I seguenti appunti non sono appunti ufficiali del corso di Sistemi Operativi e non sono stati revisionati da alcun docente. Nelle pagine seguenti troverete degli appunti riepilogativi degli argomenti trattati a lezione dal docente sugli aspetti di protezione e sicurezza, sul libro di testo consigliato e nella dispensa https://github.com/Guray00/IngegneriaInformatica/blob/master/TERZO%20ANNO/I%20SEMESTRE/Sistemi%20operativi/Dispense%20degli%20studenti/Dispensa%20lezioni%20Avvenuti%2C%20A.A.21-22.pdf. Tutte le immagini presenti sono soggette a Copyright.
Buona lettura!

D. B.

Protezione e sicurezza

Protezione: insieme delle attività che si occupano di garantire il ***controllo degli accessi*** alle risorse fisiche e logiche da parte degli utenti all'interno di un sistema.

Sicurezza: insieme di attività che si occupano di garantire ***l'autenticazione*** degli utenti impedendo accessi non autorizzati.

Modelli di protezione

Un modello di protezione definisce:

- **Soggetti**, sono la parte attiva di un sistema (processi o thread) che agiscono per conto degli utenti potendo accedere a determinati oggetti
- **Oggetti**, sono la parte passiva del sistema, possono essere le risorse fisiche o quelle logiche, ma anche altri soggetti in quanto, ad esempio, un processo può controllarne un altro.

Un soggetto può essere considerato una coppia (processo, dominio):
Dominio è l'ambiente di protezione nel quale il processo sta eseguendo: cioè **insieme dei diritti di accesso posseduti dal processo.**
Un dominio è unico per un soggetto, ma un processo può cambiare soggetto durante la sua esecuzione, cambiando quindi dominio.
Dominio: è definito come un insieme di coppie (oggetto, insieme dei diritti di accesso).

- I domini possono esse disgiunti o avere intersezioni comuni
- In ogni istante un processo è eseguito con qualche dominio
- L'associazione tra processo e dominio può essere:
  - Statica: in contraddizione con principio del minimo privilegio
  - Dinamica, un processo può cambiare dominio durante la sua esecuzione

Il caso più semplice di dominio di protezione è la modalità di funzionamento del processo user-sytem mode.

Politiche di protezione

Definiscono le regole con le quali i soggetti possono accedere agli oggetti

- **Directory Access Control (DAC),** la concessione e la revoca dei diritti di accessi agli oggetti è a discrezione dei singoli utenti che controllano gli oggetti stessi. Ad esempio questa politica è presente in UNIX.
- **Mandatory Access Control (MAC),** i soggetti possono accedere agli oggetti in base alla classe di sensibilità. È una politica molto restrittiva usata specialmente in ambito militare o governativo.
- **Role-Based Access Control (RBAC),** ogni soggetto appartiene ad un ruolo e i diritti di accesso sono definiti sui ruoli. Questo permette ad un utente di avere più ruoli e agli amministratori di cambiare i diritti di accesso a più utenti modificando quelli di un ruolo.

Tutte le politiche di protezione si basano sul ***principio del minimo privilegio***: ad un soggetto sono garantiti i diritti di accesso solo ad oggetti strettamente necessari al completamento del compito che sta svolgendo.

Meccanismi di protezione

Sono gli strumenti messi a disposizione dal sistema operativo per imporre una determinata politica.
Le politiche specificano il cosa fare, i meccanismi come va fatto.

Il modello a matrice degli accessi

Questo modello permette di:

- Rappresentare lo stato di protezione
- Garantire il rispetto dei vincoli di accesso
- Modifica controllare dello stato di protezione

La matrice degli accessi è una matrice nella quale le **righe sono i soggetti** e le **colonne gli oggetti** (tra gli oggetti sono presenti anche i soggetti stesso).
Ogni accesso comporta i seguenti passaggi:

1. Un soggetto S tenta di eseguire un'azione *a* sull'oggetto X
2. Viene generata la tripla (S,*a*,X) e passata al monitor dell'oggetto
3. Il monitor interroga la matrice di protezione: se *a* appartiene ad A[S,X] l'accesso è valido.

