PROTEZIONE

Protezione : *meccanismo* per controllare l'accesso dei processi alle risorse di un sistema di calcolo.

Obiettivo della protezione: assicurare che ciascun processo usi le risorse del sistema solo **in modi consistenti** con le ***politiche*** stabilite per il loro uso.

• **Politiche di uso delle risorse**. Possono essere fissate durante la progettazione del sistema, durante la sua gestione o definite dagli utenti per proteggere i loro file e programmi.

• **Meccanismo di protezione:** è messo a disposizione dal S.O. e può essere utilizzato a livello di applicazione.

Separazione tra meccanismi e politiche

- Il meccanismo descrive ***come*** la protezione deve essere realizzata, la politica descrive ***cosa*** deve essere protetto.
- Un meccanismo di protezione deve essere in grado di realizzare una varietà di politiche.

Esempio:

- UNIX fornisce un meccanismo per definire per ciascun file i tre bit di *read, write e execute* per il proprietario del file, il gruppo e gli altri.
- L'utente (o l'amministratore) definisce il valore di ogni bit.

DOMINIO DI PROTEZIONE

- Sistema di elaborazione come insieme di **processi e di oggetti (*risorse*)**. (Hardware: segmenti di memoria, stampanti, dischi etc: Software: file, programmi, semafori etc.).
- Ogni oggetto ha un **unico nome** ed un insieme di operazioni con le quali può essere manipolato.
- Un processo può accedere **solo** a quelle risorse per le quali è autorizzato; inoltre, **in ogni istante**, deve poter accedere **solo a quelle risorse necessarie per compiere la sua funzione.** (principio *"need to know":* limita il danno che un processo con errori può creare nel sistema)

Esempio: processo P chiama una procedura A. A deve poter accedere alle sue variabili e parametri formali passati (e non a tutte le variabili del processo P).

Dominio di protezione

- Un processo opera su un *dominio di protezione* che specifica le risorse che il processo può usare.
- Ogni dominio definisce un insieme di oggetti ed i tipi di operazione che possono essere eseguiti sugli oggetti.
- **Diritto di accesso** ➔ possibilità di eseguire una operazione su un oggetto

➔ Il Dominio è un insieme di coppie:

<nome oggetto, insieme di diritti>

- **Spesso, ogni dominio corrisponde a un utente.**

ESEMPIO

- **dominio D** < file F, (read, write)>
- **Domini disgiunti o domini con diritti di accesso in comune**. (Corrisponde alla possibilità di due o più processi di effettuare alcune operazioni comuni su un oggetto condiviso)

D1	**D2**	**D3**
<O3, (read, write>		< O1, (execute) >
<O1, (read,write>	<O2, (write) >	<O4, (print) >
<O2, (execute) >	<O4, (print) >	<O3, (read) >

STRUTTURE DI PROTEZIONE DINAMICA

- L' associazione tra un processo ed un dominio può essere ***statica* o *dinamica***.

Statica: l'insieme delle risorse disponibili ad un processo rimane fisso durante il suo tempo di vita.

➔ Occorre un meccanismo per cambiare il contenuto di un dominio.

Esempio: Un processo ha diritto di accesso "lettura" in una fase e "scrittura " in un'altra, sullo stesso oggetto.

- Non e` opportuno inserire entrambi nel dominio (principio del "need to know") ➔ Occorre che il contenuto della matrice degli accessi venga *modificato.*

Dinamica: occorre un meccanismo per consentire il passaggio da un dominio all'altro del processo che, in questo modo, acquisisce diritti di accesso diversi.

- Quando un processo passa da un dominio ad un altro, viene eseguita un'operazione (***switch, o enter***) su un oggetto (che rappresenta il dominio). Si può controllare questa operazione inserendo i domini stessi tra gli oggetti del sistema.

- Creando nuovi domini ed eseguendo commutazioni di dominio è possibile far eseguire un processo in **diverse fasi**, con un numero di risorse e operazioni su di esse **variabili.**

DOMINIO

E’ un concetto astratto che può essere realizzato in vari modi.

- **Un dominio per ogni utente**. L’insieme degli oggetti cui l’utente può accedere dipende *dall’identità dell’utente*. Il cambio di dominio è legato all’identità dell’utente (avviene quando cambia l’utente). Tutti i processi generati da un utente operano all’interno dello stesso dominio.
- **Un dominio per ogni processo**. Ogni riga descrive gli oggetti ed i diritti di accesso per un processo. Il cambio di dominio corrisponde all’invio di un messaggio ad un altro processo ed al blocco in attesa di una risposta.
- **Un dominio per ogni modulo di programma**. Il cambio di dominio corrisponde al passaggio da un modulo ad un altro.

Esempi di cambio di dominio

Standard dual mode (monitor-user mode):

- Due domini di protezione:quello dell'utente (*user mode*) e quello del kernel (*monitor o kernel mode*)
- Cambio di dominio (*switch*) determinato dalle system call
- Quando l'utente deve eseguire una istruzione privilegiata (accesso ai file, alle funzioni di rete, generazione dei thread etc.) avviene un cambio di dominio.
- Non consente la protezione tra utenti. Insufficiente per la multiprogrammazione.

