12 - Access Control

Dopo che un utente è stato autenticato il secondo step è l'autorizzazione o access control
Il compito dell'access control è di prevenire che utenti non autorizzati accedano alle risorse
Insieme di policy e meccanismi che servono a decidere se un soggetto particolare è abilitato ad eseguire certe operazioni su certi oggetti.
Elementi basilari dell'access control sono:

- soggetto: entità con la possibilità di accedere ad oggetti
- oggetti: risorse che devono essere controllate
- access right: descrive in che modo un soggetto accede ad un oggetto

Tipi di policy dell'access control

- Discretionary access control (DAC): accesso basato su identità del soggetto e regole di accesso che descrivono quale soggetto è abilitato a usare un oggetto. Discretionary perchè i soggetti decidono di dare o negare l'accesso ad altri soggetti.
- Mandatory access control (MAC): accesso basato sulla comparazione di security labels (che descrivono quando critico è l'oggetto) con le security clearance del soggetto. Mandatory perchè le security label e clearances sono impostate dal sistema e non possono essere modificate dai oggetti.
- Role-based access control (RBAC): accesso basato sul ruolo che i soggetti posseggono nel sistema e su regole che dicono quale accesso il loro ruolo garantisce.

Principi fondamentali per security policies

- design aperto
- economia di meccanismi
- fail-safe defaults
  - di default i soggetti non hanno accesso privilegiato su nessun oggetto
- complete mediation (reference monitor)
  - non è possibile accedere direttamente a un oggetto, tutti gli accessi vanno controllati
- least privilege
  - i soggetti hanno il livello minimo di privilegi che da la possibilità di accedere alla risorsa

least privilege
Ogni soggetto dovrebbe operare usando un set di privilegi minimo necessari per eseguire la task

- imitazione danni risultanti da errori accidentali
- limitazione numero di programmi con privilegi
- aiuto in debugging
- incrementa la sicurezza
- permette l'isolazione di sottosistemi critici

Viene inforzato tramite un reference monitor che implementa complete mediation. Ogni accesso a ogni oggetto è controllato.

Notazione

- Sia S set dei soggetti
- Sia O set si oggetti
  - nota: gli oggetti possono agire come soggetti
- Sia α set di access right che i soggetti hanno su gli oggetti

Protection domains

Un protection domain è un set di oggetti che di access rights per ognuno.
Formalmente è un tupla **<object, set_of_access_rights>**
I soggetti sono associati con un protection domain in cui operano
L'associazione tra soggetto e protection domain può essere statica o dinamica.

Access control Matrix model for DAC

Un modello per Discretionary Access Control (DAC)
Access control matrix

- è una matrice M con il dominio come righe e gli oggetti come colonne
- ogni entry $M(i, j)$ contengono il set di access rights α che il dominio D_i permette su O_j
  Quando un nuovo oggetto viene creato:
- si aggiunge una nuova colonna alla matrice
- il contenuto della colonna è deciso dal creatore dell'oggetto

		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

Es:

- User A in domain D_2 editing $File_2$, user B in D_3 editing $File_3$
- Users A and B turn on "spelling corrector" function based on $File_4$ which is a ***dictionary***
- The ***dictionary*** is proprietary and should not be copied

	object / domain	$File_1$	$File_2$	$File_3$	$File_4$
	D_1	read			
A	D_2		read write	read	read
B	D_3			read write	read

But now A and B can make copies of the dictionary

- Introduce a new domain D_4 such that the dictionary can only be read in that domain and add new access right "switch"

	object / domain	$File_1$	$File_2$	$File_3$	$File_4$	D_4
	D_1	read				
A	D_2		read write	read		switch
B	D_3			read write		switch
	D_4				read	

- But now users A and B cannot access the files they are editing ($File_2$ and $File_3$)
- "Switch" not only changes domains but also copies the access rights from the source domain to the destination domain
- Since there may be multiple users that switch to the same domain, they are kept logically distinct by creating multiple instances of the domain
- This mechanism effectively implements the "principle of least privilege"

	object / domain	$File_1$	$File_2$	$File_3$	$File_4$	D_4
	D_1	read				
A	D_2		read write	read		switch
B	D_3			read write		switch
A	D_4		read write	read	read	
B	D_4			read write	read	

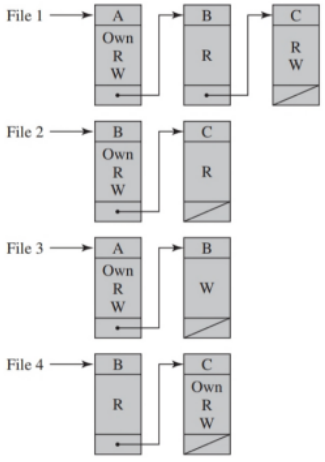
Implementazione:
come tabella globale

- si salva a matrice come un array bidimensionale (tabella) con le entries che sono <set_of_access_rights>
  Vantaggi: semplice da implementare
  Svantaggi:
- la tabella può diventare enorme
- difficile da mantenere in un sistema dinamico dove i domini e gli oggetti sono aggiunti e cancellati e gli access rights cambiano spesso.

