auch besonders sensible!)

Rollenhierarchien

Rollenhierarchien

Ziel: Vereinfachung von Verwaltungsaufgaben,

Nachbilden hierarchischer Organisationsstrukturen

- Definition einer partiellen Ordnung ≤ auf Rollen:

  $R_i, R_j \in Role$: falls $R_j \leq R_i$, dann besitzt R_i alle Rechte von R_j und ggf. noch zusätzliche Rechte

- **Beispiel:** Software-Entwickler ≤ Projekt-Leiter

  Rechte des Entwicklers: w,r,x, auf Projekt-Dateien

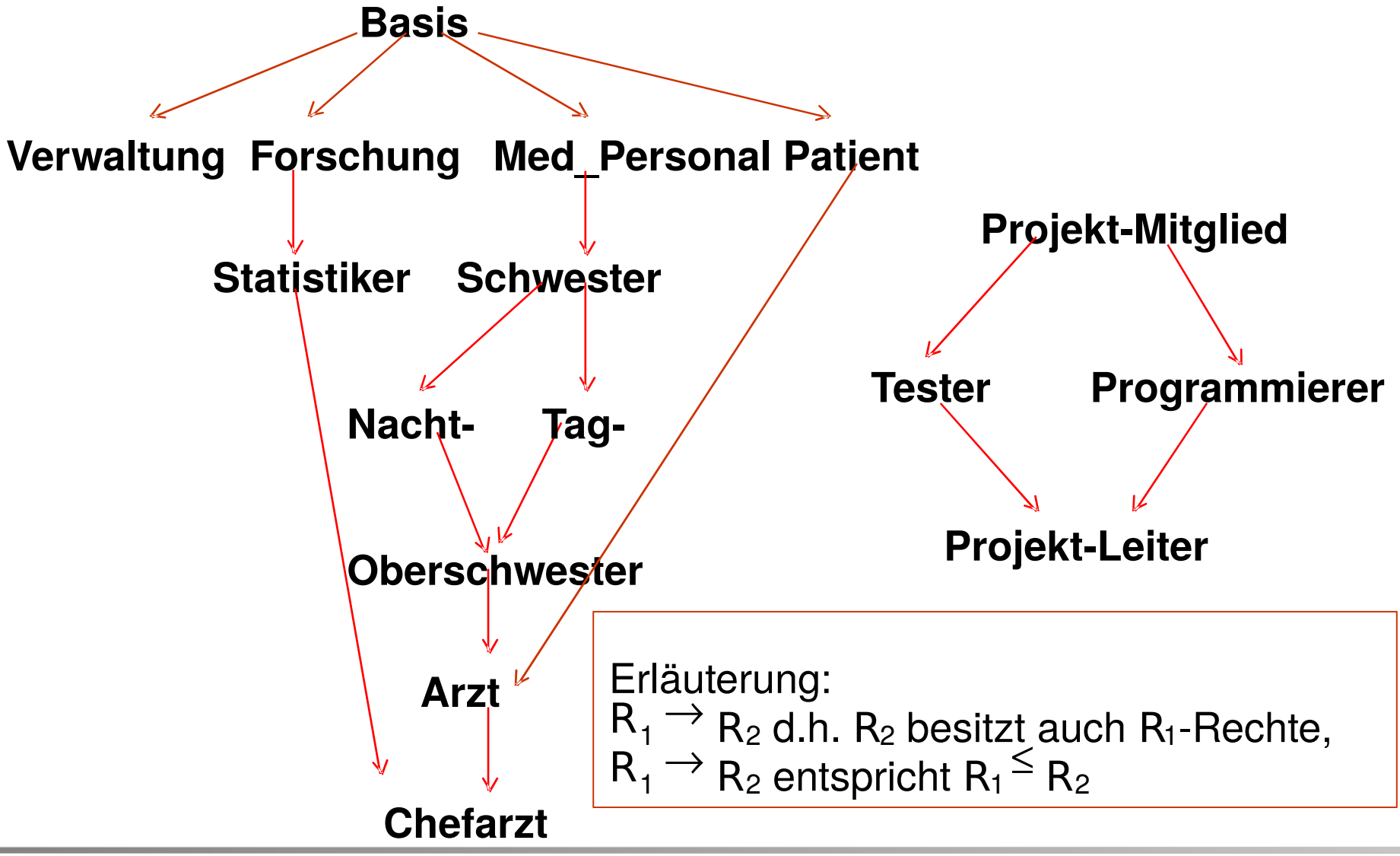
  Rechte des Leiters: w,r,x, auf Projekt-Dateien und

  w,r,x auf Projekt-Budget-Dateien, etc.

