Segurança Computacional

Modelos de controle de acesso

Prof. Carlos Maziero

DInf UFPR, Curitiba PR

Setembro de 2019

Conteúdo

Conceitos básicos

Controle de acesso

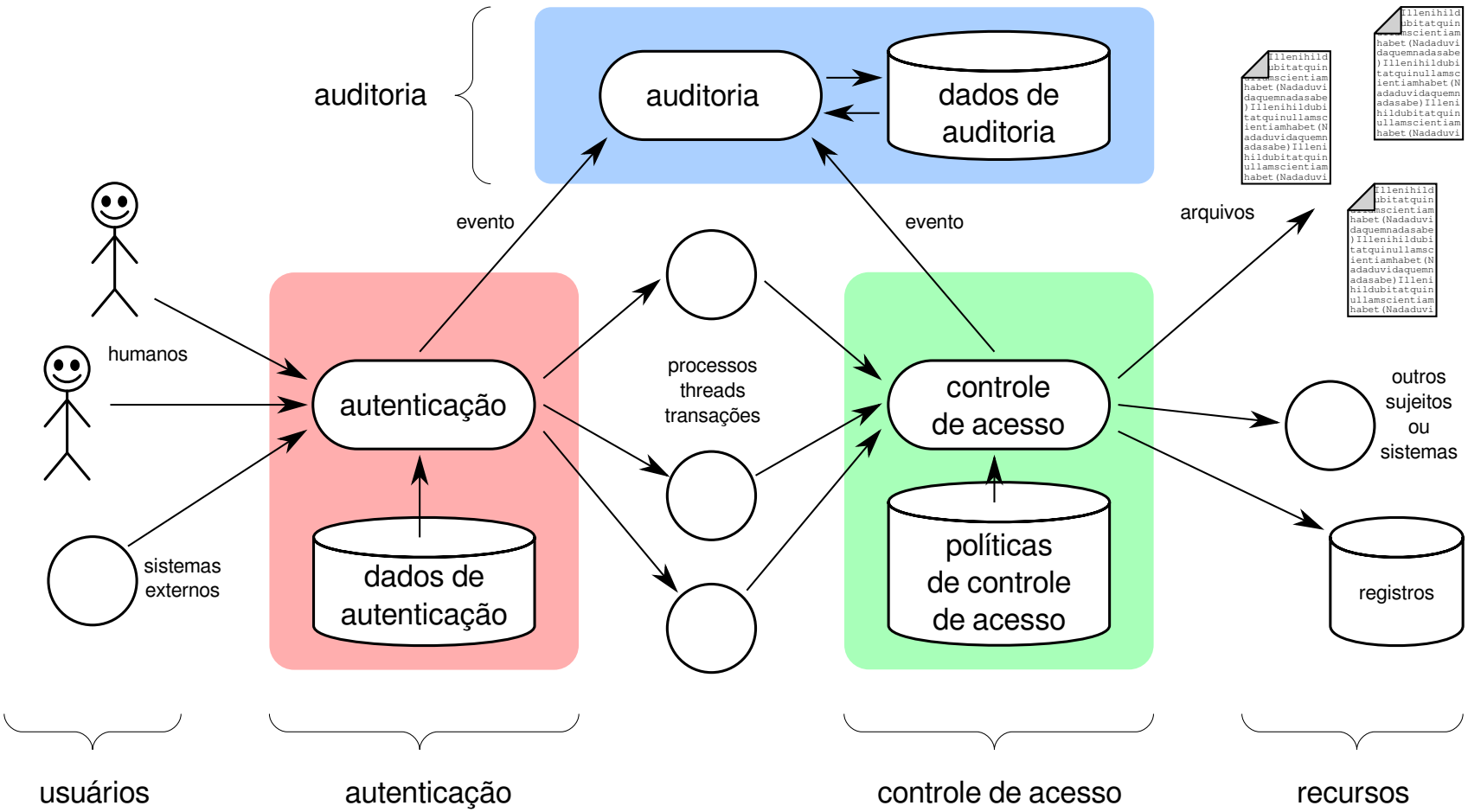
Definição

Mediar cada solicitação de acesso de um usuário autenticado a um recurso do sistema, para determinar ela deve ser autorizada ou negada

Em outras palavras:

Decidir **quem** pode acessar **que** recursos e **como**

Controle de acesso



Terminologia

Sujeito:

- Entidade que executa ações no sistema
- Processos, threads, transações
- Representa um usuário (humano ou outro sistema)

Objeto:

- Sofre os acessos dos sujeitos
- Arquivos, áreas de memória, registros, ...
- Geralmente são associados aos recursos

Terminologia

Acesso:

- Ação realizada por um sujeito sobre um objeto
- Leitura de um arquivo por um processo
- Envio de pacote de rede através de uma porta UDP
- Execução de um programa

Autorização:

- Permissão para um sujeito acessar um objeto
- Pode ser positiva (permitir a ação) ou negativa (negá-la)

Terminologia

Sujeitos e objetos possuem *atributos*:

- **Sujeito**: nome, idade, ID, grupo, ...
- **Objeto**: tipo, localização, valor, ...