	X_1	X_2	X_3	S_1	S_2	S_3
S_1	read*	read	execute		terminate	receive
S_2		owner write		control receive		terminate
S_3	write execute		read	send	send receive	

Modifica dello stato di protezione

Graham e Denning hanno definito un *insieme di comand*i che permette la modifica controllata della matrice degli accessi. Questi comandi sono dei seguenti tipi:

- aggiunta e rimozione di diritti di accesso per oggetti/soggetti;
- propagazione dei diritti di accesso.

Graham e Denning hanno dimostrato che con i comandi precedenti si ha:

- propagazione limitata e controllata (**confinement)**
- è evitata la modifica indiscriminata dei diritti di accesso (sharing parameters)
- non si hanno i cosiddetti cavalli di troia (trojan horse), cioè un uso non corretto dei diritti di accesso.

Propagazione dei diritti di accesso

Un soggetto S_i può propagare un diritto *a* per l'oggetto X al soggetto S_j se e solo se il soggetto S_i possiede quel diritto ed il diritto possiede il copy flag (indicato con *).
La propagazione **può essere "limitata di un diritto"** (cioè chi riceve, non può a sua volta propagare il diritto stesso) oppure può ricevere il diritto *a** potendolo così propagare a sua volta.
La propagazione **può essere un trasferimento del diritto**, cioè chi propaga il diritto lo perde.

Assegnazione e Rimozione di un diritto di accesso

Un soggetto S_i può assegnare un diritto *a* per l'oggetto X al soggetto S_j se e solo owner appartiene ad $A[S_i,X]$.

Un soggetto S_i può revocare un diritto *a* per l'oggetto X al soggetto S_j se:

- solo **owner** appartiene ad $A[S_i,X]$
- oppure, **control** appartiene ad $A[S_i,S_j]$

Realizzazione della matrice degli accessi

Realizzare una matrice degli accessi presenta diversi svantaggi:

- **Dimensione della matrice** (numero righe e numero colonne dinamici)
  La dimensione della matrice può essere considerevole.
- **Matrice sparsa**
  Molti elementi della matrice sono vuoti.
- **Ridondanze**
  Se un'operazione può essere eseguita su tutti gli oggetti, allora l'operazione deve essere contenuta in tutti i domini.

Access Control List

Divide la matrice degli accessi per colonne: **ad ogni oggetto è associata una lista** che contiene tutti i soggetti dai quali è possibile accedere all'oggetto e per ogni soggetto sono presenti i diritti.
Quindi la lista è formata da elementi: <soggetto, diritti>.
Quando un soggetto ***S_i tenta di esercitare un diritto M su un oggetto O_j, il meccanismo di protezione verifica nella ACL associata ad Oj se è presente una coppia <S_i, insieme di diritti che comprende M>.***
Poiché scorrere tutte le volte la lista non è efficiente, è presente ***una lista di default*** (che va scorsa per prima) che ***contiene i diritti di accesso che sono applicabili a tutti gli oggetti.***

Può essere però comodo assegnare diritti diversi ad uno stesso soggetto in base al gruppo di appartenenza in quel momento, quindi in base al ruolo, un elemento presente nella ACL presenta la forma:

UID, GID: < Insieme di diritti >

dove **< UID, GID> è l'identificativo di un soggetto.**
Si osservi che UID1, GID1: < Insieme di diritti 1 >≠ UID1, GID2: < Insieme di diritti 2 >.
Questo tipo di rappresentazione apre la porta a una serie di possibilità:

- Dire che un utente può accedere a certi oggetti indipendentemente dal gruppo a cui appartiene
  UID, *: < Insieme di diritti >
- Bloccare selettivamente uno specifico utente
  UID, *: < $Insieme\ vuoto$ >
- Definire diritti validi per tutti
  *, *: < Insieme di diritti >

Vantaggi ACL: la revoca dei diritti di accesso per un oggetto è semplice
Svantaggi: inefficienza nel determinare i diritti di accesso di un soggetto per un particolare oggetto e nel determinare tutti i diritti di accesso di un soggetto.