- Vererbung der Rollenmitgliedschaft: falls $R_j \leq R_i$, dann gilt: $\forall s \in S : R_i \in sr(s) \Rightarrow R_j \in sr(s)$

- Rechtevererbung ist aber nicht unproblematisch! Wieso?

Beispiele für Rollen-Hierarchien



Erläuterung:
$R_1 \rightarrow R_2$ d.h. R_2 besitzt auch R_1-Rechte,
$R_1 \rightarrow R_2$ entspricht $R_1 \leq R_2$

Eigenschaften, deren Einhaltung zu kontrollieren ist!

- Ein Subjekt darf nur in solchen Rollen aktiv sein, in denen es Mitglied ist:

  $\forall s \in S : R_j \in session(s) \Rightarrow R_j \in sr(s)$

- Ein Subjekt besitzt nur die Rechte seiner aktiven Rollen:

- Gegeben Funktion $exec : S \times P \rightarrow Boolean$, mit

  $exec(s,p) = true \Leftrightarrow s$ ist zu p berechtigt

  $\forall s \in S : exec(s,p) \Rightarrow \exists R_i \in Role : R_i \in session(s) \wedge p \in pr(R_i)$

- Gegeben sei die partielle Ordnung ≤ über Rollen: dann gilt

  $\forall s \in S: exec(s,p) \Rightarrow \exists R_i \in Role, R_i \in session(s) \wedge$

  $(p \in pr(R_i) \vee \exists R_j \wedge R_j \leq R_i \wedge p \in pr(R_j))$

Statische und dynamische Beschränkungen

Ziel: Regeln, die die Rollenmitgliedschaft beschränken

(1) Statische Aufgabentrennung (separation of duty):

- Wechselseitiger Ausschluss von Rollenmitgliedschaften:

  Festlegen einer Relation ***SSD*** $\subseteq Role \times Role$, mit $(R_i, R_j) \in SSD \Leftrightarrow R_i$ und R_j sind wechselseitig ausgeschlossen

- Sei $Member(R_i) = \{s \mid s \in S \wedge R_i \in sr(s)\}$

  Beschränkungsregel: $\forall R_i, R_j \in Role, \forall s \in S$:

  $(s \in Member(R_i) \wedge s \in Member(R_j)) \Rightarrow (R_i, R_j) \notin SSD$

(2) Dynamische Aufgabentrennung:

Wechselseitiger Ausschluss von Rollenaktivitäten

- Festlegen einer Relation ***DSD*** $\subseteq$ *Role* × *Role,* mit

  $(R_i , R_j) \in DSD \Leftrightarrow R_i$ und R_j sind dynamisch w.A.

- Beschränkungsregel: Für jedes Subjekt *s* sei:

  $Active(s) = \{R_i \mid \exists\, RL \subseteq Role \wedge (s, RL) \in session \wedge R_i \in RL\}$,

  dann muss gelten:

  $(s \in Member(R_i) \wedge s \in Member(R_j) \wedge \{R_i , R_j\} \subseteq Active(s)) \Rightarrow (R_i , R_j) \notin DSD$

Informationsflüsse

Informationsfluss

- Datum h ist confidential
- Datum l ist public

Direkter Informationsfluss

```
l = h;
```

Indirekter Informationsfluss

```
If p(h) {
   l = 1;
} else {
   l = 0;
}
```

Bell-LaPadula-Modell

Bell-LaPadula-Modell (Bell, LaPadula 1973) (BLP)

Beim RBAC-Modell:

- keine Kontrolle von Informationsflüssen

Lösung: Multi-level Security (MLS), Labeling-Konzepte!