PROTEZIONE IN UNIX

- **Dominio associato all'utente.** Il cambio di dominio corrisponde al **temporaneo** scambio di identità tra utenti.
- Ad ogni file sono associati l'**identificazione** del proprietario (***user-id***) e un bit di dominio (***set-uid****)*.
- Quando un utente A *(user-id=A*) inizia l'esecuzione di un file il cui proprietario è B (*user-id=B*) ed il file ha *set-uid=on*, *user-id* di A è posto uguale a B temporaneamente.
- Quando l'esecuzione termina sono ripristinate le condizioni iniziali.

Problema (?). Se un utente potesse creare un file eseguibile con *user-id=root* e con *set-uid=on* , l'utente potrebbe diventare *root* ed avere controllo completo del sistema.

[*E` possibile?*]

MATRICE DEGLI ACCESSI

Rappresenta un modello di protezione

oggetti / domini	F1	F2	F3	printer
D1	read		read	
D2				print
D3		read	execute	
D4	read write		read write	

Access (i,j) definisce l'insieme delle operazioni (diritti di accesso) che un processo che opera nel domino Dj può chiamare sull'oggetto Oj.

MECCANISMI E POLITICHE

- La matrice degli accessi rappresenta un **meccanismo** per realizzare una **molteplicità** di politiche.
- Il **meccanismo** consente di assicurare che un processo che opera nel dominio *Dj* può accedere solo agli oggetti specificati nella riga *i* e solo con i diritti di accesso indicati.
- Quando un'operazione *M* deve essere eseguita nel dominio *Di* sull'oggetto *Oj*, il meccanismo consente di controllare che *M* sia contenuta nella casella *(i,j).* In caso affermativo l'operazione può essere eseguita. In caso negativo si ha una situazione di errore.
- Le decisioni di **politica** comportano la decisione su quali:
    - diritti di accesso inserire negli elementi della matrice(utente)
    - quale dominio associare a ciascun processo.

MODIFICA DELLA MATRICE

- Le operazioni sui domini e sulla matrice illustrano la capacità del modello di consentire la realizzazione ed il controllo di una protezione dinamica.

- Le **decisioni di politica** relative a quali domini devono aver accesso a quali oggetti e secondo quali modalità devono essere prese dai progettisti di sistema e dagli utenti.

Cambio di dominio

oggetti / domini	F1	F2	F3	disco	stampante	**D1**	**D2**	**D3**	**D4**
D1	read		read				**switch**		
D2				read	print			**switch**	**switch**
D3		read	execute						
D4	read write		read write			**switch**			

Diritto di accesso “switch” per passare da un dominio all’altro da Di a Dj se e solo se **switch ? access[i,j].**

Diritti di accesso “copy” ed “owner”, “control” **per modificare la matrice degli accessi**

oggetti / domini	F1	F2	F3
D1	execute		write*
D2	execute	read*	execute
D3	execute		

Il diritto “copy” consente di copiare il diritto di accesso solo entro la colonna (cioè quell’oggetto) per la quale il diritto è definito

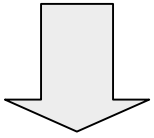
*** ⟶ “copy”**

oggetti / domini	F1	F2	F3
D1	execute		write*
D2	execute	read*	execute
D3	execute	read*	

Varianti del diritto *copy*

- Quando un diritto è copiato da **access[i,j]** ad **access[k,j]**, viene rimosso da **access[i,j]** (***trasferimento del diritto di accesso***)
- ***Propagazione limitata*** del diritto di accesso. Quando il diritto R* è copiato da **access[i,j]** ad **access[k,j],** solo R (e non R*) viene creato (copia limitata)

oggetti / domini	F1	F2	F3
D1	owner execute		write
D2		read* owner	read* owner write*
D3	execute		



oggetti / domini	F1	F2	F3
D1	owner execute		write
D2		owner read* write*	owner read* write*
D3		write	write

Owner: se **access[i,j]** contiene il diritto **owner,** un processo che esegue in **Di** può aggiungere e rimuovere ogni diritto in ogni *entry* nella colonna j

D1 è proprietario di F1

D2 è proprietario di F2 e F3

“*control*”. Serve per modificare il contenuto **in una riga.**

Se access[i,j] contiene “control” : un processo che esegue nel dominio
Di può rimuovere diritti di accesso **dalla riga j**

oggetti / domini	F1	F2	F3	disco	stampante	D1	D2	D3	D4
D1	read		read				switch		
D2					print			switch	Switch control
D3		read	execute						
D4	write		write	~~read~~		switch			

La matrice precedente è stata modificata, in quanto la presenza di control in access [D2,D4] ha consentito di rimuovere il diritto “read” su *disco* da parte di un processo operante in D2.

REALIZZAZIONE DELLA MATRICE DEGLI ACCESSI

- La matrice è vista come un insieme ordinato di triple ***<dominio, oggetto,diritti di accesso>.***

- Se è richiesta una operazione M su un oggetto Oj nel dominio Di, si ricerca nella tavola la tripla:

  <Di,Oj,Rk> con M appartenente a Rk.

