Security Management, Risk, and Security Policy

Gestione della sicurezza

In un contesto di tipo aziendale con **gestione della sicurezza** si intende un processo più o meno formale basato su tre pilastri fondamentali:

- individuazione delle risorse aziendali da proteggere (DB, macchina singola...);
- individuazione delle possibili minacce;
- come contrastare queste minacce.

Tutta questa procedura di formalizzazione viene eseguita per determinare gli obiettivi di sicurezza da raggiungere, i quali anche in questo caso dovranno preservare le classiche proprietà di sicurezza (availability, confidentiality e integrity), permettendo il corretto funzionamento del processo aziendale.
Inoltre la gestione della sicurezza permette anche la stesura del **risk assessment.** documento che definisce in maniera concreta quali sono i possibili rischi, le soluzioni attuabili e gli eventuali costi da affrontare.

Sicurezza/Rischio

Il risk assessment permette di determinare il massimo rischio accettabile, ossia l'obiettivo ragionevole a fronte di un certo investimento nell'ambito della sicurezza.
Per fare tutto ciò ovviamente è necessario avere delle tecniche per valutare il rischio.

IT Sec. Management

Alcuni documenti che definiscono quali sono le procedure da seguire per la gestione del rischio.

Rischio

La cosa fondamentale da fare è quindi quella di definire il "rischio", nell'ambito della sicurezza informatica la sua definizione più corretta è: $R = A \times V \times T$, dove con A si indicano le risorse del sistema informatico, con V le vulnerabilità e T le minacce del sistema stesso.
Questa formula può essere anche vista come il prodotto tra il costo a cui viene sottoposta l'azienda se una certa risorsa viene attaccata (A) e la probabilità che quell'attacco avvenga effettivamente (VxT).
L'analisi del rischio avviene secondo due tipologie principali:

- qualitativa;
- quantitativa.

Assets

Con **assets** si intendono tutti i beni di un sistema IT che possono essere valutabili (sia HW sia SW ma anche dati e reputazione); in particolare la valutazione di ognuno di essi viene effettuata in termini di:

- quanto costa rimpiazzare la risorsa attaccata;
- quanto costa in termini di mancato guadagno.

Vulnerabilities

Le **vulnerabilità** sono tutti i punti deboli che possono essere sfruttati per causare danni al sistema, in generale può essere composta da una serie di fattori:

- librerie non aggiornate;
- software bug;
- controlli dell'accesso configurati in maniera non opportuna;
- regole di firewall che lasciano aperti servizi vulnerabili
- ...

In ambito aziendale è possibile effettuare una scansione mediante software per capire i servizi vulnerabili (ad esempio con nmap) e un controllo delle repositories.
Grazie a ciò è possibile effettuare all'interno della valutazione del rischio una classificazione delle vulnerabilità (alta, moderata o bassa).

Threats

Con **minacce** si intendono delle azioni di avversari che tentano di sfruttare le vulnerabilità per danneggiare un sistema IT rompendo le proprietà di sicurezza.
Un modo per capire quali sono le possibili minacce è quello di utilizzare gli attack tree, in cui vengono elencati tutti i possibili passi che permettono la realizzazione di un attacco.

Esempio analisi qualitativa

Ogni assets viene classificato in base a quanto è importante che venga protetto, la stessa cosa viene effettuata anche per ogni vulnerabilità per la quale verrà indicata la criticità.
Si può effettuare un'analisi di questo tipo anche per le eventuali minacce, per ognuna delle quali verrà indicata la probabilità di realizzazione.

Esempio analisi quantitativa

Ciò che è stato detto in precedenza può essere valutato con maggiore dettaglio effettuando un'analisi di tipo quantitativa.
Se si assegnano i valori 1, 5 e 10 rispettivamente per rischio basso, medio e alto allora il rischio può essere valutato nel seguente modo: unpatched software (medium), causing a denial of service attack (medium) against the server with the "InventoryAndOrders " database (medium) → 5 x 5 x 5 = 125.
A partire da questi calcoli è possibile andare ad intervenire per abbassare il rischio di eventuali situazioni critiche.

Microsoft STRIDE DREAD model

Le analisi del rischio non sono standardizzate, anche se un esempio può essere quello proposto da Microsoft nel 2008: l'approccio STRIDE DREAD model.
Esso presentava 6 categorie di minacce (STRIDE) e 5 categorie di rischio (DREAD).

