

# **Fundamentos de Segurança Informática (FSI)**

**2021/2022 - LEIC**

**Manuel Barbosa**  
**[mbb@fc.up.pt](mailto:mbb@fc.up.pt)**

# Aula 7

# Segurança de Sistemas 1

# Princípios Fundamentais

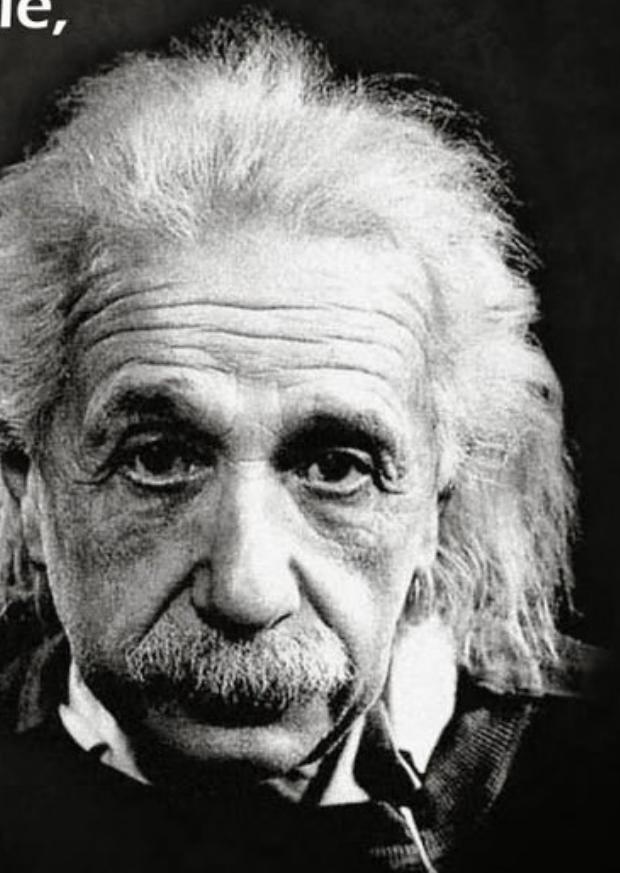
- Quando pensamos sobre a segurança de sistemas complexos, os seguintes princípios devem estar sempre presentes [Saltzer, Schroeder 75]
- Todos continuam válidos, vamos ver os mais relevantes em detalhe
  - *Economy of mechanism*
  - *Fail-safe defaults*
  - *Complete mediation*
  - *Open design*
  - *Separation of privilege*
  - *Least privilege*
  - *Least common mechanism*
  - *Psychological acceptability*
  - *Work factor*
  - *Compromise recording*

# Economia nos Mecanismos

- Keep it simple:
  - um sistema deve ter apenas as funcionalidades (e.g., serviços a correr) necessários
  - de todos os mecanismos de segurança equivalentes
  - devem ser adotados os mais simples
- Facilita a implementação, usabilidade, validação, etc.

*Everything  
should be made  
as simple as possible,  
but not simpler.*

*Albert Einstein*



# Proteção por Omissão

- A configuração de qualquer sistema deve, por omissão (*default*) impor um nível de proteção conservador
- E.g., um novo utilizador deve, por omissão, ter o mínimo de permissões
- "*Fail closed*": por oposição a reverter para uma posição permissiva



Railway semaphore signals. "Stop" or "caution" is a horizontal arm, "Clear to Proceed" is 45 degrees upwards, so failure of the actuating cable releases the signal arm to safety under gravity.

# FortiGate VPN Default Config Allows MitM Attacks



The client's default configuration for SSL-VPN has a certificate issue, researchers said.

# Desenho aberto

- A arquitectura de segurança e os detalhes de funcionamento de um sistema devem ser públicos:
  - não deve basear-se segurança em "security through obscurity"
- Os segredos são parâmetros do sistema, que podem ser alterados:
  - chaves criptográficas, passwords, etc.
- Racional:
  - permite o escrutínio => mais provável sabermos que estamos vulneráveis
  - podemos mudar os segredos sem mudar o sistema

# Security through Obscurity



# Defesa em Profundidade

- Já vimos antes: todos os mecanismos de segurança podem falhar
- Não podemos depositar toda a nossa confiança num só mecanismo:
  - e.g., blindar perímetro e assumir que não existem ameaças internas
- Inerente a qualquer sistema que faça qualquer coisa de útil para o exterior!



