Protección: Control de acceso

Sistemas Operativos Avanzados

Fernando Pérez Costoya

Control de acceso

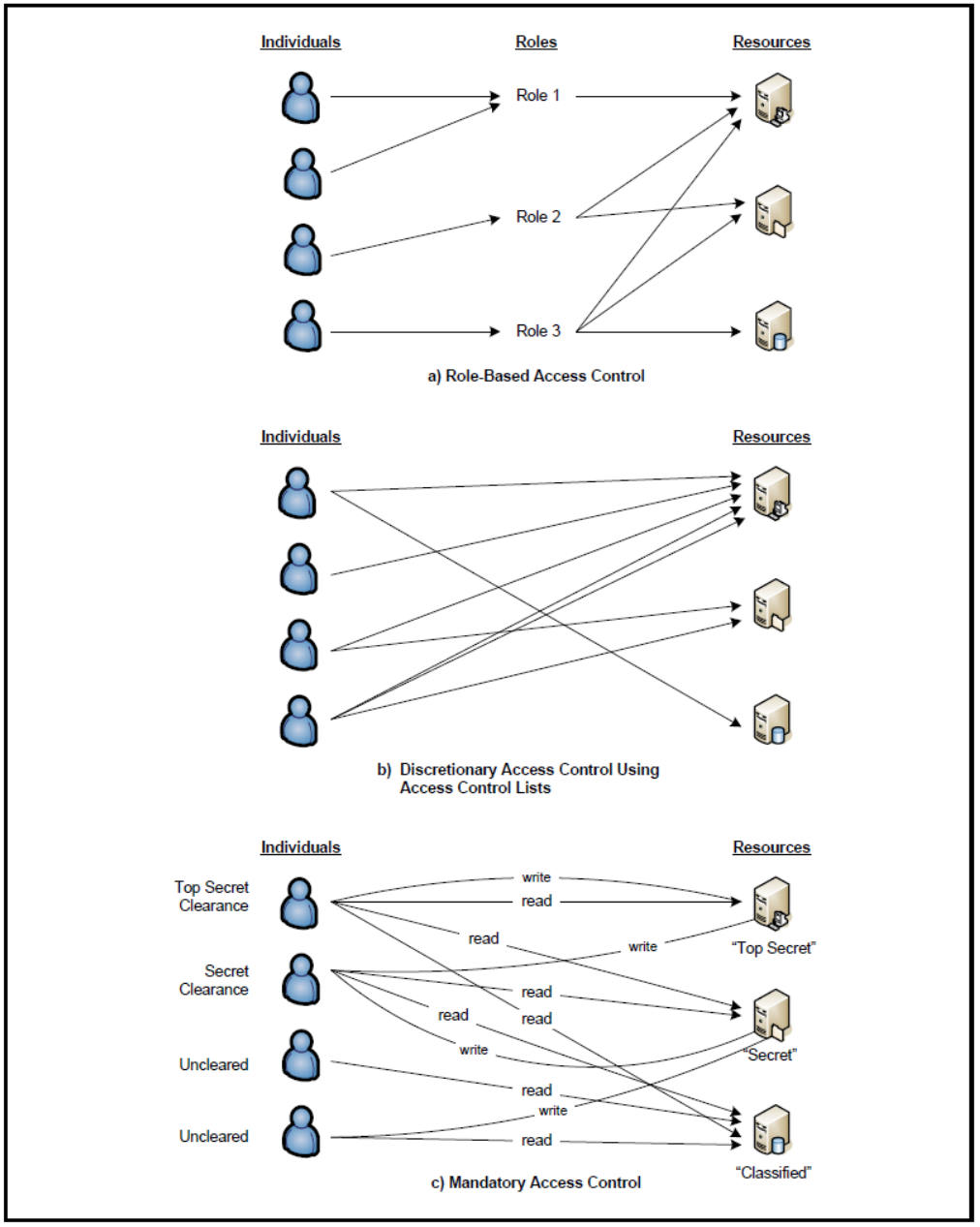
¿SujetoX puede hacer *OperaciónY* sobre *ObjetoZ*?

- Sujetos: usuarios (los procesos de los usuarios)

3 modelos formales principales:

- *Discretionary Access Control* (DAC)
  - Más usado en SO de propósito general (el “clásico”)
  - Familia UNIX, Windows...
- *Mandatory Access Control* (MAC)
  - Usado en entornos de alta seguridad (p.e. militares)
  - SELinux
- *Role-based Access Control* (RBAC)
  - Además de sujetos y objetos, introduce roles
  - Solaris, HP-UX, SELinux...

DAC vs. MAC vs. RBAC (NIST)



DAC

Dueño de Obj otorga a Usu acceso a **discreción**

- En una organización: ¿es realmente Obj suyo?

