Controllo degli Accessi

Parte I

Indice

Capitolo 1

Introduzione

Il **controllo degli accessi** valuta l'accesso richiesto alle risorse dagli utenti autenticati e, sulla base di *regole di accesso* (definite all'interno) del sistema, determina se l'accesso sia garantito o negato.

Si occupa solamente dell'**accesso diretto**. Si basa su due concetti:

- **Autenticazione/Identificazione** dell'utente che fa la richiesta
  - importante anche per le problematiche di *accountability*; posso analizzare i log per capire chi ha fatto che cosa nel caso ci sia un problema
- **Correttezza delle autorizzazioni** con cui l'accesso viene valutato

1.1 Politiche, modelli, meccanismi

È utile fare una distinzione tra:

- **Politiche:** sono i requisiti di protezione ad alto livello che voglio applicare al mio sistema
- **Model:** viene usato per rappresentare la politica
- **Meccanismi:** implementano la politica con *hw* e *sw*

Risulta utile fare questa distinzione perché comporta dei vantaggi: posso verificare se il modello è corretto rispetto alla politica che ho definito; lo stesso meccanismo può essere usato per implementare politiche o modelli diversi.

1.1.1 Meccanismo

In letteratura prende il nome di *reference monitor*, deve soddisfarre le seguenti proprietà:

- non può essere modificabile; nel caso in cui venga fatto me ne devo accorgere

- non può essere bypassabile
- deve essere confinato ad una specifica parte del mio sistema (non distribuito)
- deve essere abbastanza piccolo per essere soggetto a processi di verifica formale

Il meccanismo deve essere sicuro rispetto ai canali di comunicazione non legittimi:

- **Storage channels:** le parti di memoria, prima di essere rese disponibili ad altri dati, dovrebbero essere *pulite* (se cancello un dato non è che *sparisce* dalla memoria fisica dal computer ...)
- **Covert channels:** canali non intesi per il trasferimento di informazioni che possono essere usati per inferire informazioni

Alcuni principi di design

- *Separazione dei privilegi:* non dare troppo *potere* ad un solo utente
- *Privilegio minimo:* voglio darti il minimo privilegio di cui hai bisogno

1.2 Processo di sviluppo di un AC

Una volta definito il modello, posso verificare due aspetti:

- **Completezza:** verificare che hai rappresentato tutti i requisiti di sicurezza della politica
- **Consistenza:** dev essere privo di contraddizioni (un utente ha sia accesso/negazione per una risorsa)

Capitolo 2

Discretionary (DAC) policies: approcci base

Sono politiche basate su:

- **identità** degli utenti
- definizione di regole di accesso (**autorizzazioni**), che stabiliscono *chi può fare che cosa*

Definite *discrezionale* perché gli utenti che sono proprietari dei dati possono amministrarli come vogliono, *a loro discrezione*; tipicamente, non ho un unico amminstratore, ma ci sono più amministratori proprietari delle risorse: è in mano a qualcuno stabilire chi può accedere o meno alle risorse (non è escluso avere un unico amministratore).

2.1 Un esempio di modello

Si usa la *matrice degli accessi*, è una rappresentazione astratta della politica di protezione del sistema.
Formalmente, è caratterizzato da una tripla (S, O, A) che rappresenta lo stato del sistema, dove:

- S è il set degli utenti
- O è il set delle risorse, dove $S \subset O$ (un soggetto può essere anche un processo, e può essere anche una risorsa ...)
- A è la matrice, dove:
  - le righe corrispondono ai soggetti
  - le colonne corrispondono agli oggetti
  - $A[s, o]$ riporta i privilegi di s su o

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

I cambi di stato del sistema vengono fatti con dei comandi che chiamano delle **operazioni primitive**:

- `enter` r into $A[s, o]$
- `delete` r from $A[s, o]$
- `create` subject s
- ...

Sono della forma:

command $c(x_1, \ldots, x_k)$
  if r_1 in $A[x_{s_1}, x_{o_1}]$ and
    r_2 in $A[x_{s_2}, x_{o_2}]$ and
    .....
    $r_m$in $A[x_{s_m}, x_{o_m}]$
  then op_1
    op_2
    .....
    op_n
end

Un esempio:

command CREATE(subj,file)
  create object file
  enter Own *into* A[subj,file] **end.**

command CONFER$_{\text{read}}$(owner,friend,file)
  if Own in A[owner,file]
    then *enter* Read *into* A[friend,file] **end.**

command REVOKE$_{\text{read}}$(subj,exfriend,file)
  if Own in A[subj,file]
    then *delete* Read *from* A[exfriend,file] **end.**

2.2 Trasferimento dei privilegi

Il proprietario dei dati può dare il privilegio anche ad altri utenti. Può rappresentato in modo formale in due modi differenti:

- **Copy flag (*):** il soggetto trasferisce il privilegio ad altri; mantiene il privilegio

```
command TRANSFER_read(subj,friend,file)
        if Read* in A[subj,file]
          then enter Read into A[friend,file] end.
```

- **Transfer-only flag(+):** il soggetto trasferisce ad altri il privilegio ma perde l'autorizzazione

```
command TRANSFER-ONLY_read(subj,friend,file)
        if Read+ in A[subj,file]
          then delete Read+ from A[subj,file]
               enter Read+ into A[friend,file] end.
```

Partendo da uno stato *sicuro*, non deve accadere che applicando una o più operazioni si finisca in uno stato non sicuro.

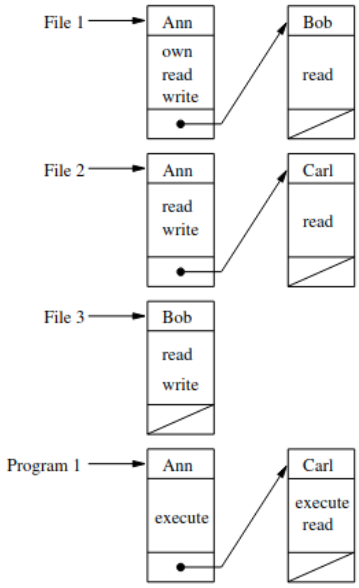
2.3 Implementazione della matrice

La matrice è spesso sparsa, salvarla sarebbe uno spreco di memoria. Ci sono diversi approcci alternativi:

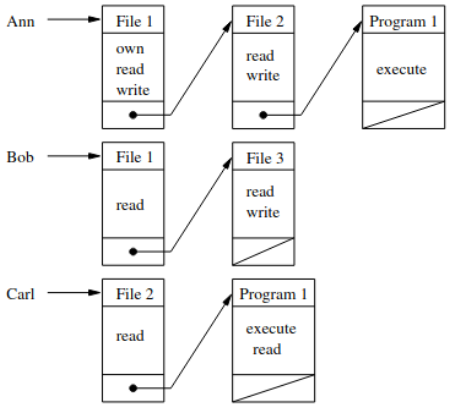
- **Tabella di autorizzazione**; tabella di tuple (S, O, A) non nulle

User	Access mode	Object
Ann	own	File 1
Ann	read	File 1
Ann	write	File 1
Ann	read	File 2
Ann	write	File 2
Ann	execute	Program 1
Bob	read	File 1
Bob	read	File 2
Bob	write	File 2
Carl	read	File 2
Carl	execute	Program 1
Carl	read	Program 1

- **Access Control Lists (ACLs)**: store by column; ad ogni risorsa associo una lista che mi dice gli utenti quali operazioni possono fare



- *Capability lists*; store by row; sono come le precedenti ma vengono fatti storando per utenti invece che per risorse. Sono state soppraffatte dalle ACLs



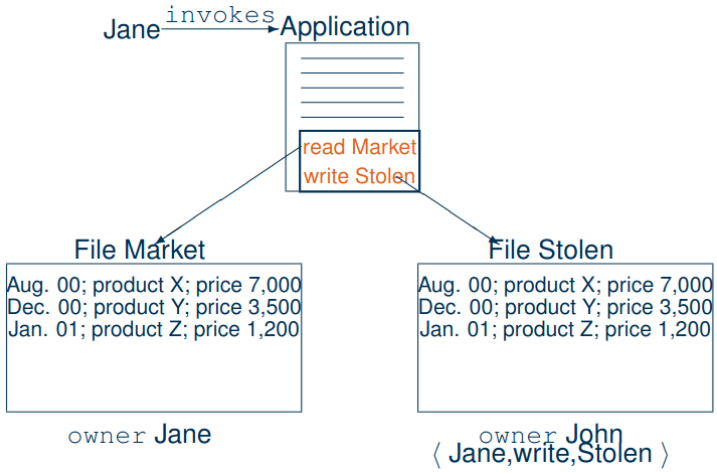
ACLs vs Capability lists

- non richiedono autenticazione del soggetto, ma richiedono la possibilità di verificare che non siano state impropriamente modificate . . . difficile da verificare (per questo non hanno avuto grande successo)

- le ACLs funzionano meglio quando fare delle operazioni di revoca per oggetto (ovviamente viceversa se devo fare revoche per soggetto)

2.4 Debolezze di DAC

Consentono il controllo solo sull'accesso **diretto**. Sono vulnerabili ai *trojan horses*, ovvero accessi indiretti.



L'idea è che vengono lasciate delle *operazioni nascoste* in una applicazione per poter fare delle operazioni che normalmente non si avrebbe l'autorizzazioni di fare.