Policies and Mechanisms

Una volta definito il rischio è necessario stabilire effettivamente cosa bisogna fare per evitare che succeda qualcosa di indesiderato, è necessario quindi stabilire una **politica di sicurezza** aziendale.
Essa specifica cosa è permesso e cosa no e può essere definita in:

- linguaggio naturale, ossia una serie di regole (in italiano, inglese…) che esprimono come un utente deve comportarsi;
- formalismo matematico, linguaggio più tecnico utilizzato per avere poi meccanismi di verifica automatici all'interno dell'azienda.

In ogni caso tutte le politiche si basano su dei **meccanismi**, come per esempio quello del controllo dell'accesso che viene utilizzato per prevenire azioni malevole o la cifratura, usata per evitare information disclosure.
Un esempio di regola aziendale potrebbe essere quello di non permettere a persone di portare chiavette USB dall'esterno.
Quando le politiche sono espresse in linguaggio naturale non in maniera chiara allora potrebbero esserci dei conflitti che potrebbero sfociare in possibili vulnerabilità.

Politiche di sicurezza

Una politica include un insieme ben definito di regole che include:

- soggetti che interagiscono con il sistema;
- oggetti e risorse del sistema di cui si vuole proteggere l'accesso (documenti, files…);
- azioni che i soggetti possono o meno fare sugli oggetti (read, write, delete);
- permessi;
- protezioni, specifiche regole che aiutano ad ottenere alcuni goal di sicurezza.

Goals of Security

In generale come goal di sicurezza generici si intendono:

- **prevenzione**, ossia prevenire che attaccanti possano violare la politica di sicurezza;
- **rilevamento,** ossia rilevare mediante strumenti adatti eventuali violazioni della politica di sicurezza da parte degli attaccanti.
  E' importante avere anche questo punto poiché nella maggior parte dei casi la prevenzione non risulta sufficiente in quanto gli attaccanti riescono comunque a violare la politica;
- **recovery,** si vuole che nonostante l'attacco il sistema continui a funzionare in maniera corretta e che esso sia in grado di reagire, valutando le conseguenze di questa azione e riparando il danno. Si vuole quindi tornare ad una situazione di sicurezza nel più breve tempo possibile.

Una politica di sicurezza da un punto di vista più formale definisce quali sono gli stati sicuri (autorizzati) di un sistema; quando avviene una violazione di sicurezza si entrerà quindi in uno stato non sicuro (non autorizzato). Detto ciò un sistema è sicuro se partendo da uno stato autorizzato non entrerà mai in uno che non lo è.
Esempio violazione politica a pag. 8

Assumptions

Una politica di sicurezza si fonda su delle **assunzioni**, ad esempio se vogliamo proteggere delle risorse all'interno di una stanza essa verrà chiusa con un lucchetto; l'assunzione in questo caso è quella che nessuno sia in grado di aprire questo lucchetto.
Una politica si basa quindi su una serie di assiomi che devono sempre essere ben verificati, inoltre le due principali assunzioni che vengono definite sono:

- la politica partiziona l'insieme degli stati del sistema in sicuri e non sicuri;

- il meccanismo di sicurezza previene che il sistema entri in uno stato non sicuro.

Se questi due punti sono errati il sistema entrerà in uno stato non sicuro.

Mechanisms and trust

Nelle politiche vengono inoltre utilizzati dei **meccanismi** (per la maggior parte crittografici); la politica di sicurezza come già detto definisce quali sono gli stati sicuri (Q) e non sicuri, mentre i meccanismi di sicurezza definiscono a loro volta un sottoinsieme degli stati sicuri (R).
Un meccanismo di sicurezza si definisce:

- **sicuro** se $R \subseteq Q$, ossia se gli stati sicuri in cui il sistema può andare sono solo quelli definiti dalla politica di sicurezza;
- **preciso** se $R = Q$, ossia gli stati sicuri definiti dal meccanismo coincidono con quelli della politica;
- **broad** se esistono stati che appartengono a R ma non a Q, ossia l'insieme degli stati sicuri del meccanismo risulta più largo di quello della politica.