Sujeitos e objetos podem ser organizados em para facilitar a gerência:

- **Grupos**: aluno BCC, aluno IBM, aluno Matemática, ...
- **Hierarquias**: soldado < cabo < sargento < tenente

Um sujeito pode ser visto como objeto por outro sujeito (exemplo: um sujeito envia uma mensagem a outro)

Políticas, modelos e mecanismos

Políticas, modelos e mecanismos

Política de controle de acesso

Visão abstrata das possibilidades de acesso objetos pelos sujeitos, definida por regras informais

Modelo de controle de acesso

Representação lógica ou matemática de uma política, que facilita sua implementação e permite a análise de erros

Mecanismo de controle de acesso

Estruturas necessárias à implementação de um modelo em um sistema real

Política de controle de acesso

Exemplo: política de um sistema de informações médicas:

- Médicos podem **consultar** os prontuários de seus pacientes;
- Médicos podem **modificar** os prontuários de seus pacientes enquanto estes estiverem internados;
- O supervisor geral pode consultar os prontuários de **todos** os pacientes;
- Enfermeiros **só podem consultar** os prontuários dos pacientes de sua seção e somente durante seu período de turno;
- Prontuários de pacientes de planos de saúde privados podem ser consultados pelo respectivo responsável no hospital;
- Pacientes podem consultar seus próprios prontuários (aceitar no máximo 30 pacientes simultâneos)

Construção de políticas

As regras ou definições individuais de uma política são denominadas **autorizações**

As autorizações podem ser baseadas em **identidades** (como sujeitos e objetos) ou em **outros atributos** (como idade, sexo, tipo, preço, etc)

As autorizações podem ser **individuais** (a sujeitos ou objetos) ou **coletivas** (a grupos)

Podem existir autorizações **positivas** (permitindo o acesso) ou **negativas** (negando o acesso)

Uma política pode ter autorizações dependentes de **condições externas** (como o tempo ou a carga do sistema)

Também deve ser definida uma **política administrativa**, que define quem pode modificar/gerenciar as políticas vigentes no

Classes de políticas

O conjunto de autorizações de uma política deve:

- Ser **completo**: cobrir todas as possibilidades de acesso
- Ser **consistente**: sem regras conflitantes entre si
- Respeitar o **princípio do privilégio mínimo**: um usuário nunca deve receber mais autorizações que aquelas que necessita

A construção e validação de políticas pode ser muito complexa

Modelo de controle de acesso

Representação lógica / matemática de uma política

- Sistematiza as regras e facilita a implementação
- Permite a análise e identificação de erros e conflitos

Um modelo é composto de:

- Expressões lógicas sobre os atributos do sujeito e objeto
- Condições externas (horário, carga no sistema, etc)

Modelos são implementados por mecanismos

Classes de políticas

Políticas *discricionárias*

Políticas *obrigatórias*

Políticas *baseadas em domínios*

Políticas *baseadas em papéis*

Políticas discricionárias

Políticas discricionárias (DAC)

DAC: *Discretionary Access Control*

Permissões atribuídas de forma discricionária

Usa regras do tipo $\langle s, o, a \rangle$:

- O sujeito "s" pode executar a ação "a" sobre o objeto "o"
- Autorização positiva: $\langle s, o, +a \rangle$
- Autorização negativa: $\langle s, o, -a \rangle$

Exemplos:

- `Bob` pode ler e escrever arquivos em `/home/Bob`
- O grupo `admin` pode ler arquivos em `/suporte`

Matriz de controle de acesso

Modelo matemático baseado em uma matriz

Conveniente para representar políticas discricionárias

Matriz de autorizações:

- Linhas correspondem aos sujeitos do sistema
- Colunas correspondem aos objetos
- Células correspondem às autorizações

Matriz de controle de acesso

Um conjunto de sujeitos $\mathbb{S} = \{s_1, s_2, \ldots, s_m\}$

Um conjunto de objetos $\mathbb{O} = \{o_1, o_2, \ldots, o_n\}$

Um conjunto de ações sobre os objetos $\mathbb{A} = \{a_1, a_2, \ldots, a_p\}$

$$\forall s_i \in \mathbb{S} \quad \forall o_j \in \mathbb{O} \quad M_{ij} \subseteq \mathbb{A}$$

Cada elemento M_{ij} da matriz M é um sub-conjunto das ações possíveis $\mathbb{A}$, que define as ações que o sujeito $s_i \in \mathbb{S}$ pode efetuar sobre o objeto $o_j \in \mathbb{O}$

Matriz de controle de acesso

$\mathbb{S} = \{Alice, Bob, Carol, David\}$,
$\mathbb{O} = \{file_1, file_2, program_1, socket_1\}$ e
$\mathbb{A} = \{read, write, execute, remove\}$

	$file_1$	$file_2$	$program_1$	$socket_1$
Alice	*read* *write* *remove*	*read* *write*	*execute*	*write*
Bob	*read* *write*	*read* *write* *remove*	*read*	
Carol		*read*	*execute*	*read* *write*
David	*read*	*append*	*read*	*read* *append*

Política Administrativa

P: Quem pode modificar as regras de controle de acesso?

R: Precisamos de regras de controle de acesso para isso!

Possibilidades:

- Um administrador define as políticas
- Definir um proprietário para cada objeto
- Definir regras de acesso a M (M é um objeto)

A noção de "proprietário" de um objeto é frequente, mas **não é necessária** em políticas DAC

Matriz administrativa

	$file_1$	$file_2$	$program_1$	$socket_1$
Alice	*read* *write* *remove* *owner*	*read* *write*	*execute*	*write*
Bob	*read* *write*	*read* *write* *remove* *owner*	*read* *owner*	
Carol		*read*	*execute*	*read* *write*
David	*read*	*append*	*read*	*read* *owner*

Matriz de controle de acesso

Modelo conceitual **inadequado para implementação**

- Muitos sujeitos e objetos = muito espaço
- Localidade de referências: matriz esparsa

Simplificações implementáveis:

- Tabelas de autorizações
- Listas de controle de acesso
- Listas de capacidades

Tabela de autorizações

Abordagem frequente em sistemas de bancos de dados (DBMS)

Sujeito	Objeto	Ação
Alice	$file_1$	*read*
Alice	$file_1$	*write*
Alice	$file_1$	*remove*
Alice	$file_1$	*owner*
Alice	$file_2$	*read*
Alice	$file_2$	*write*
Alice	$program_1$	*execute*
Alice	$socket_1$	*write*
Bob	$file_1$	*read*
Bob	$file_1$	*write*
Bob	$file_2$	*read*
Bob	$file_2$	*write*
Bob	$file_2$	*remove*