Capability List

Divide la matrice degli accessi per righe: **ad ogni soggetto è associata una lista** che contiene tutti gli oggetti accessibili dal soggetto e i relativi diritti di accesso.

Quando un soggetto ***S_i tenta di esercitare un diritto M su un oggetto O_j, il meccanismo di protezione verifica nella CL associata a S_i se è presente una capability relativa all'oggetto O_i che comprende il diritto M.***
Le capabilities possono essere passate da un processo all'altro (uguali o diminuendone i privilegi).
La lista delle capabilities è associata ad un oggetto, ma non è direttamente accessibile da esso: infatti la CL è un oggetto protetto mantenuto dal SO al quale i soggetti possono accedere indirettamente.

Vantaggi CL: inefficienza nel determinare i diritti di accesso di un soggetto per un particolare oggetto e nel determinare tutti i diritti di accesso di un soggetto.
Svantaggi: inefficiente la revoca dei diritti di accesso per un oggetto.

UNIX

Nei sistemi UNIX la matrice degli accessi è implementata come un ACL attraverso i **bit di protezione**.
Le **ACL sono così distribuite nell' iNode!**
In realtà UNIX, pur basandosi sulle ACL, **implementa una soluzione ibrida che comprende sia ACL che CL.**

- Quando ***la open verifica che l'operazione può andare a buon fine*** si usa la ACL, cioè si usano ***i nove bit di protezione nel caso di UNIX***. Il processo è eseguito in nome di un particolare soggetto (UID, GID), le informazioni indicate nel secondo parametro di open vengono usati per verificare la validità nell'accesso.
- Il file descriptor restituito dalla open è una nuova entry posta nella tabella dei file aperti di processo. A questo punto possiamo eseguire operazioni su file, per esempio lettura e scrittura: anche in questo caso dobbiamo fare i controlli. **Quando un soggetto chiede di eseguire una read o una write indicando un particolare file descriptor il sistema controllerà che a fd corrisponda una entry che ammette tale operazione. Questo controllo viene fatto solo sulla tabella dei file aperti di processo, non sulla iNode:** segue che abbiamo proprio *qua l'implementazione della Capability List.*
- **Le ACL sono persistenti** e relativamente statiche. **Le CL sono dinamiche**: non esistono CL senza processi, al momento della creazione del processo si inizializza la CL con le capability relative allo STDIN, STDOUT, STDERR.

Sicurezza Multilivello

Obiettivo: garantire in *un sistema in cui esistono oggetti con livelli di segretezza* diversi *l'estensione dei livelli di segretezza ai soggetti*, e *definire delle regole che autorizzino i soggetti negli accessi* a seconda del livello di sicurezza del soggetto stesso e dell'oggetto coinvolto.

Modello Bell-La Padula

In questo modello associamo livelli di segretezza a oggetti e soggetti.
Supponiamo di averne quattro:

- Unclassified
- Confidential
- Secret
- Top secret

Il livello di segretezza associato al soggetto indica a quali documenti il soggetto ha diritto di accesso.
Possiamo riassumere il modello di Bell – LaPadula dicendo:

Lettura verso il basso, scrittura verso l'alto

Immaginiamo una funzione SC, che dato come parametro un utente o un oggetto restituisce il suo livello di sicurezza.
Proprietà di semplice sicurezza:
Un soggetto con un livello di sicurezza k può **leggere** solo oggetti al suo livello o a livelli inferiori, cioè:

$$SC(S) \geq SC(O)$$

Proprietà STAR (*):
Un soggetto in esecuzione al livello di sicurezza k può **scrivere** solamente oggetti al suo livello o a quelli superiori, cioè:

$$SC(S) \leq SC(O)$$

Se l'utente X è a livello top secret non può modificare documenti a livello non classificato (può solo leggerli).
Bell-La Padula non risolve il problema dell'integrità: infatti un utente può leggere informazioni al livello i-1, modificarle a propagarle a livello i+1.
Il modello garantisce però la sicurezza: la proprietà STAR * impedisce che un utente legga l'informazione a livello alto per poi scriverla a livello più basso: se così non fosse sarebbe possibile rendere disponibili informazioni con livello di segretezza elevato ad un livello più basso.

Modello BIBA

"Duale" rispetto a Bell – LaPadula: garantisce integrità ma non sicurezza, possiamo riassumere il modello dicendo:

Lettura verso l'alto, scrittura verso il basso

Immaginiamo una funzione SC, che dato come parametro un utente o un oggetto restituisce il suo livello di sicurezza.

Proprietà di semplice integrità:

Un soggetto con un livello di sicurezza k può **leggere** solo oggetti al suo livello o a livelli superiore, cioè:

$$SC(S) \leq SC(O)$$

Proprietà di integrità STAR (*):

Un soggetto in esecuzione al livello di sicurezza k può **scrivere** solamente oggetti al suo livello o a quelli inferiori, cioè:

$$SC(S) \geq SC(O)$$

Se l'utente X è a livello top secret non può modificare documenti a livello non classificato (può solo leggerli).

Bell-La Padula non risolve il problema dell'integrità: infatti un utente può leggere informazioni al livello i-1, modificarle a propagarle a livello i+1.

Il modello garantisce però la sicurezza: la proprietà STAR * impedisce che un utente legga l'informazione a livello alto per poi scriverla a livello più basso: se così non fosse sarebbe possibile rendere disponibili informazioni con livello di segretezza elevato ad un livello più basso.

Le regole del modello Biba sono in conflitto con le regole di Bell-La Padula: non possono essere attuate in contemporanea!

Difesa dai cavalli di Troia con la sovrapposizione tra matrice degli accessi e modello Bell-LaPadula

Supponiamo di avere la seguente matrice degli accessi:

	O1 (Segreto)	O2	O3 (Pubblico)
S1	Read, Write	Execute	Write
S2		Execute, Owner	Read, Write, Owner

- Il soggetto S1 ha diritto esclusivo in lettura e scrittura sull'oggetto O1 che contiene informazioni riservate
- Il soggetto S2 installa un programma (O2), un trojan horse, e un file privato (O3); la presenza del diritto di **owner** permette di trasferire i diritti di **execute** e di **write** al soggetto S1.
- Supponiamo che O1 esegua il programma O2, il quale chiede l'accesso in lettura ad O1: siccome il soggetto è S1, la richiesta viene accettata dal meccanismo di controllo degli accessi.
  O2 poi chiederà di scrivere su O3 copiano ciò che ha letto da O1.
  S2 può leggere così le informazioni contenute in O1!

Se però vengono introdotti due livelli, uno pubblico e l'altro riservato, le cose cambiano:

S1 e O1 hanno livello riservato, mentre S2, O2, O3 hanno livello pubblico:

- O1 esegue il programma O2, il quale chiede l'accesso in lettura ad O1: siccome il soggetto è S1, ha diritto di lettura **e** il livello **è minore o uguale,** allora la richiesta viene accettata dal meccanismo di controllo degli accessi.
  O2 poi chiederà di scrivere su O3 per copiare ciò che ha letto da O1: anche se S1 ha diritto di scrittura, il suo livello di sicurezza è maggiore di quello dell'oggetto O3, per cui la richiesta viene negata.

Sovrapponendo (ovvero facendo un AND) la tabella degli accessi e del modello di Bell LaPadula si possono evitare i cavalli di Troia.