- Ausprägung: BLP-Modell: erstes formalisiertes Modell

Modell (Auszug der wichtigsten Aspekte)

- Subjecte S und Objecte O
- Zugriffsrechte R = {read-only, append, execute, read-write, control}
- Menge von Sicherheitsklassen SC, $X \in SC$ mit $X = (A,B)$,

  A ist **Sicherheitsmarke** (label), z.B. *vertraulich*, *geheim*

  B Menge von **Kategorien** (compartments), z.B. *Arzt*
- $\forall\, s \in S$: Clearance: $SC(s) \in SC$; Maximale $SC(s)_{MAX}$ und aktuelle $SC(s)_{AKT}$
- $\forall\, o \in O$: Classification $SC(o) \in SC$
- Partielle Ordnung auf SC: $(SC, \leq)$, für $X, Y \in SC$ gilt:

  $X = (A,B)$, $Y = (A', B')$ $\quad X \leq Y \Leftrightarrow A \leq A' \wedge B \subseteq B'$

Beispiel: MAC-Policy (mandatory access control), systembestimmt

- **Regeln:** no-read-up, no-write-down, tranquility

Intuitive Interpretation der Regeln:

- Festlegen einer part. Ordnung über Sicherheitsmarken
- Informationsflüsse sind nur von unten nach oben (entlang der Ordnung) zulässig

Simple-Security-Property (no-read-up-Regel):

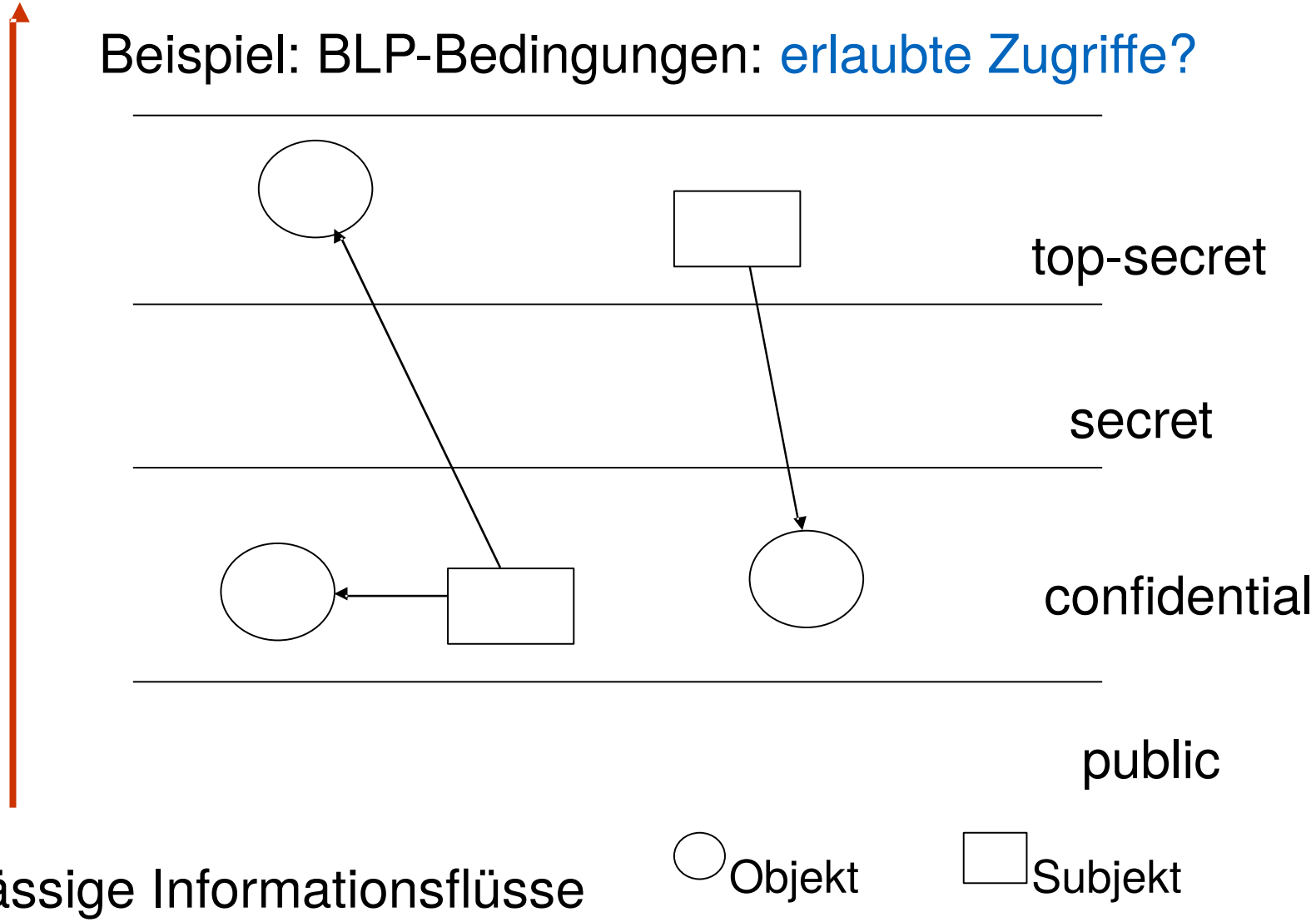
Für $z \in$ {read – only, execute} : $z \in Mt(s, o) \wedge SC(s) \geq SC(o)$