Sujeito	Objeto	Ação
Bob	$file_2$	*owner*
Bob	$program_1$	*read*
Bob	$socket_1$	*owner*
Carol	$file_2$	*read*
Carol	$program_1$	*execute*
Carol	$socket_1$	*read*
Carol	$socket_1$	*write*
David	$file_1$	*read*
David	$file_2$	*write*
David	$program_1$	*read*
David	$socket_1$	*read*
David	$socket_1$	*write*
David	$socket_1$	*owner*

Listas de controle de acesso

ACL: Lista de quem pode acessar um objeto:

- A cada objeto é associada uma ACL
- A ACL indica **os sujeitos** que podem acessar o objeto
- Para cada sujeito são definidas permissões

A ACL de um objeto é a **coluna** dele na matriz M

Implementação muito usada, simples de implementar e robusta

Listas de controle de acesso

$$
\begin{aligned}
ACL(file_1) &= \{ \quad Alice : (read, write, remove, owner), \\
&\qquad\quad Bob : (read, write), David : (read) \ \} \\
ACL(file_2) &= \{ \quad Alice : (read, write), \\
&\qquad\quad Bob : (read, write, remove, owner), \\
&\qquad\quad Carol : (read), David : (write) \ \} \\
ACL(program_1) &= \{ \quad Alice : (execute), \\
&\qquad\quad Bob : (read, owner), \\
&\qquad\quad Carol : (execute), David : (read) \ \} \\
ACL(socket_1) &= \{ \quad Alice : (write), Carol : (read, write), \\
&\qquad\quad David : (read, write, owner) \ \}
\end{aligned}
$$

Listas de capacidades

CL - *Capability List*:

- Lista de objetos que um sujeito pode acessar
- Essa lista corresponde a uma linha da matriz de acesso
- Pode ser vista como uma ficha ou *token* de acesso

Problemas:

- dificuldade de implementação (garantia de integridade)
- como modificar uma lista de capacidades já outorgada?
- como retirar uma permissão concedida a um sujeito?

Exemplo de capability: certificado digital

Listas de capacidades

$$
\begin{aligned}
CL(Alice) &= \{ \; file_1 : (read, write, remove, owner), \\
&\quad\; file_2 : (read, write), \\
&\quad\; program_1 : (execute), socket_1 : (write) \; \} \\
CL(Bob) &= \{ \; file_1 : (read, write), \\
&\quad\; file_2 : (read, write, remove, owner), \\
&\quad\; program_1 : (read, owner) \; \} \\
CL(Carol) &= \{ \; file_2 : (read), program_1 : (execute), \\
&\quad\; socket_1 : (read, write) \; \} \\
CL(David) &= \{ \; file_1 : (read), file_2 : (write), \\
&\quad\; program_1 : (read), \\
&\quad\; socket_1 : (read, write, owner) \; \}
\end{aligned}
$$

Políticas obrigatórias

Políticas Obrigatórias (MAC)

MAC: *Mandatory Access Control*

Controle definido por regras globais incontornáveis:

- não dependem das identidades dos sujeitos e objetos
- não dependem da vontade de seus proprietários
- não dependem do administrador do sistema

Regras baseadas em atributos dos sujeitos e/ou dos objetos:

- Cheques acima de R$ 8.000,00 não podem ser descontados
- Clientes com renda acima de R$5.000,00 não têm crédito consignado

Políticas Multi-nível

Abordagem MAC usual:

- **Políticas multi-nível** (MLS - *Multi-Level Security*)
- Classifica sujeitos e objetos em *níveis de segurança* $\mathbb{S}$

Exemplo: níveis de confidencialidade de um documento:

- *TS*: *Top Secret*
- *S*: *Secret*
- *C*: *Confidential*
- *R*: *Restrict*
- *U*: *Unclassified* / *Public*

Políticas Multi-nível

Considera-se que os níveis de segurança estão ordenados:

$$\mathbb{S} : \; U < R < C < S < TS$$

Níveis são associados às entidades do sistema:

- $h(s_i) \in \mathbb{S}$: **habilitação** do sujeito s_i
- $c(o_j) \in \mathbb{S}$: **classificação** do objeto o_j

As regras são construídas usando habilitações e classificações

Modelo de Bell-LaPadula (BLP)

Proposto em 1973 pela MITRE Corp para o US DoD

Protege a **confidencialidade** de dados

Consiste de duas regras e um princípio:

- regra *No-Read-Up* (propriedade simples)
- regra *No-Write-Down* (propriedade $\star$)
- princípio da tranquilidade (níveis estáveis)

O modelo BLP complementa um modelo DAC subsequente

Modelo de Bell-LaPadula

Regra *No-Read-Up* (propriedade simples):

- Um sujeito **não pode ler** objetos em níveis de segurança **acima** do seu
- Evita que um sujeito não-habilitado acesse informação indevida
- Exemplo: um sujeito habilitado como confidencial só pode ler objetos com classificação confidencial, reservada ou pública

Modelo de Bell-LaPadula

Regra *No-Write-Down* (propriedade ⋆):

- Um sujeito **não pode escrever** em objetos **abaixo** de seu nível de segurança
- Evita que um sujeito habilitado "vaze" informação
- Exemplo: Um sujeito habilitado como confidencial só pode **escrever** em objetos com classificação confidencial, secreta ou ultrassecreta

Modelo de Bell-LaPadula

Princípio da tranquilidade:

- **Forte**: os níveis de segurança não variam durante a operação
- **Fraca**: os níveis de segurança não variam durante um acesso