  Se esiste, l'operazione può continuare. Diversamente si ha errore.

Problemi realizzativi:

- Dimensione della matrice
- La matrice e` sparsa
- Se un'operazione può essere eseguita su tutti gli oggetti deve essere contenuta in tutti i domini

Realizzazione

➔ Necessita` di rappresentazioni piu` ***compatte*** della matrice:

- rappresentazione "per colonne": lista degli accessi
- rappresentazione "per righe": capability list

oggetti / domini	F1	F2	F3	printer
D1	read		read	
D2				print
D3		read	execute	
D4	read write		read write	

LISTA DEGLI ACCESSI: ACL

Per ogni oggetto viene indicata la coppia ordinata:

<dominio, insieme dei diritti >

limitatamente ai domini con un insieme di diritti non vuoto.

- Quando deve essere eseguita un'operazione M su un oggetto Oj nel dominio Di, si cerca nella lista degli accessi associata a Oj:

  <Di,Rk> con M appartenente a Rk

  Se non esiste, si cerca eventualmente in una *lista di default*; se non esiste si ha condizione di errore.

- Per motivi di efficienza, si può cercare prima nella lista di *default* e successivamente nella lista degli accessi.

Esempio: file system

- Lista dei diritti di accesso associata al file
- Contiene: nome utente (dominio) e diritti di accesso

ACL è stata descritta per *utenti singoli.* Molti sistemi hanno il concetto di *gruppo di utenti.* I gruppi hanno un nome e possono essere inclusi nella ACL.

Siano UID(user identifier) e GID (group identifier) gli identificatori di un processo. L'entry in ACL ha la forma:

UID1, GID1: diritti1
UID2, GID2: diritti2

Concetto di ruolo. Uno stesso utente può appartenere a gruppi diversi e quindi con diritti diversi.

➔Quando accede deve specificare il gruppo di appartenenza (oppure ci sono differenti coppie *login-password* per ogni gruppo)
Obiettivo: tenere separati i diritti di accesso.

L’utente può accedere a certi file, **indipendentemente** dal gruppo cui appartiene. Ad esempio per il file F1:

tina,* : RW

➔l’utente *tina* ha accesso a F1 indipendentemente dal gruppo di appartenenza.

•Possibilità di bloccare selettivamente uno specifico utente:

virgilio,* : (nessun diritto); *,* : RW

➔Tutti possono accedere al file tranne *virgilio* (le entry nella ACL sono esaminate in sequenza).

Revoca dei permessi di accesso ad un file.
Si ottiene modificando la ACL. Ogni file già aperto continuerà, tuttavia, ad avere i diritti di quanto è stato aperto.

CAPABILITY LIST

E’ una **lista di oggetti** assieme con le **operazioni** consentite su di essi, **per ogni dominio**.

- L’oggetto è rappresentato dal suo **indirizzo fisico** (***capability***)
- Per eseguire un’operazione M su un oggetto O, il processo deve fornire come parametro la *capability* (puntatore all’oggetto).
- Per ottenere la *capability* occorre che:
  - Oggetto sia nel dominio in cui opera il processo
  - Diritto di accesso sia tra quelli consentiti

	F1	F2	F3	F4	F5	F6	PR1	PR2
dominio	-	-	R	RWX	RW	-	W	-

Tipo	**Diritti**	**Oggetto**
File	**R--**	**Puntatore a F3**
File	**RWX**	**Puntatore a F4**
File	**RW-**	**Puntatore a F5**
Printer	**-W-**	**Puntatore a stampante PR1**

capability list per il dominio

Le liste di capability devono essere protette da manomissioni degli utenti. Ciò si può ottenere con:

Architettura etichettata, un progetto hardware in cui ogni parola ha un bit extra (*tag*) che dice se la parola contiene o meno una capability.
Il bit tag non e' utilizzato dall'aritmetica, dai confronti e da altre istruzioni normali e può essere modificato solo da programmi che agiscono in modo kernel (S.O.) . Es IBM AS/400.
Si possono utilizzare più bit per consentire all'hardware di distinguere tra vari tipi di dati: intero, carattere, boolean, istruzioni, capabilities etc.

Lo **spazio di indirizzi** associato ad un programma può essere **suddiviso in due parti**. La prima e' accessibile al programma e contiene dati ed istruzioni. La seconda contiene la capability list ed è accessibile solo al S.O.(spazio di memoria segmentato)

CONFRONTO

Un sistema di protezione realizzato esclusivamente con ACL o capability list può presentare alcuni problemi di **efficienza**:

- **ACL**. L'informazione di quali diritti di accesso possieda un dominio (utente, processo) è sparsa nelle varie ACL relative agli oggetti del sistema. Ogni accesso allo stesso oggetto da parte di un processo comporta una ricerca nella lista.
- **Capability list**. La rimozione di un oggetto con diritti di accesso per ogni dominio comporta la ricerca in tutte le capability list. La revoca dei diritti di accesso è complessa.