*-Property (no-write-down-Regel):

- append $\in Mt(s, o) \wedge SC(s) \leq SC(o)$; und
- read-write $\in Mt(s, o) \wedge SC(s) = SC(o)$

Ggf. Strong-Tranquility-Regel: während der Systemlaufzeit keine Änderung der Subjekt-Clearance oder der Objekt-Classification

Beispiel: BLP-Bedingungen: erlaubte Zugriffe?



Zugriffsoperation	Beschränkung: Subjekt S / Objekt O
read (file)	$SC(S) \geq SC(O)$
exec (file)	$SC(S) \geq SC(O)$
write (file)	$SC(S) = SC(O)$
append (file)	$SC(S) \leq SC(O)$
	z.B. no read up
read (directory)	$SC(S) \geq SC(O)$
search (directory)	$SC(S) \geq SC(O)$
create (directory)	$SC(S) = SC(O)$
read (signal/ipc)	$SC(S) \geq SC(O)$
kill (signal/ipc)	$SC(S) = SC(O)$

Beispiel:

BLP-Policy für Unix-Variante

Erweiterungen gegenüber Unix

- des Login: Angabe von *SC*(*s*)
- der Prozessbeschreibung:

  Prozesskontrollblock mit *SC*(*s*)
- der Dateibeschreibung:

  i-node mit *SC*(*file*)

Grenzen des BLP-Modells

Grenzen des BLP-Modells: u.a.

(1) Problem: Information/Objekte werden sukzessive immer höher eingestuft, warum tritt das Phänomen auf?

- **Lösung:** Einführung von Trusted Subjects (Multi-level Subjekte) (u.a. Systemprozesse)
- Bem.: Trusted Subjekte sind per Definition vertrauenswürdig, d.h. sie unterliegen nicht den BLP-Regeln, sie dürfen Sicherheitseinstufungen ändern (zurückstufen), Problem?

(2) Problem des Blinden Schreibens

- *s* darf *o* modifizieren, aber anschließend (wegen no-read-up) nicht lesen!
- Problematisch in Bezug auf Integrität!