Modelo de Bell-LaPadula

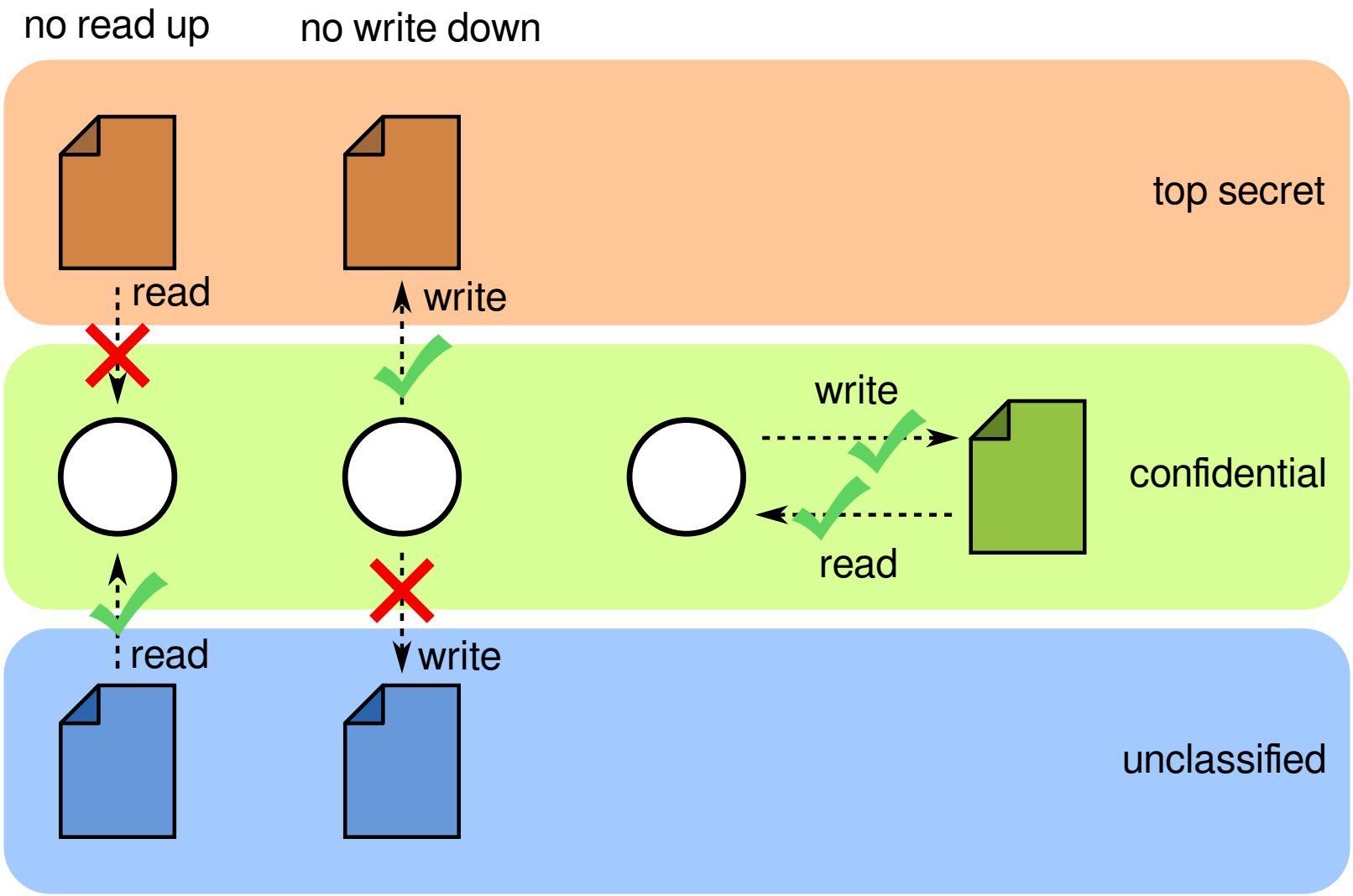
$$\mathbb{S} = \{U < R < C < S < TS\}$$

$$\forall s_i, h(s_i) \in \mathbb{S},\ \forall o_j, c(o_j) \in \mathbb{S}$$

$$request(s, o, read) \iff h(s) \geq c(o)$$

$$request(s, o, write) \iff h(s) \leq c(o)$$

Modelo de Bell-LaPadula



Modelo de Biba

Proposto por Kenneth Biba em 1975 (Mitre Corp, USA)

Dual do BLP, protege a **integridade** dos dados

Consiste de duas regras e um princípio:

- regra *No-Write-Up* (propriedade simples de integridade)
- regra *No-Read-Down* (propriedade $\star$ de integridade)
- princípio da tranquilidade (níveis estáveis)

Complementa um modelo DAC subsequente

Modelo de Biba

Considerar os níveis de integridade:

$$\mathbb{I} = \{B < M < A < S\} \quad \text{(baixa, média, alta e sistema)}$$

Regra *No-Write-Up* (propriedade simples de integridade):

- Um sujeito **não pode escrever** em objetos acima de seu nível de integridade
- Preserva a integridade de objetos críticos
- Exemplo: um sujeito de integridade média (M) somente pode escrever em objetos de integridade baixa (B) ou média (M)

Modelo de Biba

Regra *No-Read-Down* (propriedade $\star$ de integridade):

- Um sujeito **não pode ler** objetos em níveis de integridade abaixo do seu
- Evita o risco de ler informação duvidosa
- Exemplo: um sujeito com integridade alta (A) somente pode ler objetos com integridade alta (A) ou de sistema (S)