Matriz de protección (dispersa): sujetos x objetos

- Celda operaciones permitidas para ese Usu y Obj

Implementación habitual: por columnas

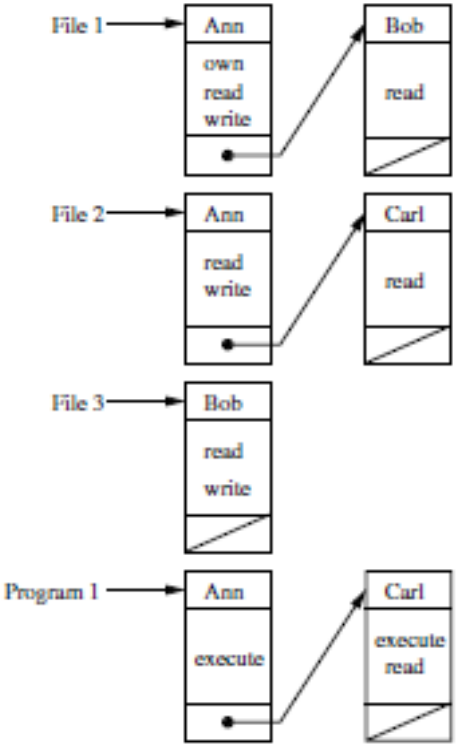
- Listas de control de acceso (ACL)
- Baja usuario ineficiente → revisar todos los recursos

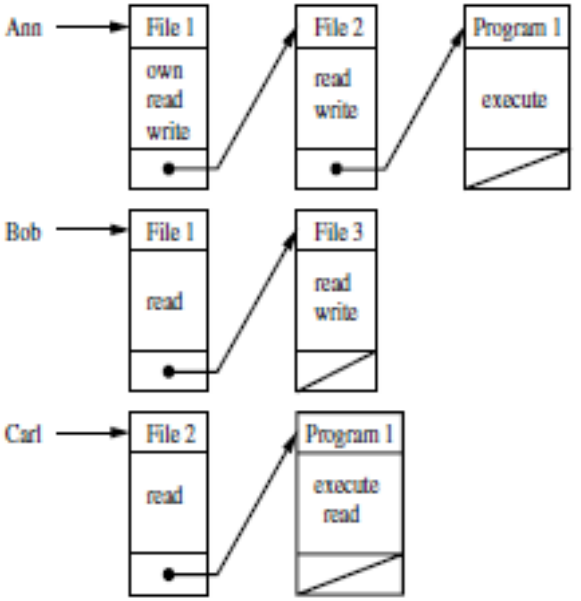
Por filas (menos frecuente): *capabilities*

- Cada sujeto → Lista de Objs accesibles
- Revocar permiso ineficiente →revisar todos los sujetos

ACLs vs. *Capabilities*

	File 1	File 2	File 3	Program 1
Ann	own read write	read write		execute
Bob	read		read write	
Carl		read		execute read





Control de acceso en UNIX

Control de acceso para ficheros: DAC con ACLs

- En UNIX original bits rwx → ACL compacto
- UNIX modernos también ACLs
  - Para otorgar a U acceso a F →insertar U en ACL de F
    - Si solo rwx (no ACLs) no es tan sencillo...
  - Superusuario (SU; *root*) acceso a todos

- Control de acceso para otros recursos:
  - Operaciones privilegiadas (solo SU)
    - Cambiar reloj, configurar interfaz de red, apagar el sistema, *bind* de puertos privilegiados, aumentar prio. proceso…
    - Rompe principio de mínimo privilegio

Ejemplo de ACLs de Linux

```
# crea un fichero y lista ACLs
$ touch f
$ getfacl fichero
# file: fichero
# owner: fperez
# group: fperez
user::rw-
group::r--
other::r--

# añade permiso rw a user1
$ setfacl -m user:user1:rw fichero
$ getfacl f
# file: fichero
# owner: fperez
# group: fperez
user::rw-
user:user1:rw-
group::r--
mask::rw-
other::r--
```

```
# añade permiso r a user2
$ setfacl -m user:user2:r fichero
$ getfacl f
# file: fichero
# owner: fperez
# group: fperez
user::rw-
user:user1:rw-
user:user2:r--
group::r--
mask::rw-
other::r--
# elimina permiso w de user1 y user2
$ setfacl -m m::r fichero; getfacl fichero
# file: fichero
# owner: fperez
# group: fperez
user::rw-
user:user1:rw-          #effective:r--
user:user2:r--
group::r--
mask::r--
other::r--
```


7

Capabilities en Linux

Aplicables solo a ciertas op. privilegiada (no a fich)

Otorga permiso para usar cierta op. privilegiada

- Aumentar prio, puertos privilegiados, ...

Pueden asignarse a ejecutables

- SETUID mejorado

Servicio usa puerto privilegiado no requiere SU

Ejemplo:

$ sudo setcap cap_net_bind_service+ep ./servidor
$ getcap servidor
servidor = cap_net_bind_service+ep
$ ./servidor 222 # servidor usa el puerto privilegiado 222 sin ser superusuario

Limitaciones del DAC

Usuario ejecuta *app* maliciosa (p.e. caballo troya)

Confidencialidad: Fuga de información

- *App* puede filtrar información de ese usuario
  - *cp fichero_secreto /tmp/copia; chmod 444 /tmp/copia*
  - Si SU puede obtener cualquier info. del sistema

Integridad:

- *App* puede modificar información de ese usuario
  - *cp /tmp/falso fichero_secreto*
  - Si SU puede modificar cualquier info. del sistema