➔ La soluzione che viene adottata in generale è di usare una **combinazione** dei due metodi.

SOLUZIONE MISTA

- Processo tenta di accedere ad un oggetto per la prima volta:
- Si analizza la ACL; se esiste una entry contenente il nome del processo (dominio) e se tra i diritti di accesso c'è quello richiesto dal processo, viene fornita la *capability* per l'oggetto.
- Ciò consente al processo di accedere all'oggetto più volte senza che sia necessario analizzare la *ACL.*
- Dopo l'ultimo accesso la *capability* è distrutta.

Esempio: S.O. Unix. Apertura di un file

```
fd=open(<nome file>, <diritti di accesso>);
```

➔ Si cerca il file nel direttorio e si verifica se l'accesso è consentito (*ACL*).

In caso affermativo:

1. viene creata una nuova entry nella tabella **dei file aperti del processo, individuata da fd alla quale vengono associati i diritti di accesso** (*capability list*).
2. Viene ritornato al processo ***fd***, cioe` il file descriptor corrispondente al nuovo file aperto.

- Tutte le successive operazioni sul file sono eseguite utilizzando direttamente **fd** (**capability**) e verificando che il diritto di accesso sia tra quelli consentiti.

STATI AUTORIZZATI E NON AUTORIZZATI

- La matrice di protezione **non è statica**: varia quando si creano nuovi oggetti, si distruggono i vecchi, vengono modificati i diritti di accesso.

- Esistono **comandi di protezione** con i quali gli utenti possono modificare la matrice.Il loro utilizzo deve essere vincolato da una ***politica di gestione.***

- Ad ogni istante la matrice di protezione determina cosa **può** fare un processo in ogni dominio, non cosa **è autorizzato** a fare. L'autorizzazione dipende dalla **politica di gestione**.

Esempio. Politica di gestione adottata:

	Compilatore	Mailbox	Documento
Enrico	RE		
Paolo	RE	RE	
Roberto	RE		RE

Roberto riesce a modificare la matrice:

	Compilatore	Mailbox	Documento
Enrico	RE		
Paolo	RE	RE	
Roberto	RE	R	RE

Si ottiene uno stato non autorizzato dalla politica di protezione. Il meccanismo di protezione consente la lettura di mailbox da parte di Roberto.

SICUREZZA MULTILIVELLO

- La maggior parte dei sistemi operativi permette a singoli utenti di determinare chi possa leggere e scrivere i loro file ed i loro oggetti (***controllo discrezionale degli accessi***).
- In alcuni ambienti è richiesta una **sicurezza più stretta** (ambiente militare,ospedali, aziende..): Vengono stabilite regole su chi può vedere cosa e non possono essere modificate senza aver ottenuto permessi speciali dal ***capo*** dell'organizzazione (***controllo degli accessi obbligatorio***)

SICUREZZA MULTILIVELLO

Obiettivo: assicurarsi che le politiche di sicurezza stabilite siano rispettate dal sistema.

Modello Bell-La Padula [1973]