# Privilégio Mínimo

- Cada utilizador, compartimento, programa, etc.
  - deve ter apenas os privilégios/permissões essenciais para desempenhar a sua função
  - racional: need-to-know
    - contrariar este princípio amplifica sem necessidade o impacto de uma brecha local na segurança
  - exemplo de violação: correr todos os serviços como *root*

# Separação de Privilégios

- As funcionalidades/utilização de recursos devem ser compartimentadas:
  - cada compartimento deve estar isolado dos outros
  - todos os compartimentos devem encarar os restantes como estando num outro domínio de confiança
- Essencial combinar com defesa em profundidade e privilégio mínimo:
  - comprometer um compartimento tem consequências localizadas
  - um compartimento tem apenas acesso aos recursos de que necessita

# Mediação Completa

- Um sistema gere, invariavelmente, recursos: ficheiros, dispositivos de hardware, memória, etc.
- Mediação completa implica:
  - Para todos os recursos definir uma política de proteção
  - Todos os acessos a todos os recursos são validados relativamente a essa política
- Exemplo que já vimos => memória virtual
  - Todos os processos acedem a um espaço de memória virtual
  - Todos os acessos a memória são mediados pelo sistema operativo
- Veremos mais à frente formas diversas de controlo de acessos = mediação

# **Controlo de Acessos**

# Controlo de Acessos

- *Controlo de acessos* refere-se a uma família de mecanismos de segurança que visam aplicar alguns dos princípios anteriores:
  - Privilégio mínimo: atribuir os privilégios estritamente necessários às entidades que interagem com/no âmbito do sistema
  - Mediação total: garantir que todos os acessos a um recurso são efetuados por entidades com privilégios adequados

# Controlo de Acessos

- Quando tratamos de *controlo de acessos* falamos de
  - Ator (entidade que realiza uma acção)
  - Recurso (sobre o qual se realiza a acção)
  - Operação (o tipo de acção que é realizada)
- A combinação de recurso/operação chama-se uma *permissão*

# Matriz de acessos

- Descreve todos os possíveis acessos:
  - Ator, Recurso, Operação
  - E.g., ler, escrever, executar
- Vantagem: clareza, eficiência
- Problema: tamanho
- As soluções seguintes são alternativas a esta solução, que permitem escalar para sistemas de grande dimensão