Access control List

Access control list (ACL) è una tabella salvata "per colonna" in cui ogni oggetto è associato ad una lista di access rights per ogni dominio<domain, set_of_access_rights>
Ottimizzazione per ridurre la lunghezza della lista

- includere solo i domini che hanno access rights diversi da quello di default
- raggruppare i domini in (piccoli) gruppi e definire access rights solo per loro
  ACL agisce come la "guest list" per una festa.

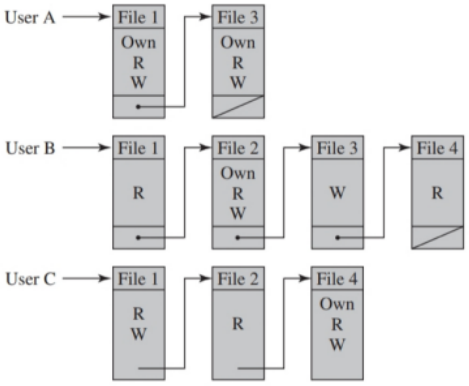


Capability

La tabella è salvata "per riga"
Ogni dominio è associato a una lista di access "rights" <object, access_rights_for_object>
Chi mantiene le capabilities?

- processi che ricordano loro di esercitare il diritto di accesso sull'oggetto

- agiscono come chiavi per aprire le serrature



Per far funzionare il meccanismo delle capability bisogna garantire che:

- i processi non sono in grado di forgere false capability
- l'oggetto (reference monitor) sia in grado di riconoscere se una capability è falsa o autentica
- i processi possono avere il permesso o no di copiare o trasferire le proprie capability

Implementazione

Possono essere implementati usando la public key cryptography
I processi ricevono le capabilities in formato di triplette: <object, access_rights_for_object, unique_code> dopo essere stati firmati con la chiave privata dell'oggetto
I processi non possono modificare le capabilities visto che non possono firmare la nuova versione perchè non posseggono la chiave privata dell'oggetto

Quando un processo necessita di accedere ad una risorsa, presenta all'oggetto le capability che possiede per quel oggetto
Quando a un oggetto è presentata una capability egli:

- verifica che abbia la sua firma
- verifica il nome
- verifica l'access control code
- verifica se l'accesso è permesso dall'access rights segnato nella capability.
  Le capability possono essere copiate e traferite ad un altro processo ma non possono essere modificate.

Revocazione degli access rights

Le revocazioni possono essere:

- immediate o ritardate
- selettive o generali
- parziali o totali (tutti i permessi o alcuni)

Revocazioni in sistemi basati su ACL:

- facile, è sufficiente aggiornare i permessi nella lista associata all'oggetto

Revocazione in sistemi basati su capabilities

- difficile, visto che i diritti di accesso non sono posseduti dall'oggetto ma sono distribuiti tra i processi tramite le capabilities, modificarli richiede di localizzarli tutti.
- Time limited capabilities
  - capabilities che hanno un expiration date dopo il quale devono essere rinnovate
  - se non sono rinnovate si cancellano
- Indirect capabilities
  - capabilities che non puntano direttamente ad oggetti ma a entità in tabelle intermedie che puntano agli oggetti
  - modificando le entries nelle tabelle intermedie si può simulare la revocazione immediata.

File ownership

Ogni processo creato da un utente inerita il suo user-id e group-id come real-user-id e real-group-id.
Quando un processo crea un nuovo file, il suo owner e gruppo diventano il real-user-id e real-group-id.

Ogni processo ha diversi id associati:

- real-user-id, real-group-id
  - identificano il vero utente e gruppo che lanciano il processo
  - sono letti dal file passwd
  - non cambiano durante l'esecuzione del processo
- effective-user-id, effective-group-id
  - impostato dinamicamente durante l'esecuzione del processo tramite il meccanismo del **setuid**
  - sono usati per determinare gli access rights del processo quando interagisce con il file system

Hybrid access control

Spesso i sistemi non sono puramente ACL o puramente Capabilities
Gli Hybrid access control combinano ACL e Capability mechanism per ottenere i vantaggi di entrmbi:

- Access control basto su identità - ACL
- Ease of revocation - ACL
- Efficienza degli accessi -Capabilities

Saved-user-ID

Ogni processo ha anche il saved-user-id e saved-group-id che contiene le copie dell' effective user id e del effective group id che esistevano al tempo dell'esecizione del programma setuid
Danno la possibilità al processo a ritornare al suo effettivo user/group id dopo l'esecuzione del setuid.

Normalmente:

- effective-user-id e real-user-id sono gli stessi
- effective-group-id e real-group-id sono gli stessi
  Al momento dell'esecuzione di un file con set-user-id:
- saved-user-id diventa effective-user-id
- effective-user-id diventa lo user id dell'owner del file
  Al momento dell'esecuzione di un file con set-group-id:
- saved-group-id diventa effective-group-id
- effective-group-id diventa group id dell'owner

Questo meccanismo permette a tutti gli utenti di eseguire l'esguibile con i permessi dell'owner dell'owner o gruppo dell'eseguibile.
Nuovi permessi rimangono in effetto solo durante l'esecuzione
Quando l'esecuzione termina i permessi ritornano allo stato precedente
Da la possibilità al processo di cambiare il suo dominio di protezione dinamicamente durante l'esecuzione
Può essere usato per implementare il "principle of least privilege"