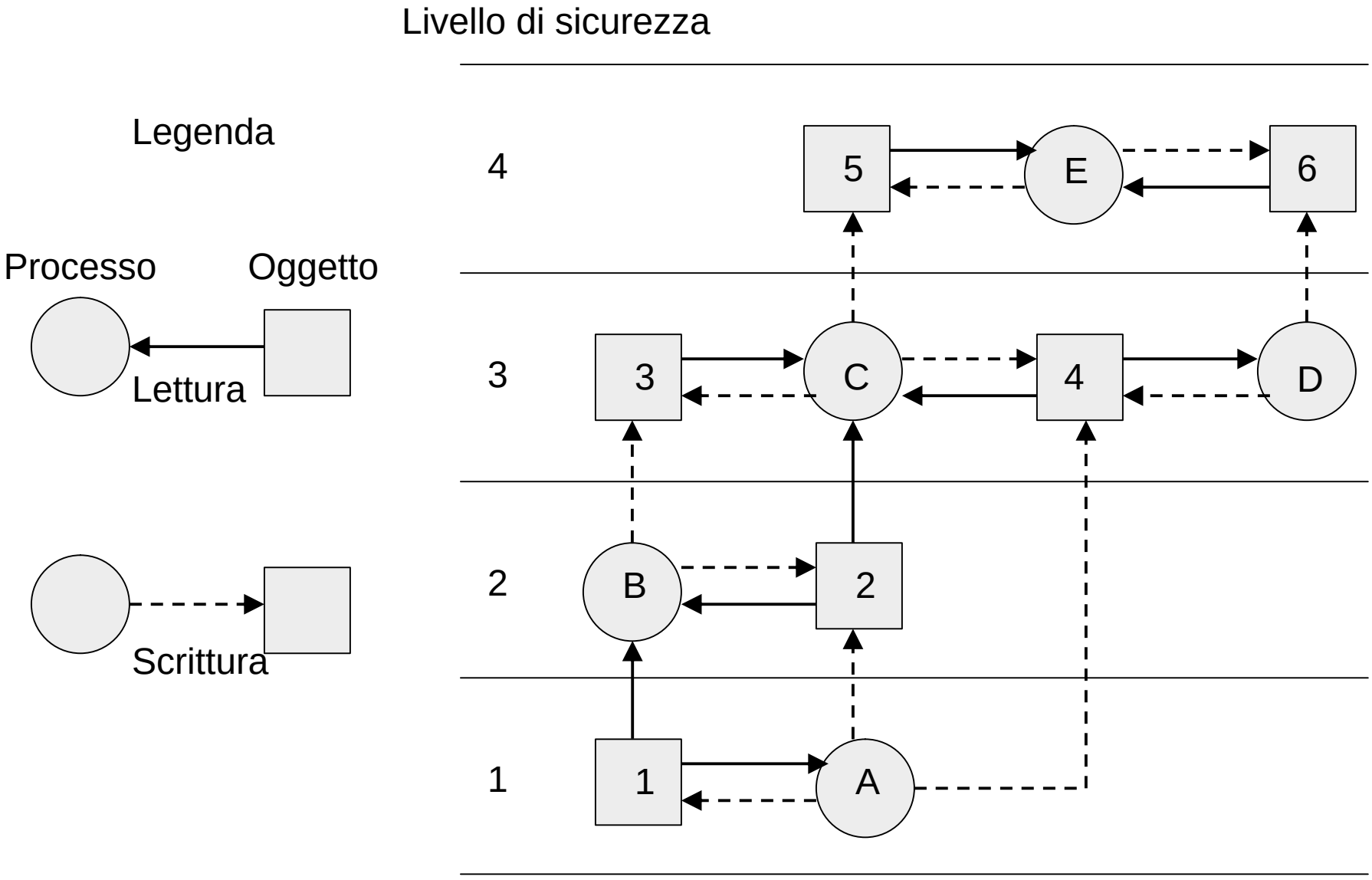
- Progettato per gestire la sicurezza in ambiente militare.
- Livelli di sicurezza dei documenti:
  - non classificato
  - confidenziale
  - segreto
  - top secret
- Le persone sono assegnate ai livelli a seconda dei documenti che è loro consentito esaminare

Regole su come le informazioni possono circolare:

Proprietà di semplice sicurezza: un processo in esecuzione al livello di sicurezza k può **leggere** solo oggetti al suo livello o a livelli inferiori.

Proprietà*: un processo in esecuzione al livello di sicurezza k può **scrivere** solamente oggetti al suo livello o a quelli superiori.

➔I processi possono **leggere verso il basso** e **scrivere verso l'alto**, ma non il contrario.
➔Nessuna informazione puo` essere trasferita da un livello piu` alto a uno piu` basso.


Livello di sicurezza
Legenda
Processo
Oggetto
Lettura
Scrittura
4
3
2
1
5
E
6
3
C
4
D
B
2
1
A

Il modello Bell-La Padula è stato concepito per mantenere i segreti, **non per garantire l'integrità dei dati**. E' possibile infatti sovrascrivere l'informazione appartenente ad un livello superiore.

Modello Biba [1977]:

> **Proprietà di semplice sicurezza:** un processo in esecuzione al livello di sicurezza k può scrivere solamente oggetti al suo livello o a quelli inferiori (nessuna scrittura verso l'alto).
> **Proprietà di integrità*:** un processo in esecuzione al livello k può leggere solo oggetti al suo livello o a quelli superiori (nessuna lettura verso il basso)

➔I due modelli sono in conflitto tra loro e non si possono realizzare contemporaneamente.