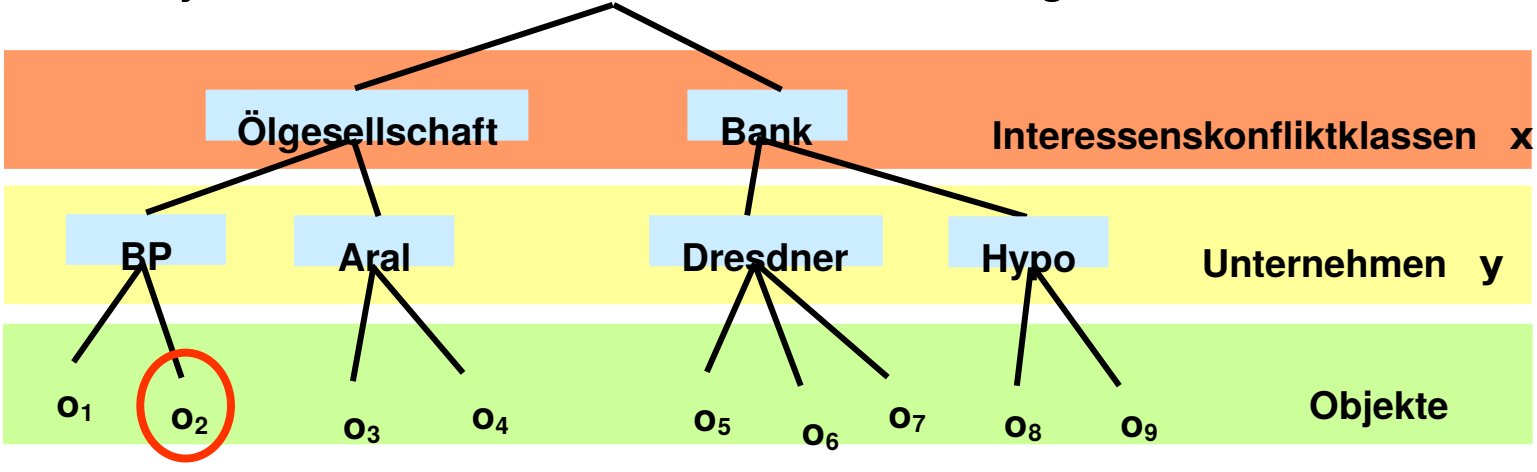
5.1.4 Chinese Wall Modell

Chinese Wall Modell

- Klassifikation von Objekten,
- Zugriffe sind abhängig von der Zugriffshistorie

COI (Conflict of Interest)-Regel:

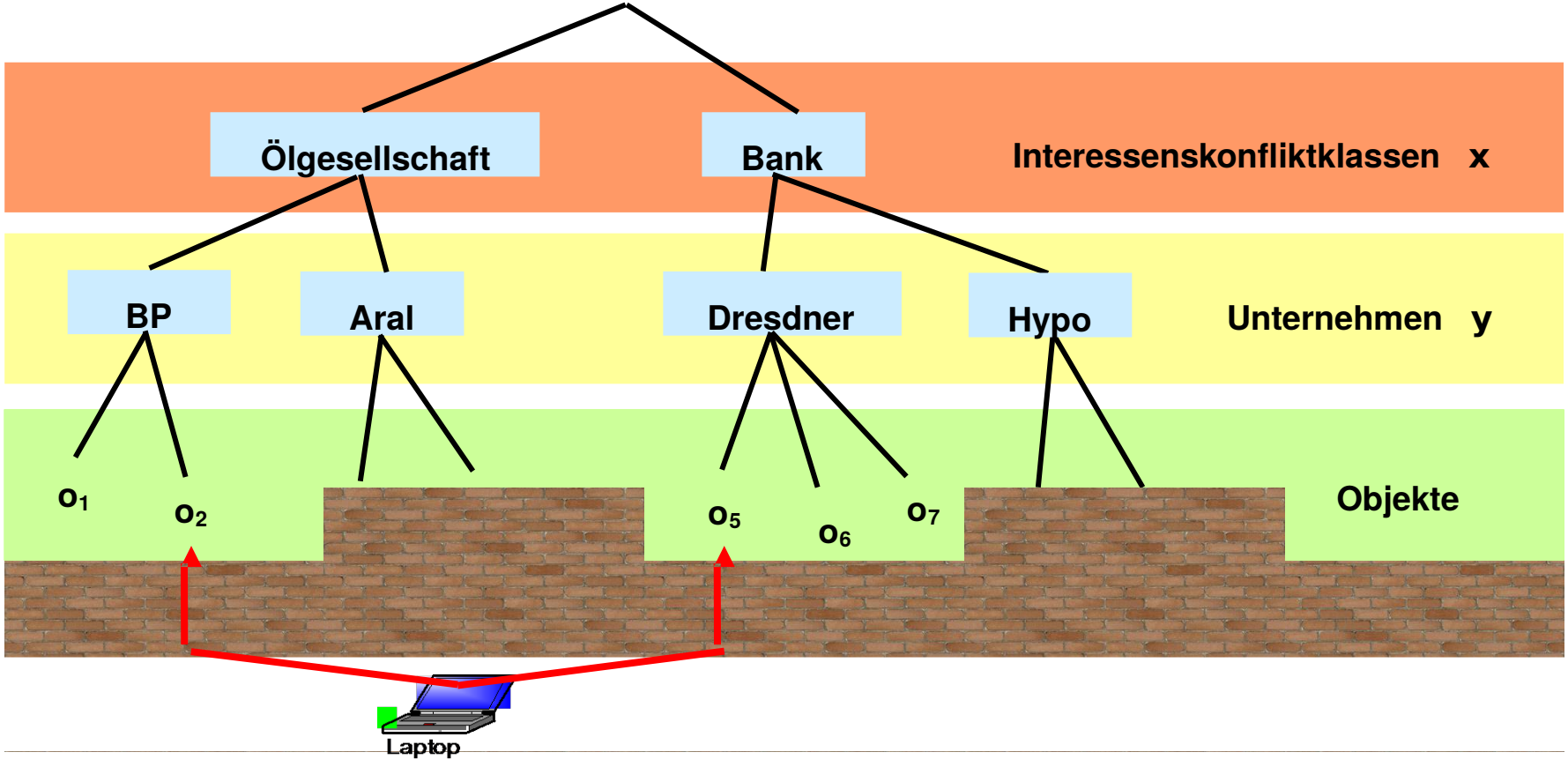
- Wer auf Objekte eines Unternehmens *U* in einer Konfliktklasse *K* zugreift, darf nicht mehr auf Objekte eines Unternehmens *U'* ≠ *U* zugreifen.



