Politicas/Modelos Baseadas em Reticulados (*lattices*)

Políticas baseadas em reticulados

- Objectivo: prevenir o fluxo indevido de informação.

- Modelo:
  - Classificar sujeitos e objectos em níveis (classe de acesso).
  - Definir regras de acesso a objectos baseadas nas classificações dos sujeitos e objectos.

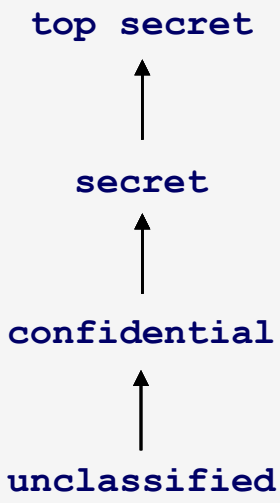
- Para classificar os sujeitos e objectos define-se um reticulado:
  - Um reticulado $(R, \leq)$ consiste num conjunto R e numa ordem parcial $\leq$, em que para qualquer $a,b \in R$ existe um *minimum upper bound* $u \in R$ e um *greatest lower bound* $l \in R$, i.e.:

    - $a \leq u,\ b \leq u,\ \forall\ v \in R : (a \leq v\ \wedge\ b \leq v) \Rightarrow (u \leq v)$

    - $l \leq a,\ l \leq b,\ \forall\ k \in R : (k \leq a\ \wedge\ k \leq b) \Rightarrow (k \leq l)$