Reference Monitor

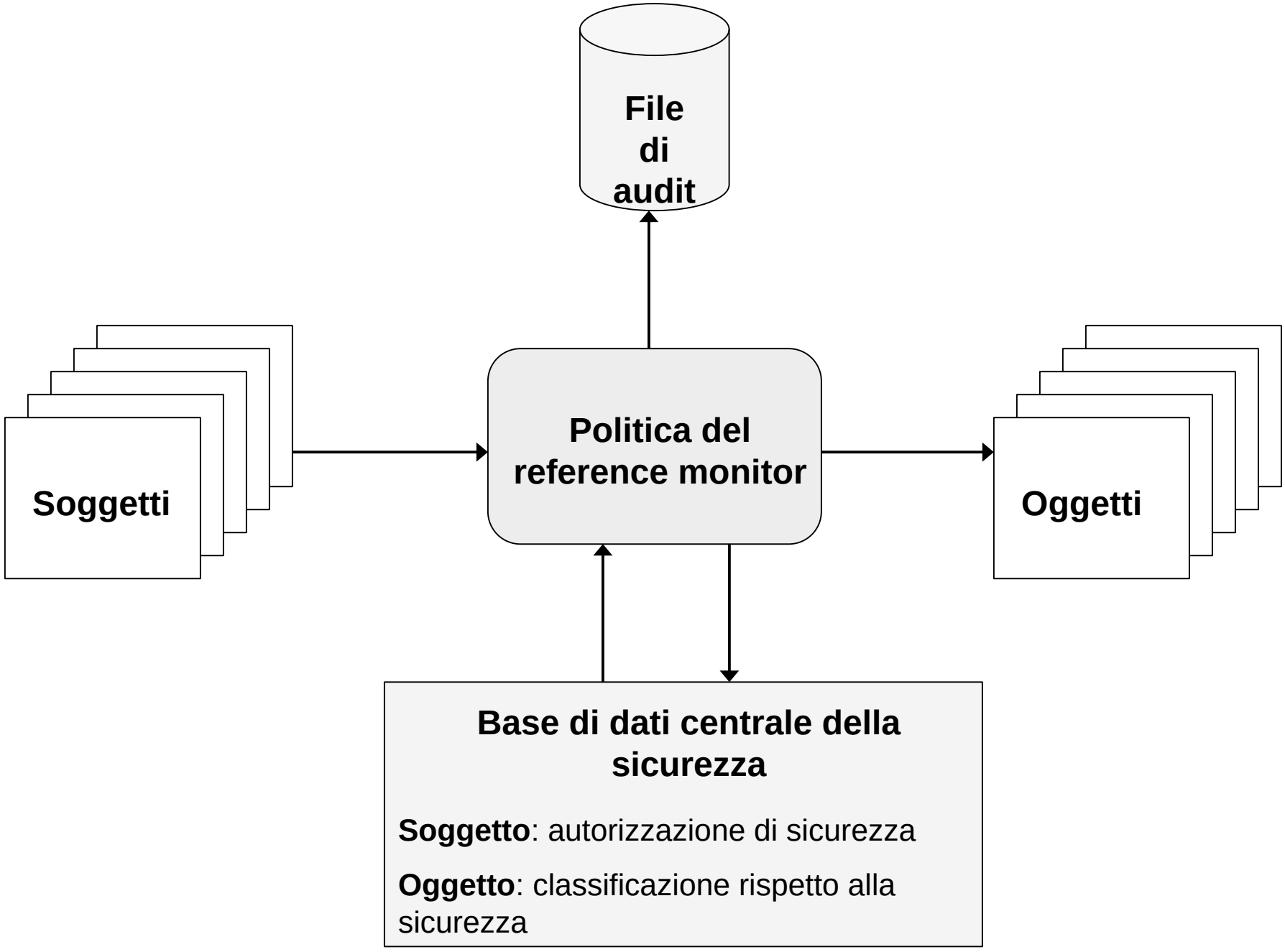
- E’ un *elemento di controllo* realizzato *dall’hardware e dal S.O*. che regola l’accesso dei soggetti agli oggetti sulla base di parametri di sicurezza del soggetto e dell’oggetto.
- Ha accesso a file (**base di dati del nucleo di sicurezza** - *security kernel data-base*) che elencano:
  - Privilegi di sicurezza (***autorizzazioni** di sicurezza*) di ogni soggetto.
  - Attributi di protezione (***classificazione rispetto alla sicurezza***) di ciascun oggetto.

Il monitor dei riferimenti *impone le regole di sicurezza* (***no read-up***, ***no-write down***) ed ha le seguenti proprietà:

- Mediazione completa: le regole di sicurezza vengono applicate *ad ogni accesso* e non solo, ad esempio, quando viene aperto un file.

- Isolamento: il monitor dei riferimenti e la base di dati *sono protetti* rispetto a modifiche non autorizzate.

- Verificabilità: la correttezza del monitor dei riferimenti deve esser provata, cioè deve esser possibile dimostrare *formalmente* che il monitor impone le regole di sicurezza ed fornisce mediazione completa ed isolamento


File
di
audit
Soggetti
Politica del
reference monitor
Oggetti
Base di dati centrale della sicurezza
Soggetto: autorizzazione di sicurezza
Oggetto: classificazione rispetto alla sicurezza

• Il requisito di ***mediazione completa*** impone, per motivi di efficienza, che la soluzione debba essere almeno parzialmente hardware.

• Il requisito *dell'**isolamento*** impone che non sia possibile per chi porta l'attacco, modificare la logica del reference monitor o il contenuto della base di dati centrale della sicurezza.

• Il requisito della ***dimostrazione formale*** è difficile da soddisfare per un sistema general-purpose.

Audit file

Vengono mantenuti in questo file gli *eventi importanti per la sicurezza*, come le violazioni alla sicurezza che sono state scoperte e le modifiche autorizzate alla base di dati del nucleo di sicurezza.

Esempio: Difesa dai cavalli di Troia

• Nell’esempio viene usato un cavallo di Troia per aggirare un meccanismo di controllo basato sulle liste di controllo degli accessi (ACL).

• Un utente, Paolo, ha creato un file, contenente la stringa di caratteri riservati “CPE1704TKS”, con i permessi di *lettura/scrittura* solo per i processi che appartengono a lui.

• Un utente ostile, Piero, ottenuto l’accesso al sistema installa sia il *cavallo di Troia* sia un file privato che verrà utilizzato come “tasca posteriore”.

• Piero ha permessi di lettura e scrittura per il suo file e dà a Paolo il permesso di scrittura.