→ è un accesso indiretto; è il processo che Jane sta eseguendo a chiedere l'accesso ai file

Capitolo 3

Mandatory (MAC) policies

Partono dall'assunzione che c'è una differenza tra *utente* e *soggetto*; è ciò che serve per bloccare i *trojan horses*:

- **Utente:** essere umano (di cui mi fido)
- **Soggetto:** processo nel sistema; opera per conto dell'utente; **non sono fidati**

La politica più comune sono quelle **multilivello**: ogni soggetto e oggetto sono classificate con una etichetta. Si differenziano in politiche che si focalizzano su:

- confidenzialità (Bell La Padula)

  oppure

- integrità (Biba)

3.1 Classificazione di sicurezza

Ogni soggetto ed oggetto è associato ad una coppia di elementi:

- **Livello di sicurezza:** livelli su cui è definita una relazione d'ordine totale (li posso mettere *in fila*).

  $$Secret > Confidential > Unclassified$$

- **Categoria:** insieme di elementi su cui non è definita alcuna relazione di ordinamento; serve per partizionare aree differenti del sistema. Ad esempio, l'università ha un sacco di informazioni di vario tipo: anagrafiche, finanziare, accademiche, ...

  Hanno l'obiettivo di classificarle in classi diverse. Viene fatta sia lato oggetto che lato soggetto.

La combinazione di queste due permette di definire una **relazione di dominanza:**

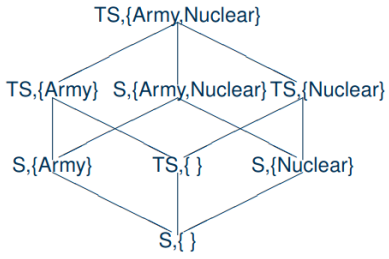
$$(L_1, C_1) \geq (L_2, C_2) \Leftrightarrow L_1 \geq L_2 \wedge C_1 \supseteq C_2$$

Questa relazione soddisfa una serie di proprietà che, in matematica, permette di formare un *reticolo* (quando combinata fra tutte le classi); nel nostro caso parliamo di **reticolo di classificazione**.

- Reflexivity of $\succeq$ $\quad \forall x \in SC : x \succeq x$
- Transitivity of $\succeq$ $\quad \forall x, y, z \in SC : x \succeq y, y \succeq z \Longrightarrow x \succeq z$
- Antisymmetry of $\succeq$ $\quad \forall x, y \in SC : x \succeq y, y \succeq x \Longrightarrow x = y$
- Least upper bound $\quad \forall x, y \in SC : \exists\, !z \in SC$
  - $z \succeq x$ and $z \succeq y$
  - $\forall t \in SC : t \succeq x$ and $t \succeq y \Longrightarrow t \succeq z$.
- Greatest lower bound $\quad \forall x, y \in SC : \exists\, !z \in SC$
  - $x \succeq z$ and $y \succeq z$
  - $\forall t \in SC : x \succeq t$ and $y \succeq t \Longrightarrow z \succeq t$.

Esempio di reticolo di classificazione:

Levels: Top Secret (TS), Secret (S)
Categories: Army, Nuclear



- lub(⟨TS,{Nuclear}⟩,⟨S,{Army,Nuclear}⟩) = ⟨TS,{Army,Nuclear}⟩
- glb(⟨TS,{Nuclear}⟩,⟨S,{Army,Nuclear}⟩) = ⟨S,{Nuclear}⟩

3.2 Semantica della classificazione di sicurezza

- **Classi di sicurezza**
  - associato ad un *soggetto*, rilfette la fiducia verso quell'utente; quanto mi fido di quell'utente
  - associato ad un *oggetto*, riflette la sensisibilità dell'informazione
- Le **categorie** definiscono l'area di competenza di utenti e dati.

3.3 Bell La Padula

È un modello che si preoccupa della confidenzialità (e non del resto).
Considerando di essere in un ambiente multilivello, l'**obiettivo** è prevenire flussi di informazioni ai livelli più bassi o a classi incomparabili.

- ***Simple property:*** un soggetto s può leggere un oggetto o solo se $\lambda(s) \geq \lambda(o)$
- ****-property:*** un soggetto s può scrivere un oggetto o solo se $\lambda(o) \geq \lambda(s)$

$\Rightarrow$ **NO READ UP**
$\Rightarrow$ **NO WRITE DOWN**
Se sono Secret, e scrivo un file secret in uno Top-Secret, non è mica un problema per la confidenzialità. Potrebbe esserlo se scrivo secret in un file Unclassified (potrebbe causare problemi a livelli di integrità, ma ci stiamo occupando solo di confidenzialità).