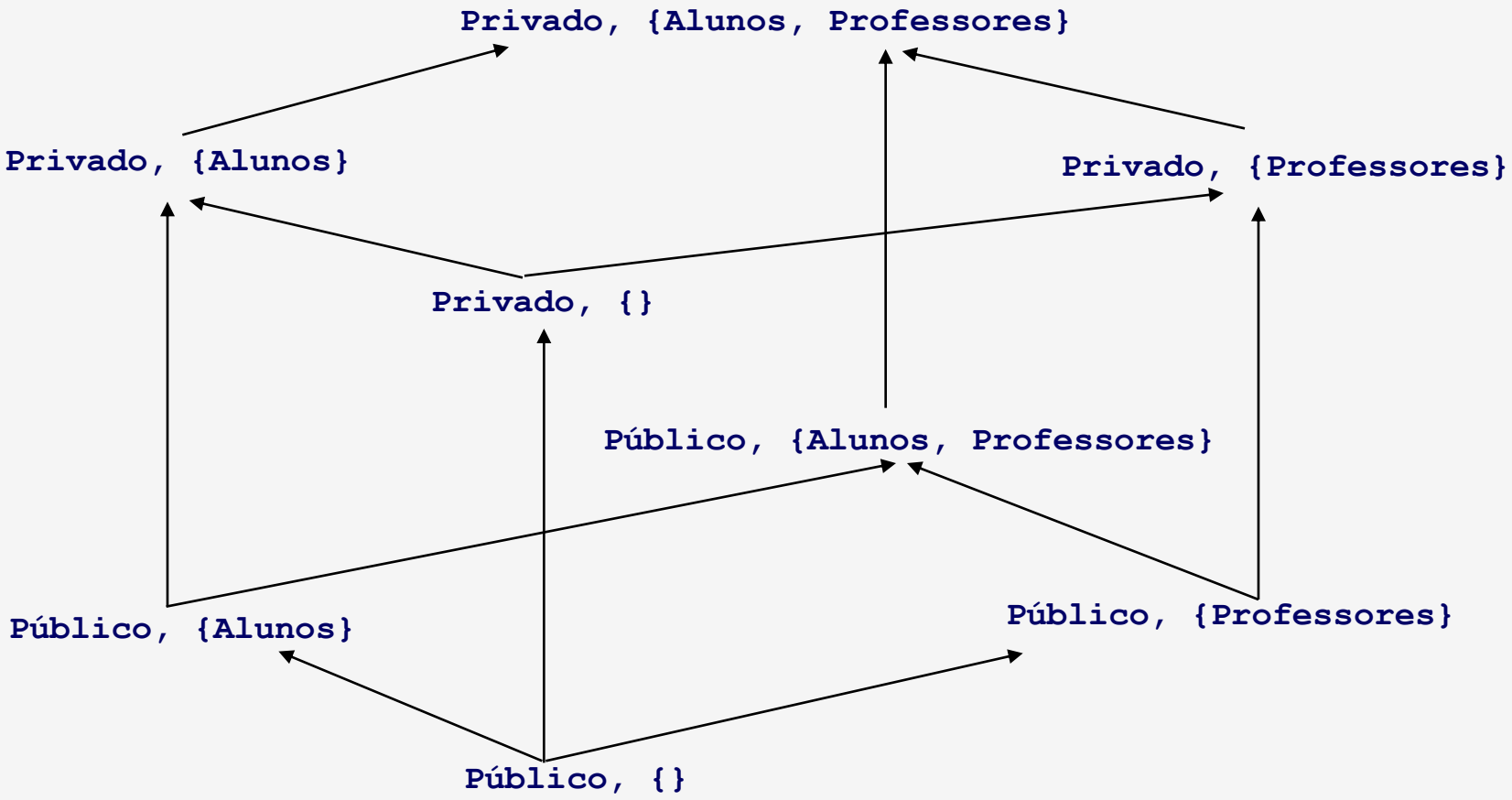
Exemplo I

- Reticulado (R, ≤) tal que:
  - R = {top secret, secret, confidential, unclassified}
  - unclassified ≤ confidential ≤ secret ≤ top secret



Exemplo II

- Exemplo de reticulado em que a relação de ordem parcial depende de duas classificações.



Modelo Bell-LaPadula

Introdução

- Bell and LaPadula: “Secure computer systems: Mathematical foundations”, Technical report ESD-TR-278, The Mitre Corp., 1973.

- Desenvolvido a pensar nos aspectos de confidencialidade de organizações militares e governamentais (controlo de fluxo de informação).

- Modela os seguintes aspectos de confidencialidade no controlo de acessos:
  - Não permitir leituras de objectos classificados acima da classificação do sujeito (no read-up).
  - Não permitir escritas em objectos classificados abaixo da classificação do sujeito (no write-down).

- Baseado numa classificação de sujeitos e objectos com base num reticulado e numa matriz de controlo de acessos.

- Implementação sobre a abstracção “máquina de estados” em que a avaliação do controlo de acessos é feita com base no estado actual do sistema.

Formalização

- O modelo Bell-LaPadula é composto por:
  - Um conjunto de sujeitos *S*.
  - Um conjunto de objectos *O*.
  - Um conjunto de operações de acesso *A*={read, write, append, execute}.
  - Um reticulado *R* com uma ordem parcial ≤.

- O estado do sistema é definido por *B x M x F*, onde:
  - *B = P(S x O x A)* indica os acessos activos, i.e. $b \in B$ é uma colecção de triplos (*s,o,a*) que indicam que o sujeito *s* esta a efectuar a operação *a* sobre o objecto *o.*
  - *M* é a matriz de acessos $M = M_{so}$
  - $F \subset R^s \times R^s \times R^o$ é o conjunto de classificações. Um elemento $f \in F$ é um triplo (f_s, f_c, f_o), onde:
    - f_s : *S -> L,* retorna a classificação máxima que um sujeito possuí.
    - f_c : *S -> L,* retorna a classificação actual que um sujeito possuí.
    - f_o : *S -> L,* retorna a classificação dos objectos.

Propriedades requeridas para o estado do sistema

- **Propriedade SS** (simple security property)
  - Um estado (*b,M,f)* satisfaz a propriedade **SS** sse para cada elemento (*s,o,a*) $\in$ *b,* onde a operação é *read* ou *write,* a classificação de segurança do sujeito domina a classificação de segurança do objecto, i.e. $f_o(o) \leq f_s(s)$. (no read-up)

- **Propriedade *** (star property)
  - Um estado (*b,M,f)* satisfaz a propriedade ***** sse para cada elemento (*s,o,a*) $\in$ *b,* onde a operação é *append* ou *write,* a classificação de segurança actual do sujeito é dominada pela classificação de segurança do objecto, i.e. $f_c(s) \leq f_o(o)$. (no write-down)

- **Propriedade ds** *(discretionary property)*
  - Um estado (*b,M,f)* satisfaz a propriedade **ds** sse para cada elemento (*s,o,a*) $\in$ *b* temos $a \in M_{so}$.