Modelo de Biba

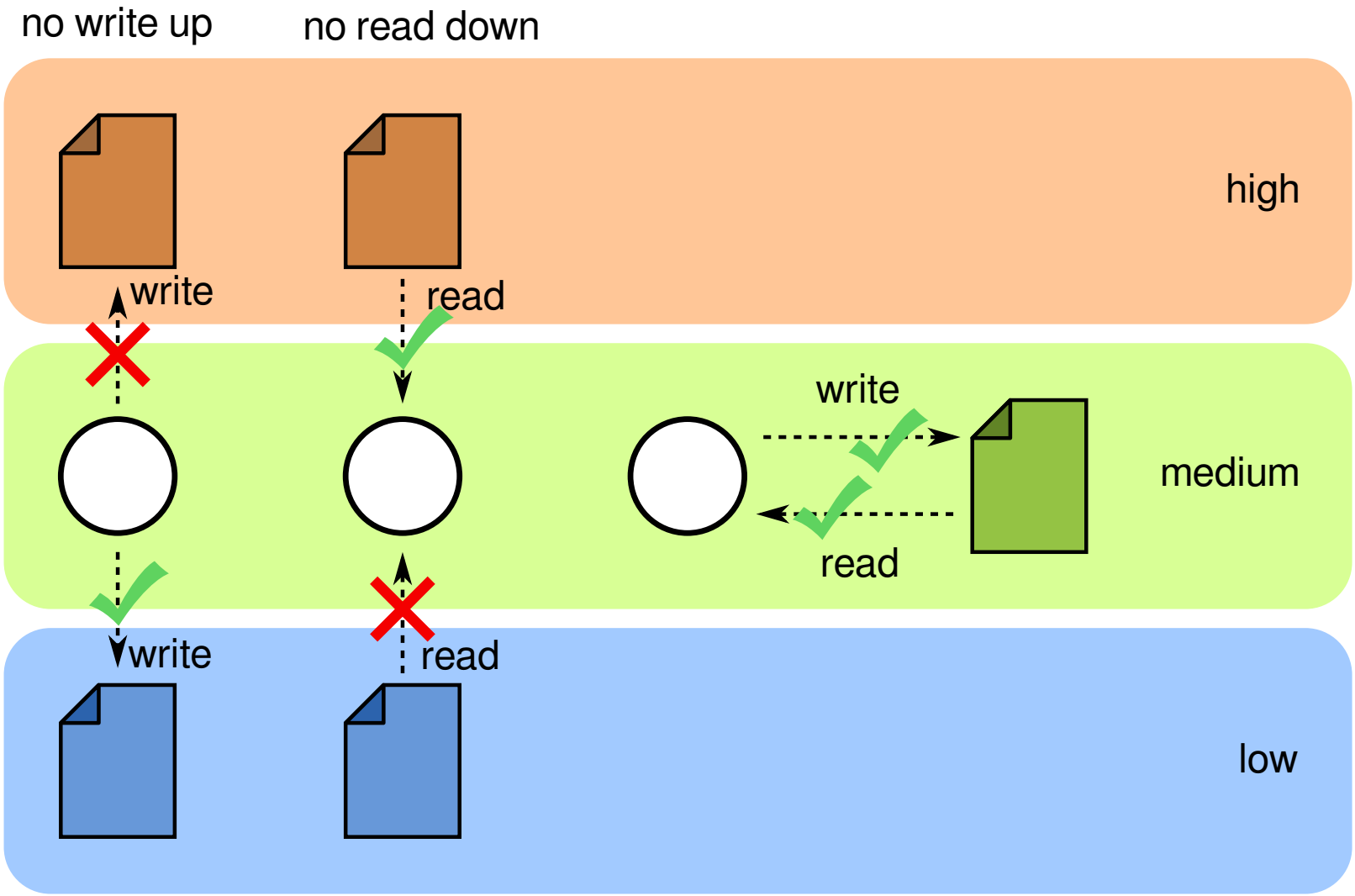
$$\mathbb{I} = \{B, M, A, S\}, B < M < A < S$$

$$\forall s_i, i(s_i) \in \mathbb{I}, \ \forall o_j, i(o_j) \in \mathbb{I}$$

$$request(s, o, write) \iff i(s) \geq i(o)$$

$$request(s, o, read) \iff i(s) \leq i(o)$$

Modelo de Biba



Categorias em BLP e Biba

Uma categoria:

- define uma área funcional dentro do sistema:
  - “pessoal”, “projetos”, “financeiro”, “suporte”, etc
- O conjunto de categorias é estático
- Não há uma ordem hierárquica entre categorias

Categorias permitem separar os objetos em áreas de interesse

Uso das categorias

Cada sujeito e objeto é rotulado com uma ou mais categorias

Um sujeito somente acessa objetos pertencentes às mesmas categorias dele, ou a um sub-conjunto destas

Exemplo: um sujeito com categorias $\{suporte, financeiro\}$ só acessa objetos rotulados como:

- $\{suporte, financeiro\}$
- $\{suporte\}$
- $\{financeiro\}$
- $\{\phi\}$

Formalmente: $acesso(s, o) \iff cat(s) \supseteq cat(o)$

Políticas baseadas em domínios

Políticas baseadas em domínios

Domínio de segurança:

- Conjunto de objetos que um sujeito pode acessar
- Geralmente está implícito nas regras das políticas

Políticas baseadas em domínios:

- Definição explícita de domínios de segurança
- Cada sujeito s é associado a um domínio $domain(s)$
- Cada objeto o é associado a um tipo $type(o)$

Modelo DTE

DTE - *Domain and Type Enforcement*

Permissões *sujeito* $\rightarrow$ *objeto* :

- Definidas em uma tabela global DDT
- DDT - *Domain Definition Table*
- Linhas associadas a domínios, colunas a tipos
- $DDT[x, y]$: permissões de sujeitos em D_x a objetos em T_y

$$req(s, o, action) \iff action \in DDT[domain(s), type(o)]$$

Modelo DTE

Interações entre sujeitos:

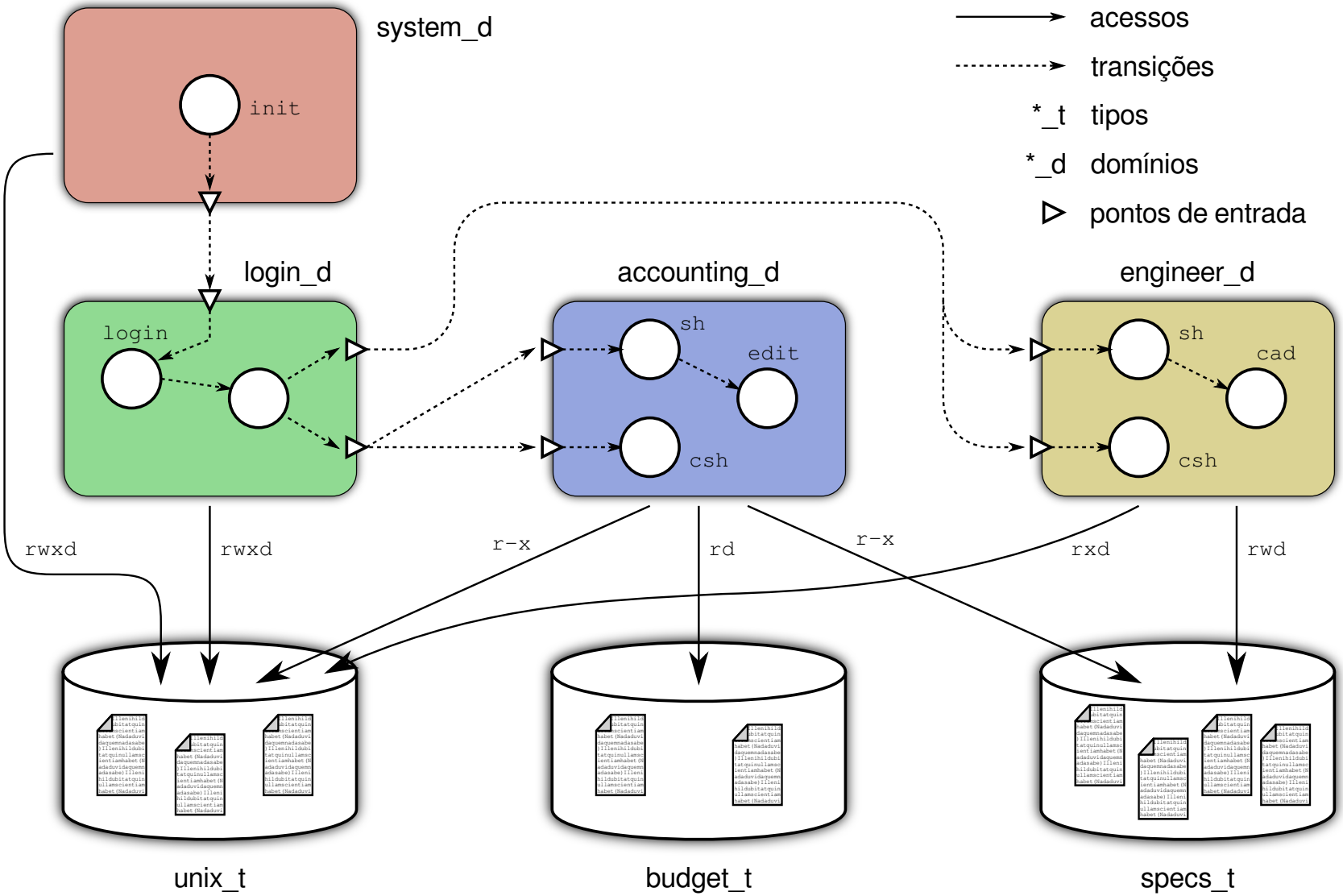
- Trocas de mensagens, sinais, mudanças de domínio, etc
- DIT - *Domain Interaction Table*
- Linhas e colunas correspondem a domínios
- $DIT[x, y]$: interações possíveis entre sujeitos em D_x e D_y:

$$req(s_i, s_j, int) \iff int \in DIT[domain(s_i), domain(s_j)]$$

Exemplo de política DTE

```
1  type unix_t,          /* normal UNIX files, programs, etc. */
2       specs_t,         /* engineering specifications */
3       budget_t,        /* budget projections */
4       rates_t;         /* labor rates */
5
6  #define DEFAULT (/bin/sh), (/bin/csh), (rxd->unix_t) /* macro */
7
8  domain engineer_d   = DEFAULT, (rwd->specs_t);
9  domain project_d    = DEFAULT, (rwd->budget_t), (rd->rates_t);
10 domain accounting_d = DEFAULT, (rd->budget_t), (rwd->rates_t);
11 domain system_d     = (/etc/init), (rwxd->unix_t), (auto->login_d);
12 domain login_d      = (/bin/login), (rwxd->unix_t),
13                       (exec-> engineer_d, project_d, accounting_d);
14
15 initial_domain system_d;  /* system starts in this domain */
16
17 assign -r  unix_t     /;          /* default for all files */
18 assign -r  specs_t    /projects/specs;
19 assign -r  budget_t   /projects/budget;
20 assign -r  rates_t    /projects/rates;
```

Exemplo de política DTE



Políticas baseadas em papéis

Políticas baseadas em papéis

Administração de políticas é um grande problema:

- Políticas MAC são consideradas pouco flexíveis
- Políticas DAC acabam sendo muito mais usadas
- Gerenciar autorizações em um ambiente dinâmico:
  - usuários mudam de cargo
  - usuários assumem novas responsabilidades
  - usuários entram e saem da empresa
- Erros podem ocorrer!

RBAC - *Role-Based Access Control*

Define um conjunto de **papéis** no sistema

- "diretor", "gerente", "suporte", "programador"

Atribui a cada papel um conjunto de autorizações

- Autorizações atribuídas por MAC ou DAC

Atribui a cada usuário um ou mais papéis

- O usuário ativa seus papéis conforme necessário

RBAC - *Role-Based Access Control*

Vantagens:

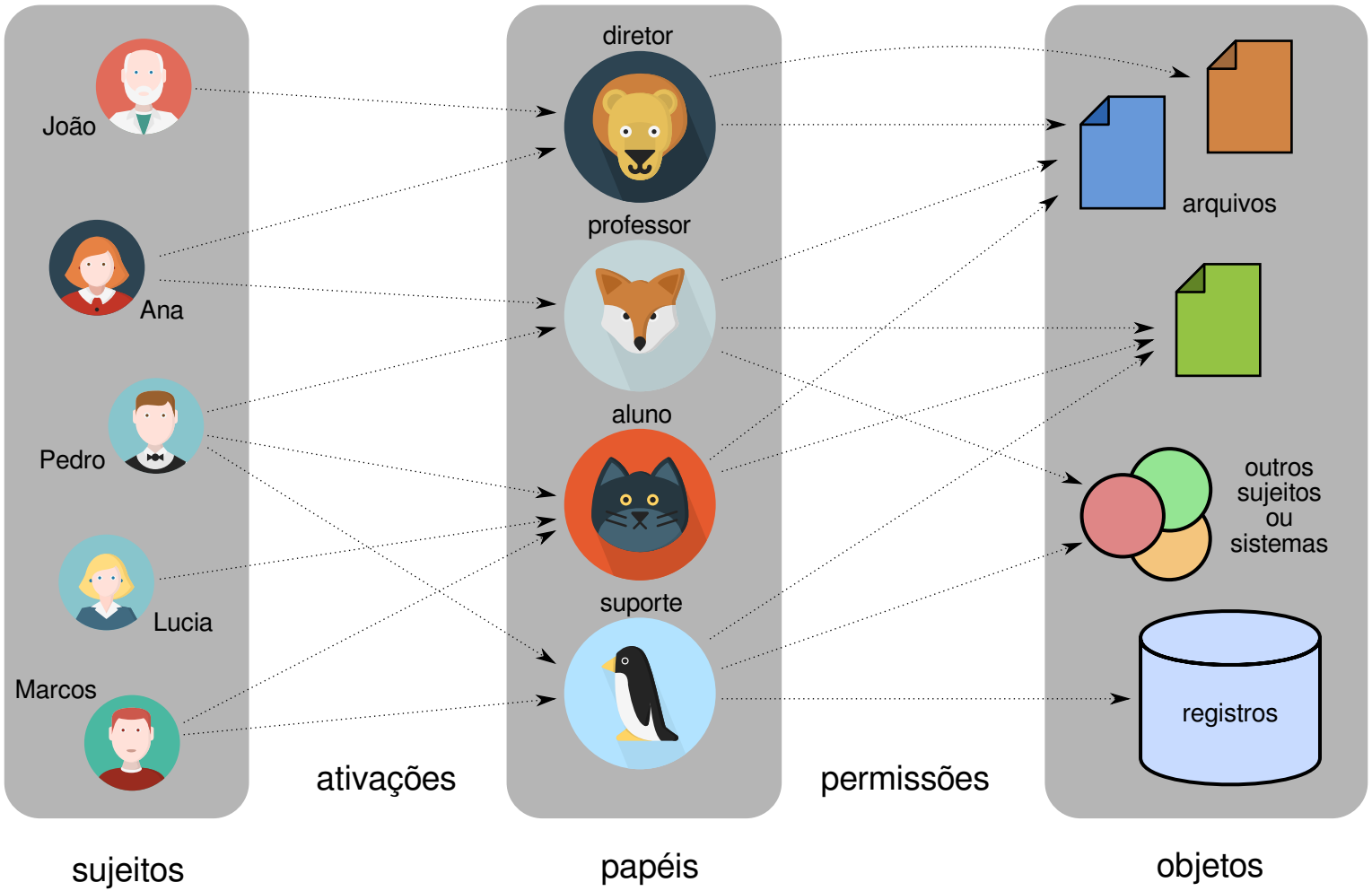
- Permite desacoplar os usuários das permissões
- Um conjunto de papéis bem definido é estável
- A gerência apenas atribui os papéis aos usuários

Exemplos de uso:

- Docker UCP (Universal Control Plane)
- Azure Resource Manager
- Moodle

Proposto pelo NIST em 1992, virou padrão em 2004

Políticas baseadas em papéis



Variantes de RBAC

RBAC hierárquico:

- Os papéis são classificados em uma hierarquia
- Papéis superiores herdam permissões dos papéis inferiores

RBAC com restrições:

- Restrições à ativação de papéis pelos usuários
- Número de usuários que podem ativar um mesmo papel simultaneamente
- Papéis conflitantes não podem ser ativados ao mesmo tempo

Controle de uso

Controle de uso

Controle de acesso convencional:

- Dar **autorizações** para **sujeitos** acessarem **objetos**

Essa forma de concessão de autorização geralmente é estática:

- decidida antes do acesso
- não é revogada enquanto durar o acesso

Problemas:

- Pouco flexível para aplicações modernas: streaming, DRM
- Proliferação de soluções de controle de acesso *ad hoc*

Modelo UCON$_{abc}$

Modelo:

- UCON: Usage CONtrol
- a-b-c: *Authorizations*, *oBligations*, and *Conditions*

Principais características:

- Framework formal consistente
- Avaliação **contínua** dos direitos
- Mutabilidade de atributos

Modelo UCON$_{abc}$

A – **Autorizações**:

- Regras de acesso convencionais, permitem/negam acessos
- Dependem dos **atributos** do sujeito e do objeto