MAC

Autoridad única asigna *etiquetas* de seguridad a

- Objetos (ficheros, sockets, tuberías, interfaces red...):
  - Etiqueta refleja grado de sensibilidad de la información
- Sujetos: Etiqueta refleja sus credenciales
- Control acceso depende de etiquetas sujeto y objeto
  - Reglas definen si la operación de acceso está permitida

Puede/suele combinarse con DAC:

- 1º control por DAC → si OK control por MAC

Distintos esquemas MAC alternativos:

- Type Enforcement (*TE*): etiqueta = tipo
  - Reglas de acceso basadas en tipos de sujetos y objetos
- MLS: Problemas confidencialidad e integridad

Multi-Level Security (MLS)

Sujetos y objetos usan etiqueta con 2 valores:

- Asociadas a distintos niveles de seguridad. Ejemplos:
  - *top secret(TS) > secret(S) > confidential(C) > unclassified(U)*
  - *restricted > proprietary > sensitive > public*

Extensión Multi-Category Security (MCS)

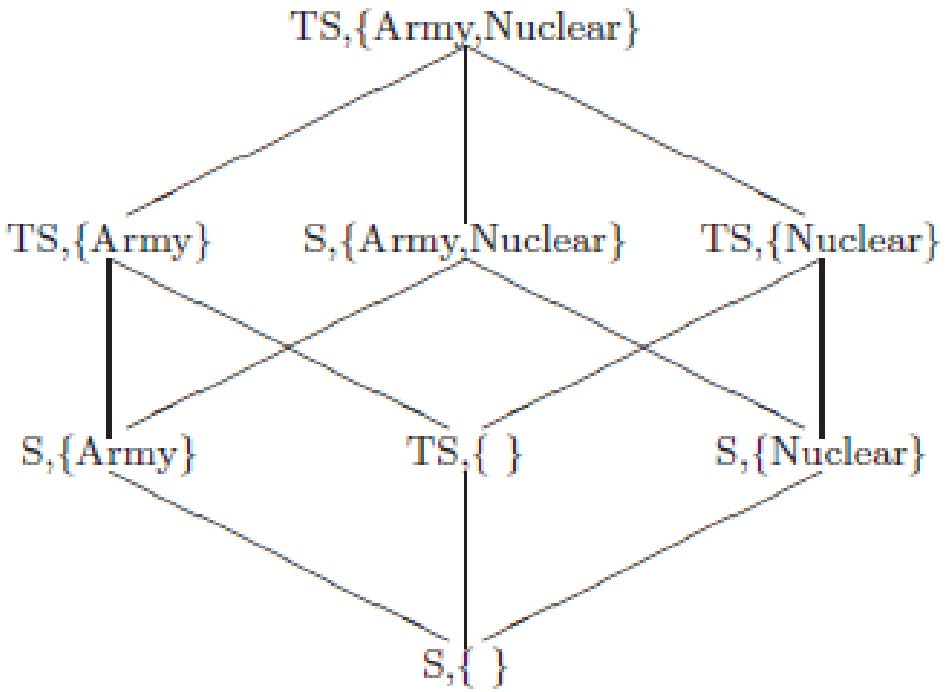
- Asociadas a distintas categoría. Ejemplos:
  - *army, navy, air force* | *financial, administration, research*
- Requerido para satisfacer *need-to-know*
  - A un sujeto solo le conciernen ciertas categorías de objetos
  - Restringe tipo de información accesible por un sujeto

Ejemplos: *suj1→(S,{army,navy}); obj2→(TS,{army})*

Multi-Level Security (MLS)

Rejilla de seguridad.

- Ej: 2 niveles: *TS* > *S* y 2 categorías: *Army, Nuclear*
- Arista ascendente: sujeto>privilegio; objeto>restricción



Access Control: Policies, Models, and Mechanisms. Pierangela Samarati, Sabrina de Capitani di Vimercati

Modelo MAC Bell-La Padula (1973)