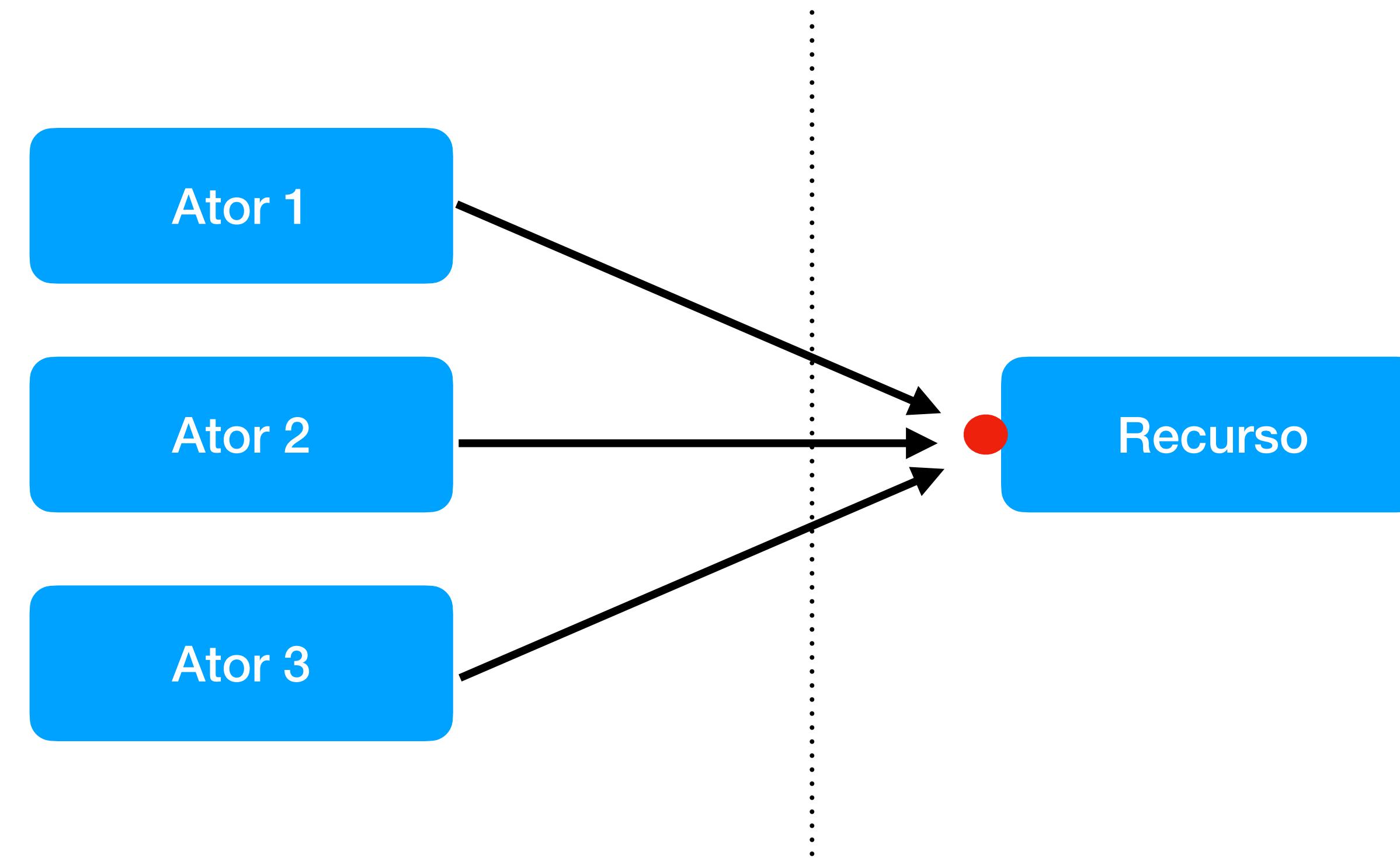
	R1	R2	R3
A1	r	rw	n
A2	rw	n	r
A3	r	r	r

# Listas de controlo de acessos (ACL)

- Para cada recurso, as permissões dos atores sobre esses recursos (omissão = ausência de permissão, intuição = lista de convidados)
- Vantagens:
  - faz-nos pensar sobre como proteger cada recurso individualmente  
=> armazenamento associado ao próprio recurso
  - permite garantir facilmente que um recurso apenas pode ser acedido por um numero limitado de atores (isolamento)
- Desvantagens:
  - Não é possível determinar as permissões que um ator tem sem ver todos os recursos (e.g., para remover um ator)
  - Não há agregação de *perfis* de permissões