- Um estado é considerado seguro se todas as três propriedades se verificarem.

- Como é que um chefe envia uma mensagem para os seus subordinados?

Transições de estado

- Uma transição entre estados $E_1 = (b_1, M_1, f_1) \rightarrow E_2 = (b_2, M_2, f_2)$ é considerada segura sse ambos os estados forem seguros.

- **Basic Security Theorem (BST)**
  - Se todas as transições de estados forem seguras e se o estado inicial do sistema for seguro, então qualquer estado subsequente ao inicial é seguro não importa quais as entradas que possam ocorrer.

Nota: o BST não é consequência das propriedades mas sim do modelo de máquina de estados.

Limitações do modelo Bell-LaPadula

- Só contempla aspectos de confidencialidade deixando de fora os aspectos relacionados com a integridade.
- Contém canais encobertos (*covert channels).*
- Não define o modelo de gestão do controlo de acessos.

Modelo Biba

Introdução

- Biba: “Integrity considerations for secure computer system”. Technical report ESDTR-76-372, MTR-3153, The Mitre Corp., 1977.

- Desenvolvido a pensar nos aspectos de integridade no controlo de acessos, e.g. não permitir que entidades limpas de alto nível (*clean*) sejam contaminadas por entidades sujas de baixo nível (*dirty*).

- Baseado num reticulado *R* de níveis de integridade e funções f_s e f_o:
  - f_s : *S -> L,* retorna o nível de integridade dos sujeitos.
  - f_o : *S -> L,* retorna o nível de integridade dos objectos.

- Ao contrário do modelo Bell-LaPadula existem várias abordagens ao modelo (políticas de integridade).

Abordagem de níveis de integridade estáticos

- **Propriedade de tranquilidade**
  - Não existe mudança na associação dos nível de integridade.

- **Propriedade SS**
  - Se um sujeito *s* só pode escrever (modificar) um objecto *o* sse $f_o(o) \leq f_s(s)$. (no write-up)

- **Propriedade * de integridade**
  - Se um sujeito *s* pode ler (observar) um objecto *o,* então *s* só pode ter acesso de escrita a um outro objecto *p* sse $f_o(p) \leq f_o(o)$

- Estas propriedades previnem que sujeitos e objectos limpos sejam contaminados por informação suja.

Abordagem de níveis de integridade dinâmicos

- Nesta abordagem existe uma actualização automática do nível de integridade sempre que se efectua o contacto com informação suja.

- **Subject low watermark property**
  - Um sujeito *s* pode ler (observar) um objecto *o* de qualquer nível de integridade. O nível de integridade do sujeito é automaticamente actualizado para $inf(f_s(s), f_o(o))$.

- **Object low watermark property**
  - Um sujeito *s* pode escrever (alterar) um objecto *o* de qualquer nível de integridade. O nível de integridade do objecto é automaticamente actualizado para $inf(f_s(s), f_o(o))$.

- $inf(f_s(s), f_o(o))$ é o *greatest lower bound* entre $f_s(s)$ e $f_o(o)$.

Alternativas para políticas de invocação

- O modelo Biba pode ser estendido para suportar politicas de invocação de objectos/sujeito (ex: invocação de um programa/serviço).

- Propriedades/alternativas para política de invocação:
  - **Invoke property**
    - Um sujeito s_1 pode invocar um sujeito s_2 sse $f_s(s_2) \leq f_s(s_1)$.
    - Os sujeitos só podem invocar programas/serviços a de um nível igual ou inferior, logo não podem usar um programa/serviço de nível superior para contaminar informação.
  - **Ring Property**
    - Um sujeito s1 pode ler (observar) objectos de qualquer nível mas só pode escrever (alterar) objectos *o* sse $f_o(o) \leq f_s(s_1)$. O sujeito pode ainda invocar um sujeito s2 sse $f_s(s_1) \leq f_s(s_2)$.
    - Assim é possível controlar o acesso para escrita a objectos de um nível superior através de verificações efectuadas pelo sujeito invocado.