Resuelve problema de filtrado de información

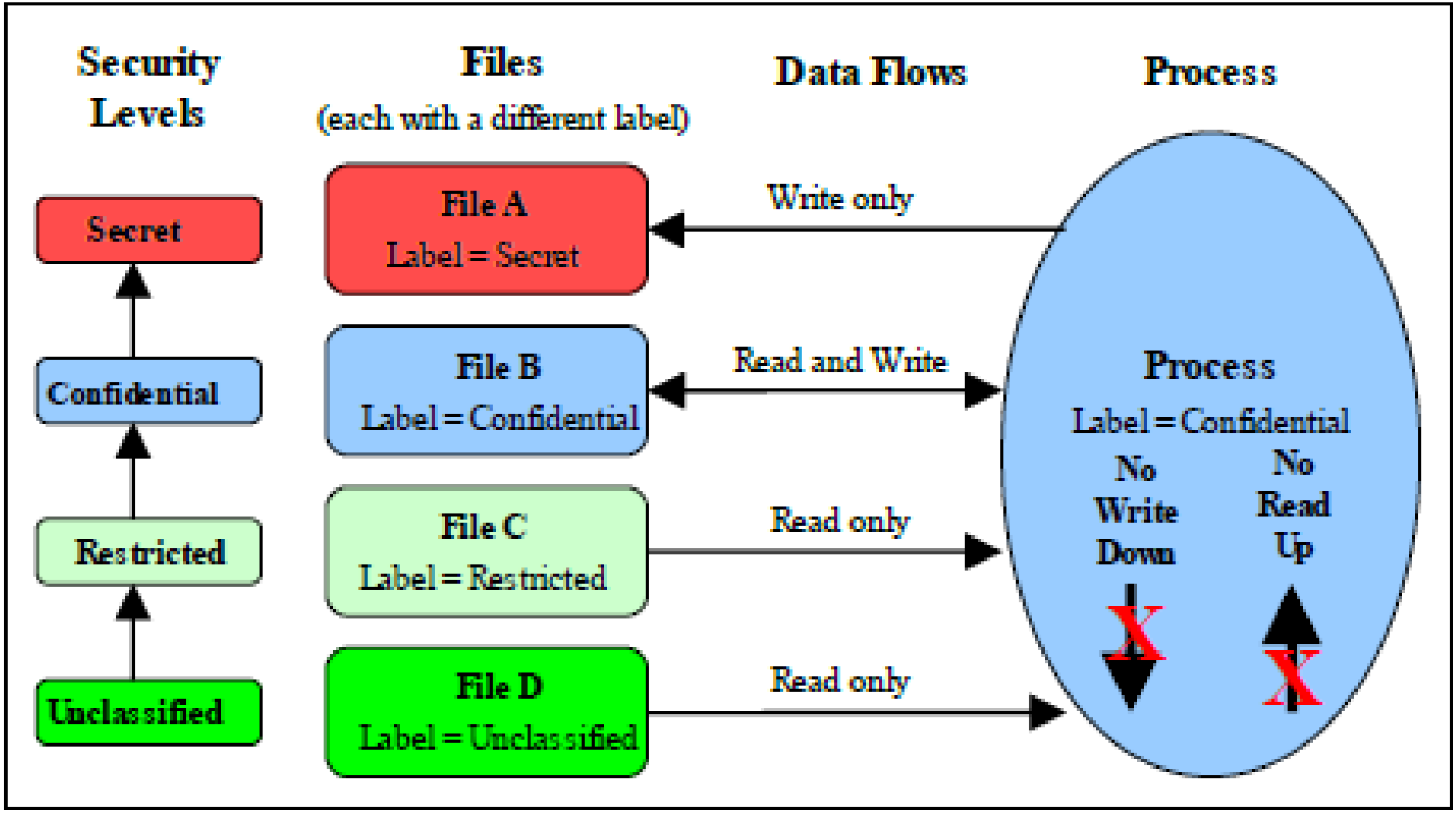
Propiedad simple (*no read-up*)

- Sujeto *S* puede leer objeto *O* $\rightarrow$ $nivel(S) \geq nivel(O)$
  - Y además $categorías(S) \supseteq categorías(O)$

Propiedad * (*no write-down*)

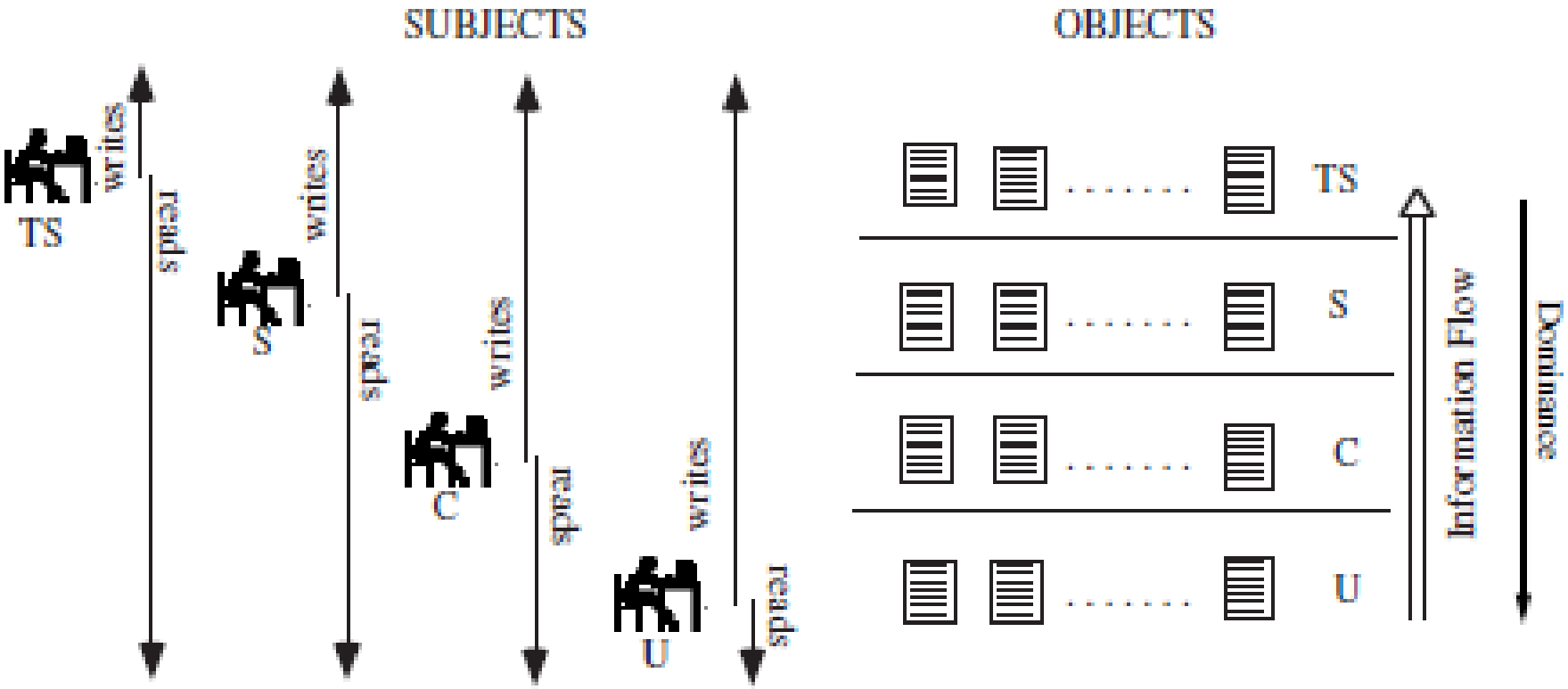
- Sujeto *S* puede escribir objeto *O* $\rightarrow$ $nivel(S) \leq nivel(O)$
  - Y además $categorías(S) \subseteq categorías(O)$
- Problema de integridad. Alternativas:
  - Uso complementario de DAC para limitar acceso
  - Modelo Bibba (menos usado que Bell-La Padula)