Zugriff: Subjekt s_1 auf das Objekt o_2 (zum Zeitpunkt t')

Nach Zugriff (Zugriffskontext-abhängig) wird eine‚Zugriffsmauer' aufgebaut:

Beispiel: Mauerbau nach Zugriff auf o_2 und o_5



Chinese-Wall-Modell-Konzepte

Chinese-Wall-Modell-Konzepte Objekte

- Zweistufige Objekthierarchie
- Einteilung in unterschiedliche Konfliktklassen *K* (Branchen) z.B.: Banken und Ölgesellschaften
- Unternehmen werden Konfliktklassen zugeordnet z.B.: Dresdner und HypoVereins-Bank der Klasse Bank

$o \in$ Objekte und deren Klassifikation

$y(o)$: Unternehmen, zu dem das Objekt gehört

$x(o)$: Interessenkonfliktklasse von o

y_0 : Öffentliches Objekt/Information

x_0 : Interessensfreie Information

Schutzzustand: Besteht aus zwei Teilen

- M_t Zugriffsmatrix
  - Benutzerbestimmbare, generische Rechte R = { Read, Write, Execute }
- N_t Zugriffshistorie

Zugriffshistorie

- $N_t : S \times O \rightarrow 2^R$ mit $N_t(s, o) = \{r_1,..., r_n\}$, falls Subjekt s mit den Rechten $\{r_1,..., r_n\}$ auf Objekt o vor dem Zeitpunkt t zugegriffen hat

Leseregel: Lesezugriff (Read-Access): Simple Property

- Leserecht, falls: $r \in M_t(s, o) \wedge \forall\, o' \in O$,

  $N_t(s, o') \neq \emptyset \rightarrow (y(o') = y(o) \vee x(o') \neq x(o) \vee y(o') = y_0 \vee y(o) = y_0)$

Schreibzugriff (Write-Access)

- Nur zulässig, wenn alle vorherigen Lesezugriffe auf
  - Objekte des gleichen Unternehmens oder auf
  - interessensfreie Objekte erfolgt sind
- Ein Subjekt $s \in S$ erlangt Schreibzugriff auf ein Objekt o zur Zeit t genau dann, wenn

$$\text{write} \in M_t(s, o) \wedge \forall\, o' \in O$$

$$\text{read} \in N_t(s, o') \rightarrow (\, y(o') = y(o) \vee y(o') = y_0 \,)$$

Problem bei Chinese-Wall Modell

- Aufhebung von Interessenskonflikten nicht vorgesehen
- Historie wächst im Prinzip unbegrenzt

Sicherheitsmodelle für zukünftige IT Systeme?

Zur Erinnerung: Charakteristika der zukünftigen Systeme

- Heterogen, dynamisch, mobil, vertrauenswürdige und nicht-vertrauenswürdige Komponenten kooperieren, …

Anforderung: u.a.

- Vertrauensmodelle:
  - Identitätsbasiert,
  - Verhaltensbasiert (z.B. Reputation, aber sehr schwach)
- Kontext-abhängige Modelle
  - Kontext-bewusste Zugriffskontrollen
  - Beispiel für Kontext-basierte Zugriffe**:**

    Chinese-Wall Modell: Historie der Zugriffe beschränkt die Zulässigkeit weiterer Zugriffe

Nächste Woche

Umsetzungskonzepte für den Zugriffsschutz