• Piero induce Paolo ad attivare il cavallo di Troia (spacciandolo come un programma di utilità).

• Il programma, eseguito da Paolo copia la stringa dei caratteri riservati nel file “tasca posteriore” di Piero: sia l’operazione di lettura che quella di scrittura soddisfano i vincoli imposti da ACL.

➔ **Utilizzo di un S.O. sicuro**. Vengono fissati due livelli di sicurezza, *riservato e pubblico*. Ai processi ed al file dati di Paolo viene assegnato il livello di sicurezza “riservato”. A quelli di Piero il livello “pubblico”.

• Quando Paolo attiva il cavallo di Troia, questo acquisisce il livello di sicurezza di Paolo e può vedere la stringa di caratteri riservata. Quando il programma tenta di memorizzarla in un file pubblico (file della tasca posteriore) la proprietà * è violata ed il tentativo non viene consentito dal *reference monitor* (anche se la ACL lo permetterebbe).

• **La politica di sicurezza *ha la precedenza sul meccanismo delle ACL***

Classificazione della sicurezza dei sistemi di calcolo

Orange Book. Documento pubblicato dal Dipartimento della Difesa americano (D.O.D). Sono specificate quattro categorie di sicurezza:A,B,C,D(in ordine decrescente).

- **Categoria D.** Non ha livelli di sicurezza. Esempio MS-DOS, Windows 3.1.

- **Categoria C.** Suddivisa in C1 e C2:
  - **C1.** La TCB consente:
    - Autenticazione degli utenti (password). I dati di autenticazione sono protetti rendendoli inaccessibili agli utenti non autorizzati.
    - Protezione dei dati e programmi propri di ogni utente.
    - Controllo degli accessi a oggetti comuni per gruppi di utenti definiti

Esempio: Unix

• *C2*. La TCB consente, oltre a quanto definito per la C1, il controllo degli accessi su una *base individuale*. Esempio UNIX, Windows NT e 2000.

• **Categoria B**. Suddivisa in B1, B2 e B3

B1. La TCB consente, oltre a quanto definito in C2, l'introduzione dei livelli di sicurezza (modello Bell-La Padula). Almeno due livelli.

B2. La TCB estende l'uso di etichette di riservatezza ad ogni risorsa del sistema, compresi i canali di comunicazione.

B3. La TCB consente la creazione di liste di controllo degli accessi in cui sono identificati utenti o gruppi *cui non è consentito* l'accesso ad un oggetto specificato.

• **Categoria A**. Suddivisa in A1 e classi superiori

A1. E' equivalente a B3, ma con il vincolo di essere progettato e realizzato utilizzando metodi formali di definizione e verifica.

Un sistema appartiene ad una classe superiore ad A1 se è stato progettato e realizzato in un impianto di produzione affidabile da persona affidabile

SICUREZZA IN WINDOWS 2000

Adotta uno schema di controllo degli accessi uniforme che si applica a processi, threads, files, semafori, finestre ed altri oggetti.

Il controllo degli accessi utilizza due entità:

- **access token** associato ad ogni processo
- **security descriptor** associato ad ogni oggetto su cui è possibile l'accesso di più processi

Autenticazione dell'utente tramite nome/password. Viene creato un processo per l'utente cui viene assegnato un access token.

Access token contiene un security ID (SID) che è l'identificatore per l'utente per scopi di sicurezza. Tutti i processi generati dal processo iniziale hanno lo stesso access token.

Access token contiene le seguenti informazioni:
Security ID: identifica un utente in maniera univoca.

Group SID: Una lista dei gruppi ai quali l'utente appartiene ai fini del controllo degli accessi. Un gruppo è un insieme di ID di utente. Ogni gruppo ha un unico SID.

Privilegi: Una lista di servizi di sistema "security sensitive" che l'utente può chiamare. Esempio creare token.

Default owner: se l'utente genera un nuovo oggetto può specificare se il proprietario dell'oggetto è l'utente stesso o un gruppo cui l'utente appartiene.

Default ACL: lista iniziale di protezione associata agli oggetti che l'utente crea. La lista può essere successivamente modificata

Security descriptor contiene le seguenti informazioni:

Flags: definiscono il tipo ed i contenuti di un descrittore. Esempio: se sono contenute o no *SACL e DACL.*

*Owner:*il proprietario dell'oggetto può essere un individuo o un gruppo. Il proprietario può cambiare i contenuti della *DACL.*

Discretionary Access Control List *(DACL):* determina quali utenti e con quali operazioni possono accedere all'oggetto. E' costituita da una lista di *access control entries* (*ACEs*).

*System Access Control List(SACL):*specifica quali tipi di operazioni sull'oggetto generano *audit messages.*

• *Access Control Entry:* contiene un SID individuale o di gruppo e una *maschera di accesso* che definisce i diritti che sono garantiti al SID.