Ejemplo modelo Bell-La Padula



The SELinux Notebook (4th Edition) Richard Haines

Evitando fugas de información



Access Control: Policies, Models, and Mechanisms. Pierangela Samarati, Sabrina de Capitani di Vimercati

Modelo MAC Bibba (1977)

Resuelve problema de integridad

no read-down

- Sujeto *S* puede leer objeto *O* $\rightarrow$ *nivel(S)*$\leq$*nivel(O)*
  - Y además *categorías(S)* $\subseteq$ *categorías(O)*

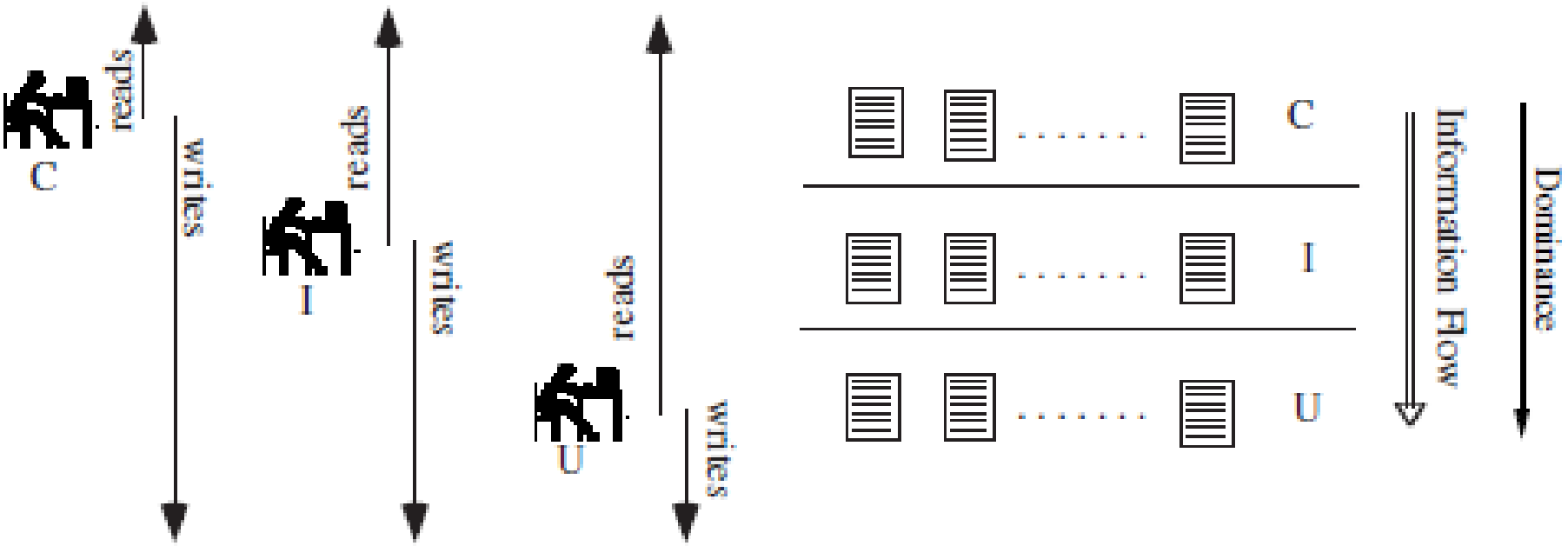
no write-up

- Sujeto *S* puede escribir objeto *O* $\rightarrow$ *nivel(S)*$\geq$*nivel(O)*
  - Y además *categorías(S)* $\supseteq$ *categorías(O)*

