15 Bell Lapadula (Fortführung), Weitere Zugriffskontrollmodelle

Problem 1: Need-to-Know-Prinzip aufwändig umsetzbar
Nur in Zugriffsmatrix: Für alle Subjekte und Objekte

Lösung: Festlegung von Zuständigkeitsbereichen

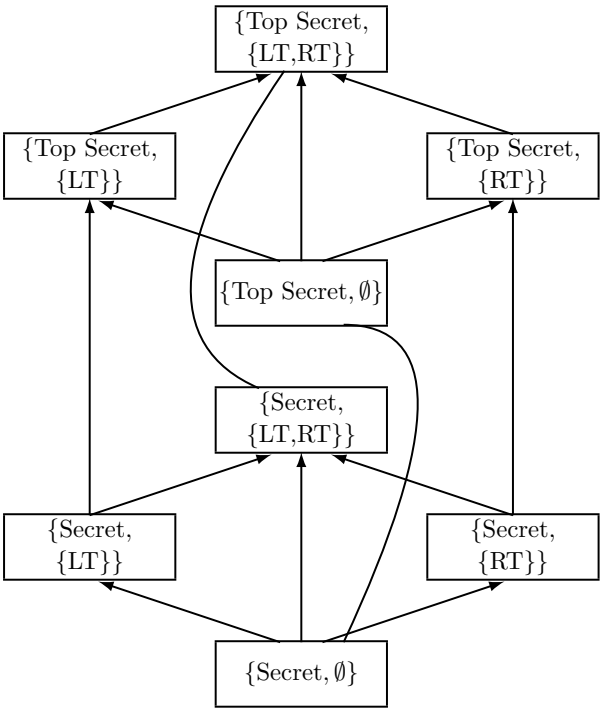
Erweiterung des Modells:

- Menge von Zuständigkeitsbereichen Z
  z.B. Links-, Rechtsterrorismus, Org. Kriminalität, Islamismus, ...
- Sicherheitsklassen $SK = SM \times \mathcal{P}(Z)$ mit partieller Ordnung $\leq$
  $(A, B) \leq (A', B') :\Longleftrightarrow A \leq A'$ und $B \subseteq B'$
- Erweiterung der Funktionen cl zu
  - $sc : S \longrightarrow SK$. Bedeutung $sc(s) = (A, B)$
    Subjekt s hat die Ermächtung A
    s ist Subjekt im Zuständigkeitsbereich b für alle $b \in B$.
  - $sc : O \longrightarrow SK$: Bedeutung $sc(o) = (A, B)$
    Objekt o hat die Eintstufung A
    o ist Objekt im Zuständigkeitsbereich b für alle $b \in B$.

Übung: Wie müssen die Eigenschaften (ii) und (iii) angepasst werden?

Beispiel. Zwei Sicherheitsmarken: $SM = \{\text{secret}, \text{top secret}\}$
Zwei Zuständigkeitsbereiche: $Z = \{\text{Linksterrorismus}, \text{Rechtsterrorismus}\}$.

Darstellung von $(SM \times \mathcal{P}(Z), \leq) : ((A, B) \to (A', B')$ heißt $(A, B) \leq (A', B')$
(Im Bild sind nur die unmittelbaren Nachfolger dargestellt)



Formal: SK zusammen mit $\leq$ bildet einen Verband:

- $(SK, \leq)$ ist eine partielle Ordnungsrelation
  (transitive, reflexive und antisymmetrische)
- Für je zwei Elemente $A, B \in SK$ gilt:
  - es gibt eindeutiges Infimum $\text{glb}(A, B) \in SK$ (greatest lower bound)
    d.h. $\text{glb}(A, B) \leq A, B$ und für alle $L \leq A, B$ gilt $I' \leq \text{glb}(A, B)$.
  - es gibt eindeutiges Supremum $\text{lub}(A, B) \in SK$ (least upper bound)
    d.h. $\text{lub}(A, B) \geq A, B$ und für alle $U \geq A, B$ gilt $U \geq \text{lub}(A, B)$.
- Informationsfluss nur über die Halbordnung
- Erweitertes BLP: Bsp. für verbandbasiertes Informationsflussmodell

Problem 2: Kein Schreiben von oben nach unten
Vorgesetzter kann keine Anweisung nach unten erteilen

Lösung: Zeitlich beschränkte untere Sicherheitsklasse
Es gibt für Subjekte zwei Funktionen

- $sc_s(s)$: Sicherheitsklasse von Subjekt s
- $sc_c(s)$: aktuelle Sicherheitsklasse von Subjekt s
  Aktuelle Klasse wird vom Nutzer selbst festgelegt
- Es gilt stets $sc_c(s) \leq sc_s(s)$
  Sicherheitsklasse dominiert die aktuelle Sicherheitsklasse

Übung: Wie muss die Eigenschaften (*iii*) angepasst werden?