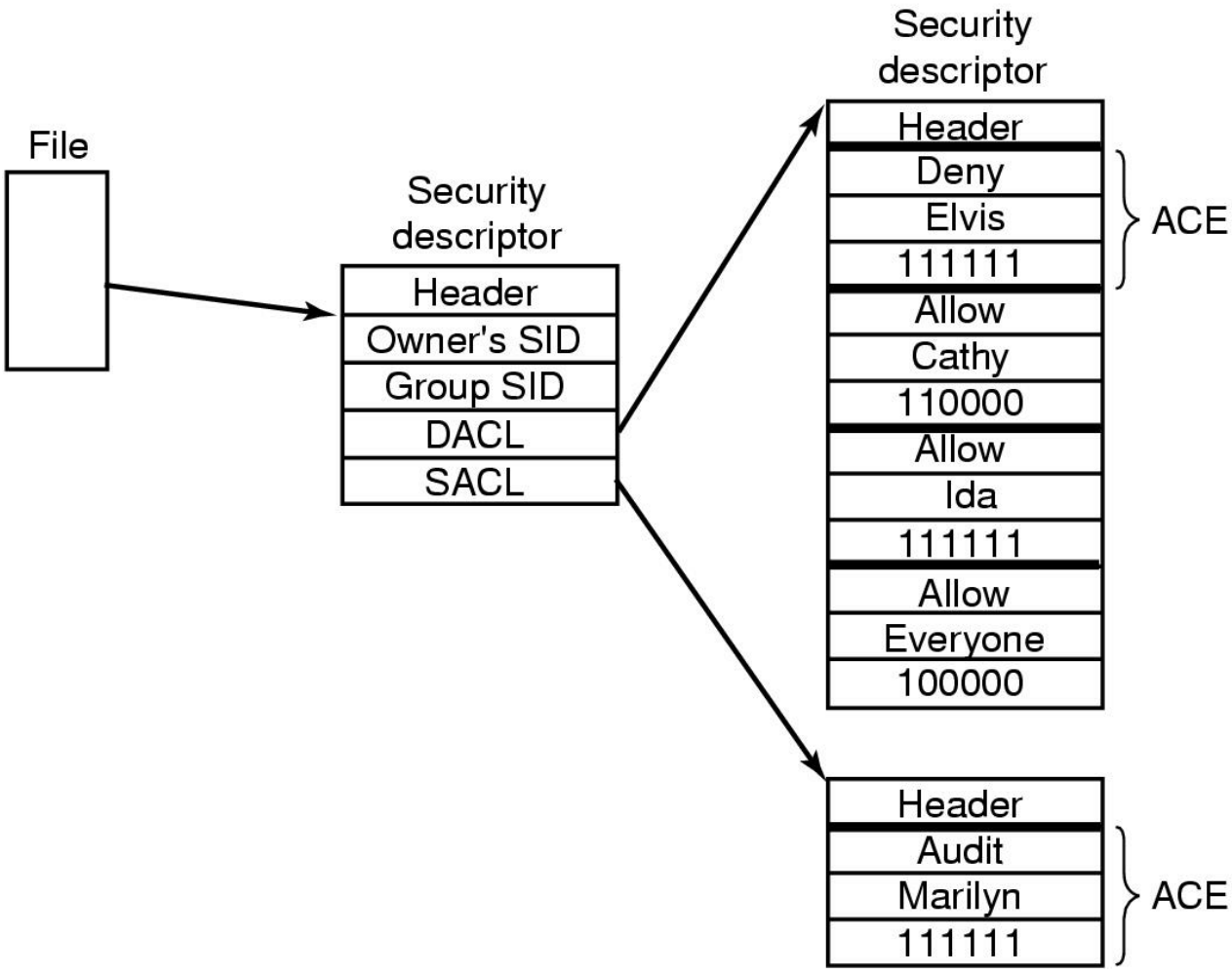
• Quando un processo tenta di accedere ad un oggetto, *l'object manager* legge il *SID* e *group SID* da *access token* e analizza la lista DACL. Se la lista contiene il SID il processo ha i diritti di accesso che compaiono nella maschera di accesso..

• La maschera contiene:

quattro bit per indicare i diritti di accesso di *lettura, scrittura, esecuzione e tutti i diritti*.

Altri bit per indicare ad esempio se un programma può modificare il proprietario dell'oggetto, cancellare l'oggetto etc..

Esempio: security descriptor



STENOGRAFIA

• Scrittura nascosta. Inserimento di segni di riconoscimento nascosti (*watermark*, marchi sull'acqua) nelle immagini utilizzate in pagine web, nella musica, nei film etc.

• Utilizzo delle immagini. Ogni pixel è composto da tre numeri di 8 bit, uno per ogni intensità di rosso, verde, blu (RGB).Il colore del pixel è dato dalla sovrapposizione lineare dei tre colori.

• Il metodo di codifica utilizza i bit meno significativi dei tre numeri come *canale nascosto*. Impossibilità di distinguere colori a 7 bit da colori a 8 bit.

• Esempio: immagine da 1024x768 pixel. Possibilità di immagazzinare 1024x768 pixel, ossia 294912 byte. Il testo da trasmettere viene compresso (se necessario), crittato ed inserito nei bit meno significativi di ogni valore di colore.