	R1	R2	R3
A1	r	rw	n
A2	rw	n	r
A3	r	r	r

# Listas de controlo de acessos (ACL)



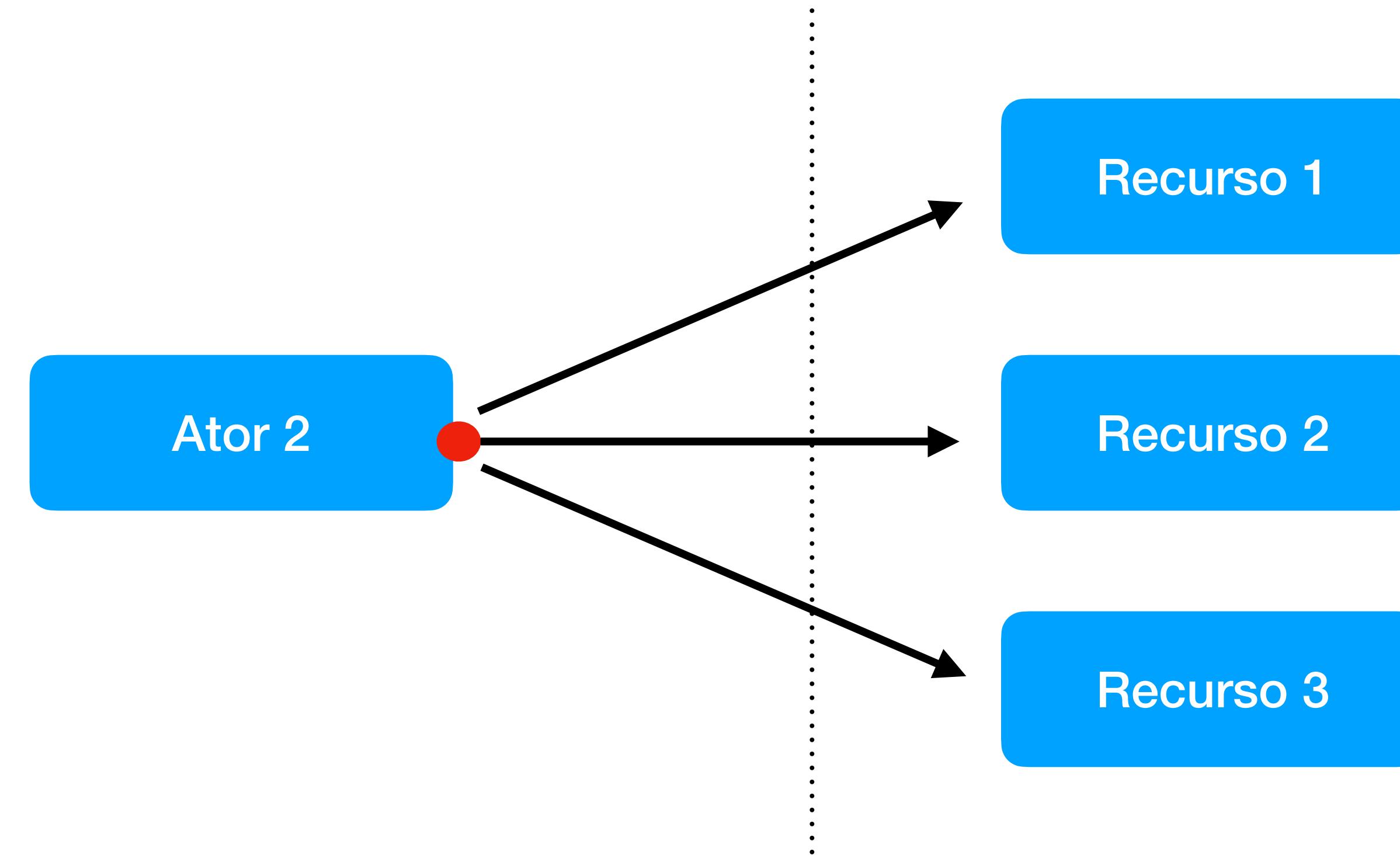
**Inbound Access Control:** gestão orientada a quem é dirigido o acesso

# Listas de permissões (Capabilities)

- Para cada ator, as operações que pode realizar sobre cada recurso (omissão = ausência de permissão, intuição = passe de transportes).
- Vantagens:
  - faz-nos pensar sobre como lidar com cada ator individualmente  
=> armazenamento associado ao ator
  - permite garantir facilmente que um ator apenas acede a um numero limitado de recursos
- Desvantagens:
  - Não é possível determinar todos os atores que podem aceder a um recurso sem percorrer todos os atores
  - Não há agregação de *perfis* de permissões