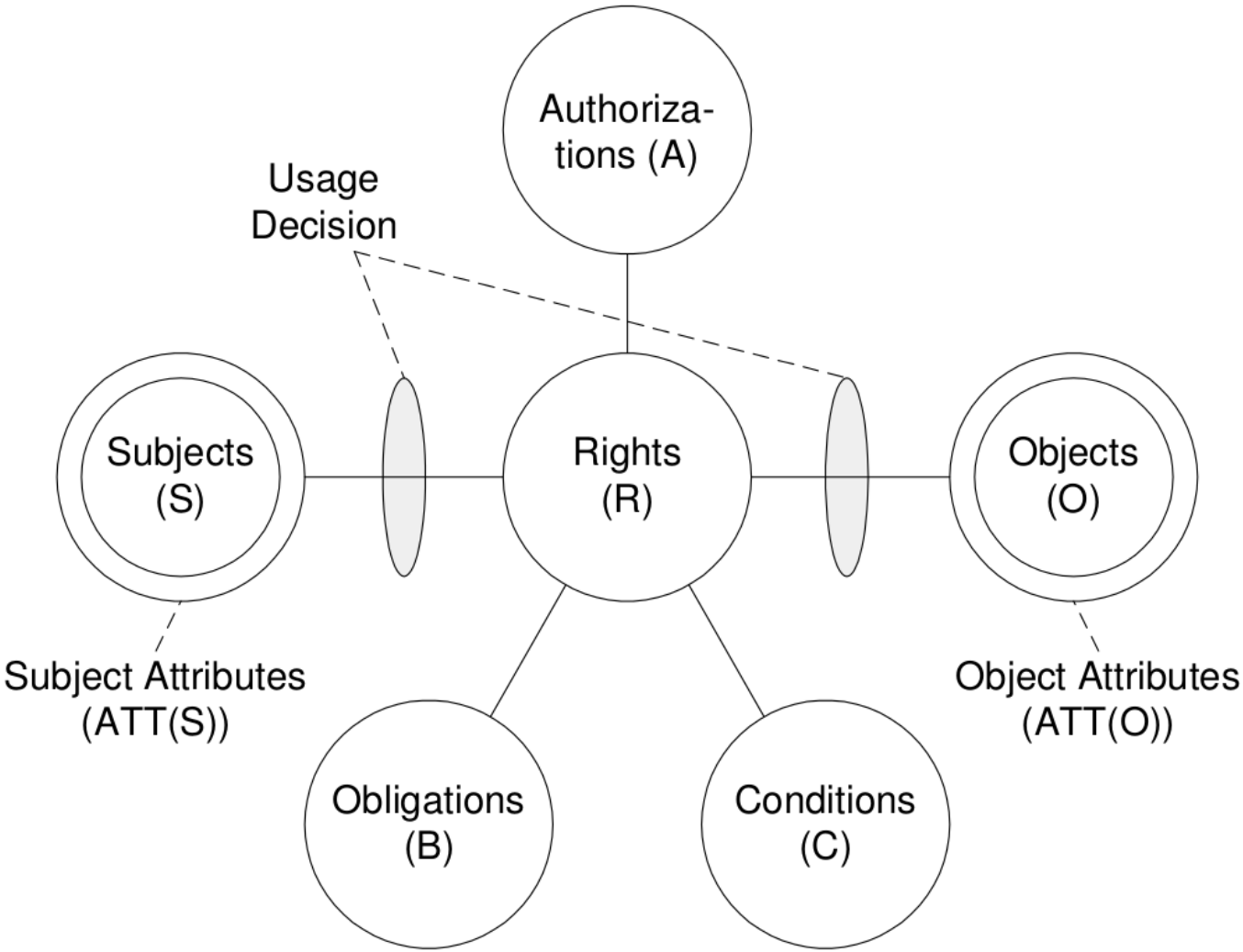
B – **oBrigações**:

- Ações do sujeito para permitir **ou manter** o acesso
- Ex: Aceitar uma EULA; manter uma janela de ads aberta

C – **Condições**:

- Condições externas (ambientais ou do sistema)
- Exemplos: horário do dia, carga no sistema

Modelo UCON$_{abc}$



Modelo UCON$_{abc}$

Atributos do sujeito:

- Identidade
- Papéis ativos
- Nível de segurança
- Crédito disponível
- Lista de capacidades

Atributos do objeto:

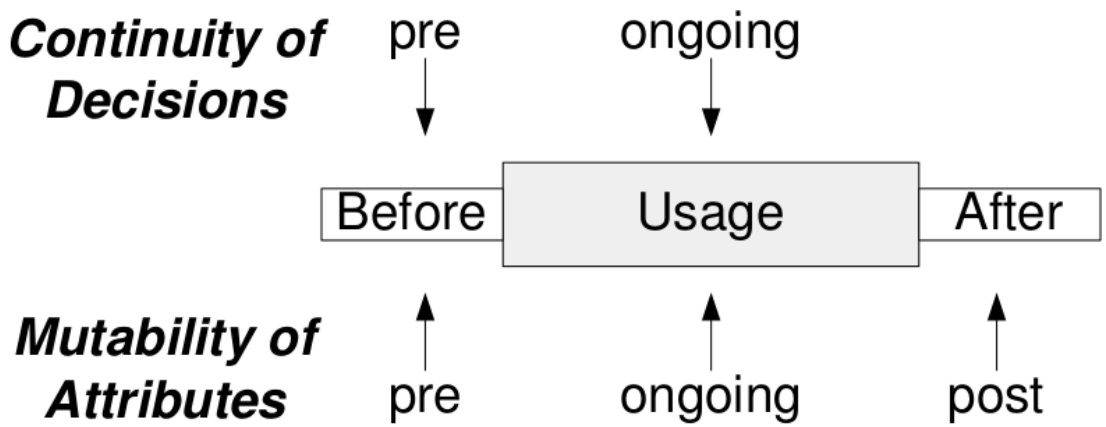
- ID dos proprietários
- Caminho (*file path*)
- Rótulos de segurança
- Lista de controle de acesso

Modelo UCON$_{abc}$

Atributos são **mutáveis**:

- em consequência de regras de acesso avaliadas
- podem revogar acessos em andamento
- Exemplos: número de usuários usando um objeto, créditos de um usuário

Momentos de avaliação de regras e mudança de atributos:



Outros modelos relevantes

Outros modelos de controle de acesso

- ABAC – Attributed-based Access Control (atributos e álgebra booleana)
- Chinese Wall ou Brewer & Nash (usado na área de finanças para gestão de conflitos de interesse)
- Clark-Wilson (controle de integridade)
- Graham-Denning (criação/deleção segura de sujeitos e objetos)
- Take-Grant (controle de acesso por delegação)
- Low Watermark (controle de integridade)
- Lipner Integrity (controle de integridade)
- ... (dezenas de outros)