Uso combinado de ambos modelos:

- Se asignan 2 etiquetas a sujetos y objetos:
  - Etiquetas de confidencialidad y de integridad

Evitando problemas de integridad



Access Control: Policies, Models, and Mechanisms. Pierangela Samarati, Sabrina de Capitani di Vimercati

RBAC

Autoridad única define roles

- Asignándoles permiso para ciertas operaciones
  - Orientado a operaciones más que a permisos sobre objetos
  - Médico prescribe medicina; enfermero la administra
  - Ej. de Solaris: rol *Operator* → *solaris.system.shutdown*
- Pueden anidarse (médico|enfermero→ pers. sanitario)

Autoridad única asigna a usuario uno o más roles

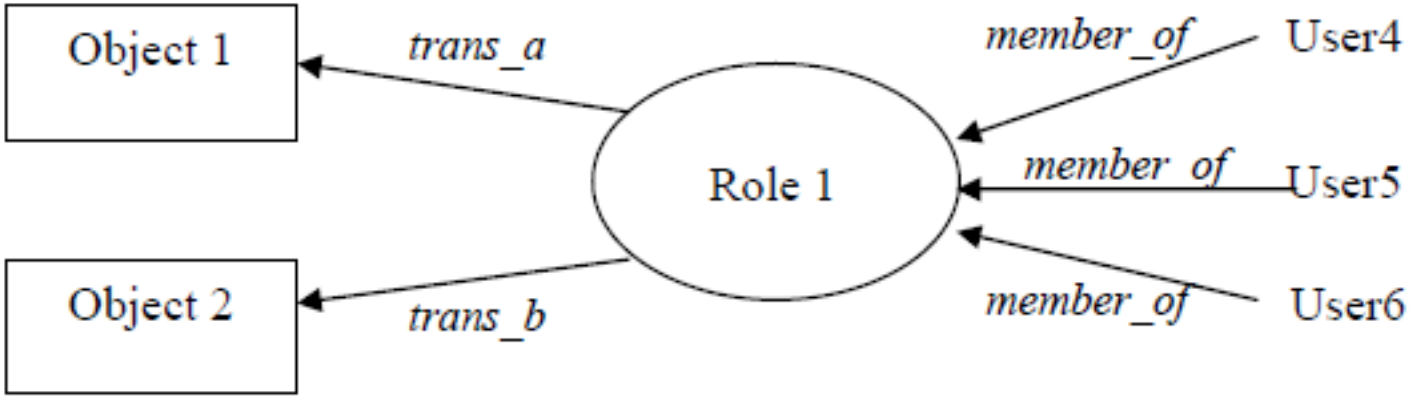
- Facilita mínimo privilegio y delegación administración

Baja usuario eficiente: solo eliminarle de los roles

Cierta similitud con grupos de usuarios:

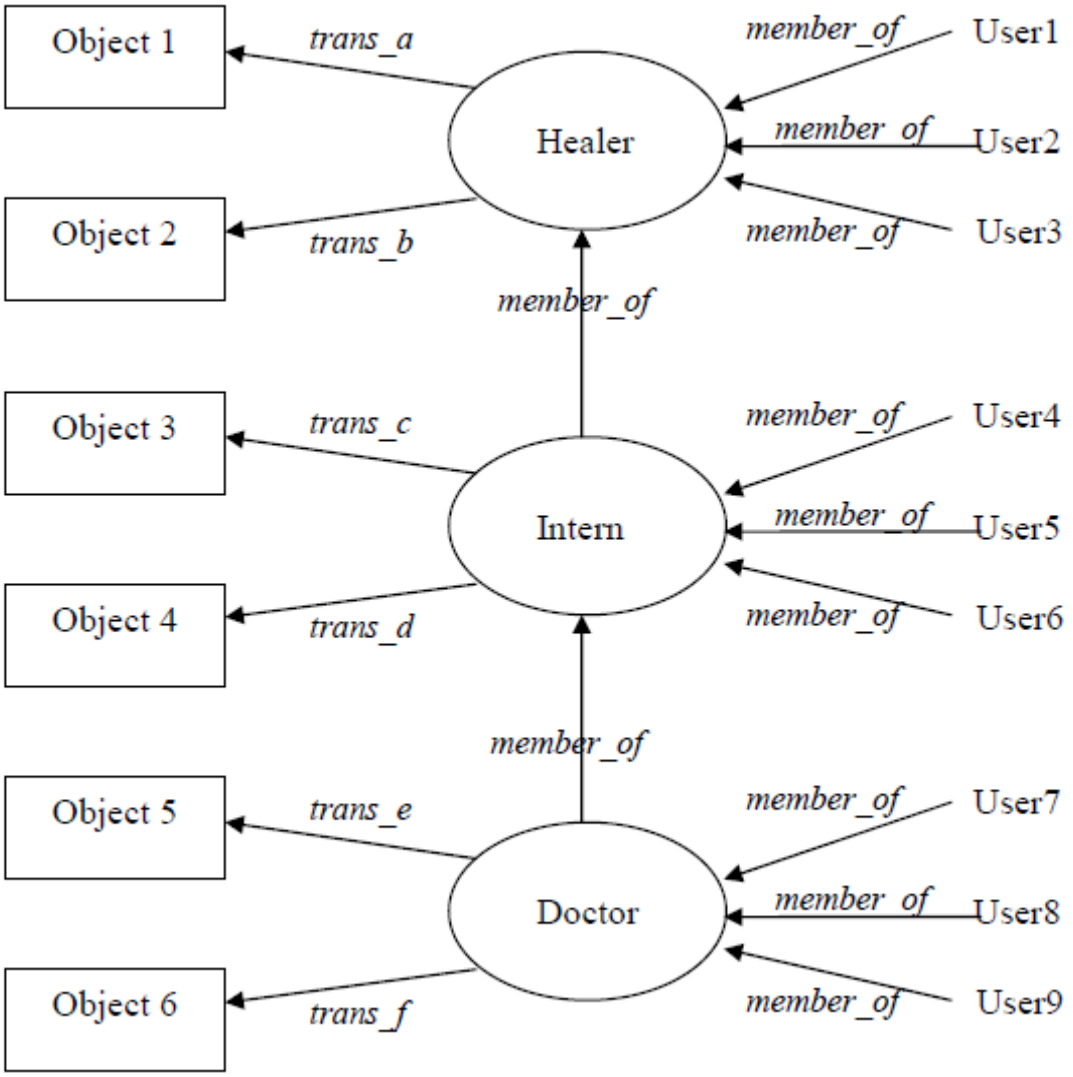
- Pero rol es jerárquico y orientado a operaciones