In generale l'unione di tutti i singoli meccanismi di sicurezza messi in atto vorrei che non mi facesse uscire il sistema dagli stati Q sicuri definiti dalla politica, ciò che però succede nella pratica è che il sistema può entrare in alcuni stati non sicuri (ad esempio perché il meccanismo di cifratura viene rotto). Questa situazione ricade nella categoria broad.
Il fatto di affidarsi ad un certo meccanismo robusto richiede inoltre di considerare alcune assunzioni che devono essere verificate:

- ogni meccanismo è progettato per implementare una o più parti della politica di sicurezza;
- l'unione dei meccanismi deve implementare tutti gli aspetti della politica;
- i meccanismi devono essere implementati correttamente;
- i meccanismi devono essere installati e amministrati correttamente.

Policies and Standards

Quando si definisce una politica di sicurezza è necessario considerare anche delle regole a più alto livello, come quelle definite dal GDPR il 25 maggio 2018.
Altri esempi sono relativi al modello statunitense.

Security management standards

ISO 17799 definisce principi generali per la gestione della sicurezza nell'ambito aziendale:

- definizione della politica di sicurezza organizzativa;
- definizione dell'infrastruttura di sicurezza organizzativa;
- classificazione e controllo delle risorse;
- sicurezza fisica e ambientale;
- sicurezza del personale;
- gestione delle comunicazioni e delle operazioni;
- controllo dell'accesso;
- sviluppo e manutenzione dei sistemi;
- continuità del business;
- conformità.

Type of Access Control

Come detto in precedenza la politica di sicurezza specifica chi ha l'accesso a qualcosa e quali azioni quel qualcuno può effettuare. Questo tipo di regolamentazione si chiama controllo dell'accesso e si divide in tre categorie standard:

- **DAC** (Discretionary Access Control), esempio di Linux in cui ogni utente ha dei meccanismi di controllo dell'accesso su ogni oggetto che lui possiede. Ciascun utente può determinare i permessi per altri utenti, per questo motivo sono chiamate discrezionarie;
- **MAC** (Mandatory Access Control), in questo caso un amministratore di sistema stabilisce delle regole standard secondo le quali i vari oggetti possono essere acceduti. Nessun utente può andare ad alterare tali regole;
- **ORCON** (Originator Controlled Access Control), in questo caso il creatore dell'oggetto è l'unico che può specificarne le regole di accesso.

Trust management system

Le politiche di controllo dell'accesso sono parte di un sistema di gestione della fiducia, sistema formale che serve per specificare la politica di sicurezza.
In questo contesto gli utenti possono eseguire solo le azioni autorizzate dalla politica e per assicurare che le regole all'interno di essa siano messe in atto in modo corretto si utilizza un compliance checker.

Confidentiality Policy

Una prima tipologia di politica di sicurezza è quella basata sulla confidenzialità, il cui obiettivo è quello di prevenire la divulgazione non autorizzata di informazioni.
Questo modello ha origine in ambito militare, in cui ogni informazione viene catalogata secondo diversi livelli di classificazione (confidenziale, segreta, top secret) ed ognuna di esse può essere letta solo da qualcuno il cui livello è alto almeno come quello dell'informazione stessa.
Sono di particolare interesse in questo caso le politiche mandatorie, nello specifico la loro variante più diffusa ossia quella delle politiche multilivello.

Bell-La Padula Model

Questo modello fa parte delle politiche multilivello basate sulla confidenzialità delle informazioni. I livelli di sicurezza secondo i quali vengono classificati oggetti e soggetti sono: top secret, secret, confidential e unclassified.
Esempio banale a pag. 13

Reading and writing Information

L'obiettivo del modello Bell-La Padula è di prevenire flussi di informazioni da classi di sicurezza basse verso classi di sicurezza più alte o non comparabili. Per evitare questo flusso improprio vengono applicati due principi:

- **NO READ UP**, se c'è un soggetto che ha una determinata classe di sicurezza e vuole eseguire delle operazioni di lettura, esse non possono essere eseguite su oggetti che hanno classi di sicurezza maggiore rispetto a quella del soggetto stesso;

- **NO WRITE DOWN**, se c'è un soggetto che ha una determinata classe di sicurezza e vuole eseguire delle operazioni di scrittura, esse non possono essere dirette verso oggetti la cui classe di sicurezza è più bassa del soggetto stesso.

Combinando i due principi non vi possono essere flussi di informazione che vanno dall'alto verso il basso e sono proprio i flussi che voglio bloccare per garantire la confidenzialità all'interno del sistema.