	R1	R2	R3
A1	r	rw	n
A2	rw	n	r
A3	r	r	r

# Capabilities

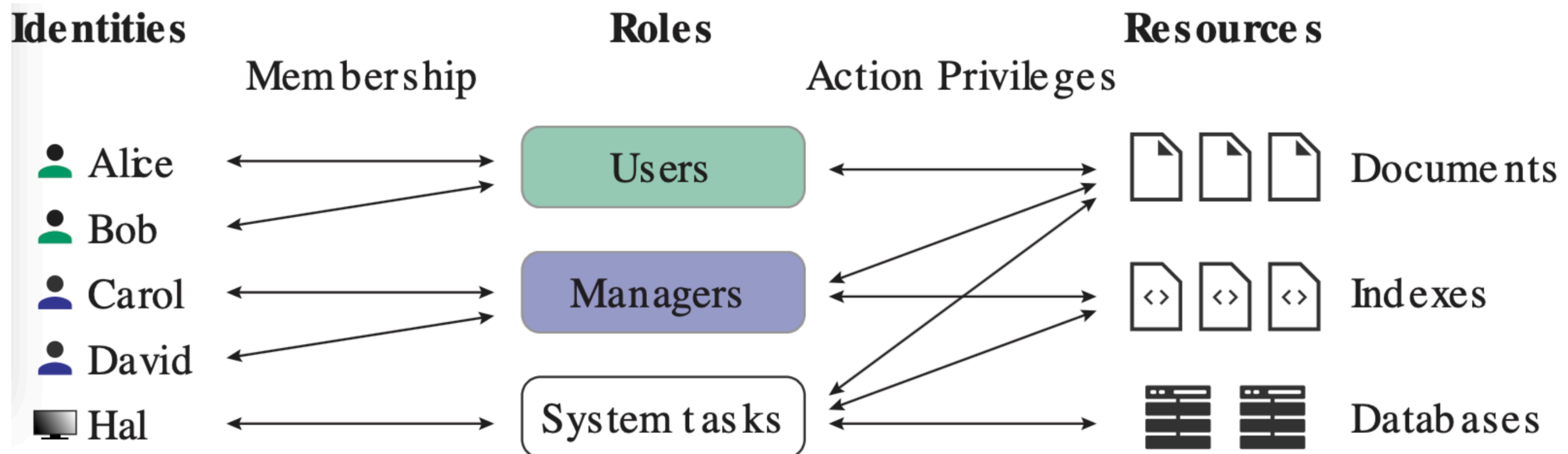


**Outbound Access Control:** gestão orientada a quem inicia o acesso

# Role Based Access Control

- Modelo com dois tipos de relações que permitem separar a gestão de recursos da gestão de atores:
  - Ligação de perfis (*roles*) a conjuntos de permissões (agregação, menos espaço)
  - Ligação de atores a perfis
- As relações entre perfis e permissões são geralmente muito estáveis:
  - são administrados por quem gere recursos => semelhante a ACL
- Os perfis dos atores podem ser mais dinâmicos:
  - são administrados por quem gere atores (e.g., utilizadores) => semelhante a listas de permissões

# Role Based Access Control



# Attribute-based Access Control

- O RBAC é um caso particular de ABAC:
  - atores e recursos têm atributos associados
  - matriz de acessos descreve permissões com base nos atributos:
    - para aceder a recurso com atributo A o ator deve possuir atributo B
    - permite políticas mais expressivas, como por exemplo ter em conta o contexto geográfico e temporal, ou sistemas hierárquicos

# **Sistemas Operativos**

## **Aplicação dos Princípios Anteriores**

# Sistema Operativo

- Interface entre os utilizadores de um computador e o hardware:
  - gestão do acesso a recursos por parte de aplicações
    - armazenamento, processador, memória, I/O, rede, etc.
    - partilha destes recursos por vários utilizadores e aplicações.
    - software que trata operações de baixo nível e oferecem abstrações convenientes para o desenvolvimento de aplicações
  - muito complexos e gerem aspectos críticos para a segurança

# Aspetos de Segurança

- Múltiplos utilizadores com diferentes níveis de acesso:
  - administradores, utilizadores frequentes, utilizadores esporádicos, etc.
  - diferentes necessidades e privilégios relativamente a recursos a utilizar
  - o SO tem de garantir que estes requisitos são cumpridos e, ao mesmo tempo, garantir que não existem comportamentos abusivos
  - os utilizadores são sempre potenciais ameaças, que modelo de ameaças?
  - os recursos são sempre ativos a proteger, que propriedades?

# Aspetos de Segurança

- Múltiplas aplicações/serviços a executar em simultâneo:
  - as aplicações são também potenciais ameaças (aulas sobre controlo)
  - devem estar protegidas de interferência de outras => isolamento
  - esta proteção aplica-se mesmo quando não estão a executar:
    - a informação que armazenam como estado está geralmente em recursos partilhados (e.g., disco)

# Aspetos de Segurança

- Os sistemas operativos modernos visam garantir:
  - isolamento virtual entre utilizadores, aplicações e processos
  - apesar de estarem a partilhar os recursos disponíveis no sistema
  - através de mecanismos de acesso mediado
- Administrar um sistema operativo => configurar estes mecanismos
  - devem aplicar-se os princípios fundamentais: privilégio mínimo, separação de privilégios, etc. => boas práticas