RBAC



Role-Based Access Controls. David F. Ferraiolo and D. Richard Kuhn

Jerarquía de roles



Role-Based Access Controls. David F. Ferraiolo and D. Richard Kuhn

Security-Enhanced Linux

Definición de la NSA:

“Set of patches to the Linux kernel and some utilities to incorporate a strong, flexible mandatory access control (MAC) architecture into the major subsystems of the kernel”

Security-Enhanced Linux

Implementado como un módulo (LSM)

Presente por defecto en algunas distribuciones

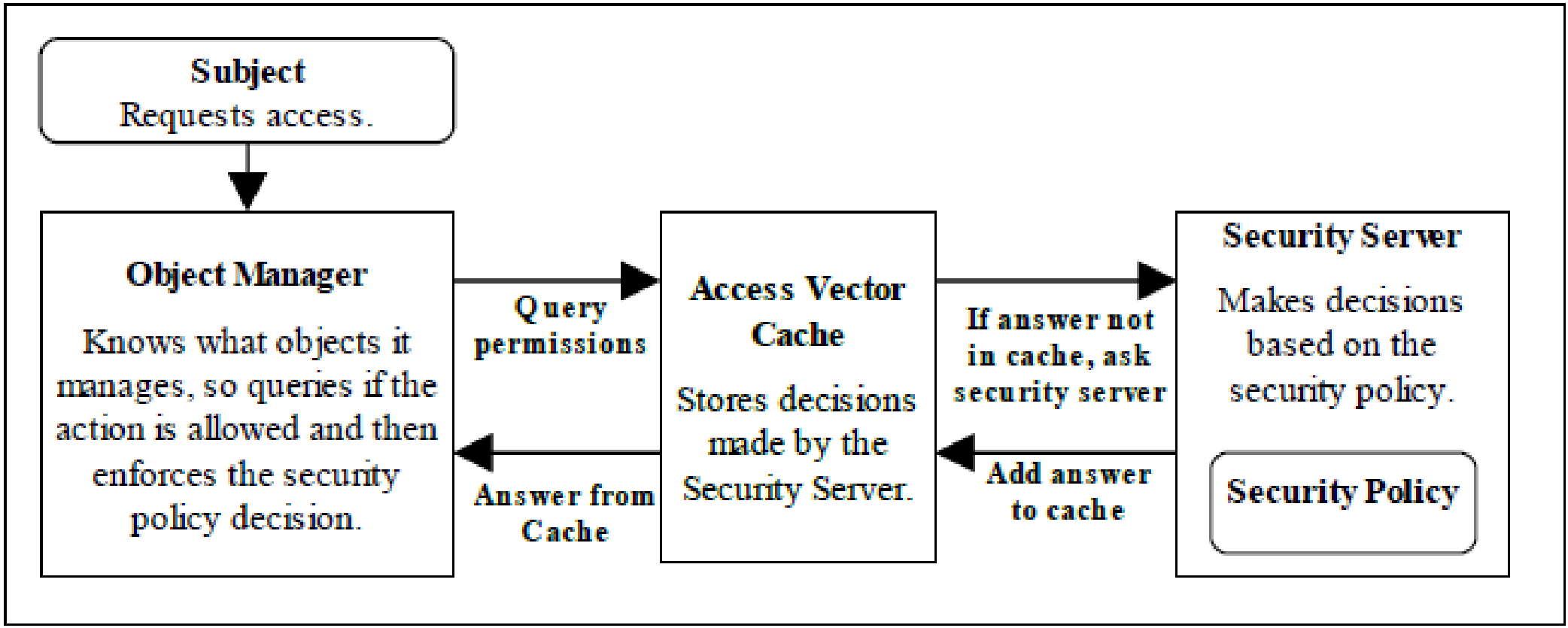
Distintos modos de configuración:

- *enforcing:* Control de acceso de SELinux activo
- *disabled:* Control de acceso de SELinux inactivo
- *permissive:* Control inactivo pero *log* de eventos de seguridad
- *targeted* (solo para ciertos demonios) | *strict* (global)

Soporta DAC, MLS, RBAC, TE y MCS

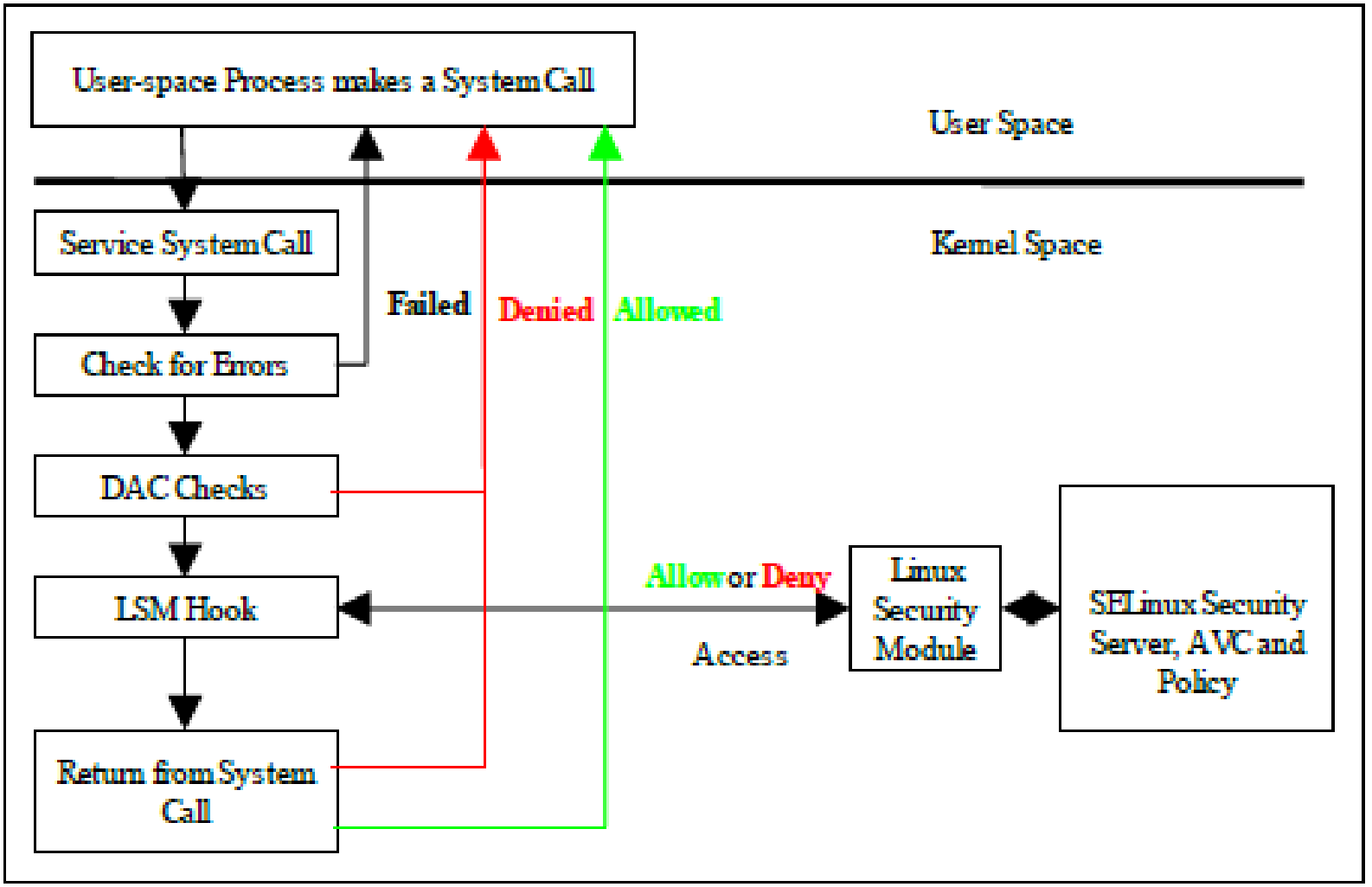
Control de acceso mediante *security contexts*

Arquitectura *SE-Linux*



The SELinux Notebook (4th Edition) Richard Haines

Control de acceso *SE-Linux*



The SELinux Notebook (4th Edition) Richard Haines

SE-Linux: security contexts

Cada sujeto y objeto tienen asignado uno

user:role:type[:range]

Modos de configuración:

- *user:* normalmente distinto a usuario convencional de Linux
  - P.e. *user_u* para usuarios normales y *admin_u* para administradores
- *role*: soporte de RBAC
- *type*: soporte de TE
- range: soporte MLS/MCS
  - Puede especificarse un nivel de seguridad y una categoría o rangos
  - P.e. s1:c0
  - P.e. s0 - s15:c0.c1023