Problem 3: Kein Integritätsschutz

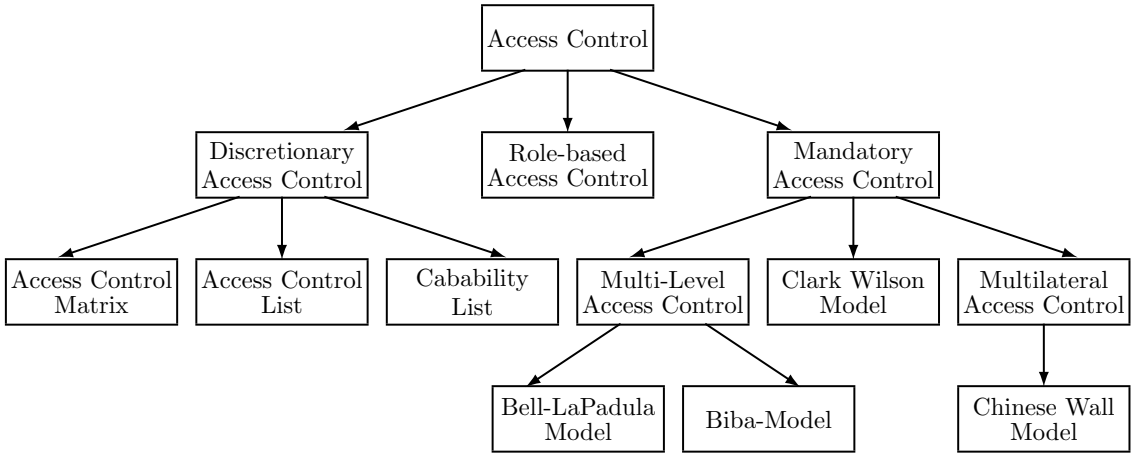
- $\star-$property: Erlaubt schreiben nach oben
  und damit Änderung eingestufter Informationen
- Beispiel für Sicherheitsmodelle für Integritätsschutz: Biba (Übung)

Problem 4: Keine Erfassung verdeckter Informationskanäle
Kanäle, die nicht für Informationsfluss vorgesehen sind

Beispiel.
- Ressourcen-Konflikt: High-Level-Prozess erzeugt eine Datei, so dass ein Low-Level-Prozess eine Fehlermeldung erhält, wenn eine Datei mit gleichem Namen erzeugt wird
- High-Level-Prozess kann gemeinsam genutzte Ressourcen (Position des Festplattenkopfes, Inhalt des Caches) in einem Zustand hinterlassen, so dass der Low-Level-Prozess anhand von Antwortzeiten zusätzliche Informationen gewinnen kann (Seitenkanalangriff)

Diskussion am Donnerstag

Überblick:



- Discretionary Access Control (benutzerbestimmt): behandelt
- Role-based Access Control: Abstraktionslevel: Rollen
- Mandatory Access Controll (systembestimmt): teilweise behandelt
- Multi-Level: Sicherheitsstufen mit Ordnung $\leq$ (Hierarchie)
  Ziel: Informationen und Änderungen nur entlang der Ordnung
- Clark Wilson: Übung
- Multilateral: Nicht nur Sicherheitsstufen, auch Need-to-Know-Prinzip
  Einführung von Sicherheitsklassen mit partieller Ordnung (Verband/Lattice)

Chinese Wall

Ziel: Keine (bewusste und unbewusste) Weitergabe von Insiderwissen

- Entwickelt von David F.C. Brewer und Michael J. Nash 1989
- Auch bekannt unter Brewer-Nash-Modell

Beispiel (Beratungsunternehmen). Ein Berater berät verschiedene Firmen der selben Branche. Wie wird sichergestellt, dass ein Berater Informationen, die er in einer Firma gesammelt hat nicht für eine andere ausnutzt.

Zugriffe müssen also auch von der Historie abhängen

Formalisierung

- R: Zugriffsrechte read, write-only, S: Subjekte (Berater)
- O: Objekte (Dateien), F: Firmen
- $y : O \longrightarrow F$: Jedem Objekt ist eine Firma zugeordnet
- Konfliktklassen $K_1, \ldots, K_n \subseteq F$ (disjunkt)
  Ziel: Kein Informationsfluss innerhalb einer Konfliktklasse
- $x : O \longrightarrow \{\emptyset, K_1, \ldots, K_n\}$: Objekt o ist Konfliktklasse $x(o)$ zugeordnet
  Sonderfall $x(o) = \emptyset : o$ ist nicht sensible, darf von allen gelesen werden
  $(x(o), y(o))$ heißt Sicherheitslabel von o
- $N_t = (n^t_{so})_{\substack{s \in S \\ o \in O}}$: Zugriffshistorie zum Zeitpunkt t
  $n^t_{so} \subseteq R$: Alle Rechte, mit denen s bisher auf o zugegriffen hat

Zugriffsregeln:

- ss-property: Direkte Informationsflusskontrolle:
  read-Zugriff auf o abhängig von bereits getätigten Zugriffen durch s
  Für alle $o' \in O$ mit $n^t_{so'} \neq \emptyset$ gilt
  - $y(o) = y(o')$: Objekte gehören zur selben Firma, oder
  - $x(o) \neq x(o')$: $y(o), y(o')$ sind in unterschiedl. Konfliktklassen, oder
  - $x(o) = \emptyset$: o ist nicht sensibel

ss-property kontrolliert nur direkten Informationsfluss

Beispiel (indirekter Informationsfluss).

Bank B, Firmen X und Y mit: X, Y sind in der selben Konfliktklasse, B in einer anderen. s_1 arbeitet für B und X, s_2 arbeitet für B und Y.

- s_1 liest o_1 mit $y(o_1) = X$ und schreibt dies in o_2 mit $y(o_2) = B$
- s_2 liest o_2 und schreibt dies in o_3 mit $y(o_3) = Y$

Indirekter Informationsfluss von o_1 in o_3.

Also weitere Regel erforderlich

- $\star-$property: Indirekte Informationsflusskontrolle:
  write-Zugriff von s auf o ist erlaubt, wenn
  Für alle $o' \in O$ mit read $\in n^t_{so'}$ gilt
  - $y(o) = y(o')$: Objekte gehören zur selben Firma, oder
  - $x(o') = \emptyset$: o' ist nicht sensibel