Basic Security Theorem

Definite queste due proprietà allora uno stato del sistema è **sicuro** quando sono soddisfatte:

1. la simple security property, ossia qualunque sia l'accesso eseguito nel sistema se l'operazione è di lettura allora vale la proprietà di NO READ UP;
2. la star security property, ossia qualunque sia l'accesso eseguito nel sistema se l'operazione è di scrittura allora vale la proprietà di NO WRITE DOWN.

Questo equivale a dire che se il sistema parte da uno stato sicuro e ogni transazione del sistema soddisfa queste due proprietà allora ogni stato del sistema è sicuro per induzione.

PARTE LABORATORIO DA 1H:50M VIDEOLEZIONE 17/10

Extension with Lattices

Questo modello estende la nozione di livello di sicurezza includendo anche delle categorie che inducono un **reticolo**: ogni oggetto e ogni soggetto sarà rappresentato da un livello (TS, S, C, U) e una categoria.
Esempio reticolo a pag. 15

Dominance

Come viene gestita la dominanza in questo contesto dove c'è l'aggiunta di queste categorie? Si dirà che un livello di sicurezza (L, C) **domina** il livello di sicurezza (L', C') se e solo se L <= L' e C' $\subseteq$ C.
Detto questo è possibile adattare le due regole del modello di Bell - La Padula anche nel contesto dei reticoli.
Esempi dominanza a pag. 15

Implementation of MLS

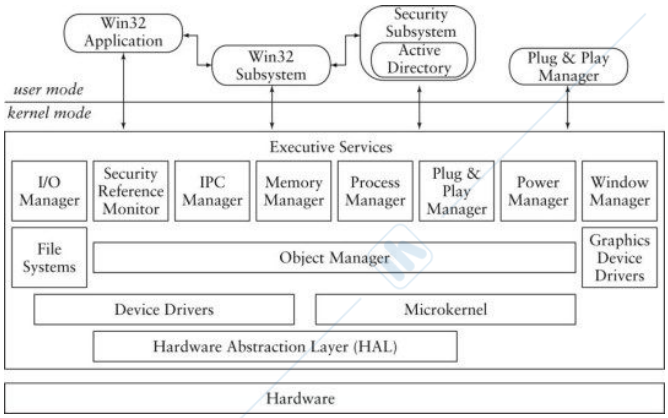
Come abbiamo detto Bell - La Padula è un esempio di politica multilivello, in cui esiste un'autorità che definisce le regole di accesso.
Le politiche multilivello possono essere implementate in diversi altri modi ma in generale l'idea base è quella di basarsi su un **reference monitor**, struttura astratta all'interno del SO la quale ha il compito di verificare i permessi relativi ad ogni richiesta di accesso.
Esso è implementato nel **security kernel**, parte hardware, firmware e software che fornisce i meccanismi di protezione delle risorse.
Dal punto di vista hardware e software la totalità dei componenti necessari per la sicurezza del sistema è detta **Trusted Computing Base** (TCB).

Windows security

Per quanto riguarda la sicurezza di Windows si può distinguere:

- **user mode,** istruzioni non critiche per la sicurezza eseguibili lato utente;
- **kernel mode**, istruzioni privilegiate eseguibili solo in modalità kernel.

La distinzione tra le due modalità avviene mediante un flag di stato all'interno di un registro del processore.



Objections to BLP

Il problema del modello BLP riguarda alcuni processi di sistema, in particolare di gestione memoria o che devono avere permessi di lettura/scrittura su tutto.
Essi infatti non potendo abbassarsi di livello non possono scrivere un file lato utente poiché altrimenti violerebbero il principio di no write down; una possibile soluzione a tale problema è di inserire tutto il necessario all'interno della TCB.
Tutto questo però causerebbe man mano l'aumento della dimensione della TCB stessa, cosa che renderebbe più complicata la verifica delle regole del modello BLP; inoltre è stato provato che in tale modello non è vietato che un utente possa temporaneamente abbassarsi di livello.
Se però questo avvenisse per tutti gli utenti allora essi potrebbero accedere a tutti gli oggetti causando uno stato del sistema non sicuro. Per difendersi Bell disse che questo downgrade può essere effettuato solo se viene autorizzato, ossia solo se il modello di sicurezza lo prevede: non è quindi un problema del modello di Bell - La Padula.

Covert Channels

Ciò che mise in crisi veramente il modello BLP è il paper che introdusse i cosiddetti **covert channels**, canali che non sono stati progettati per trasportare delle informazioni ma che in realtà lo fanno.
Essi infatti entrano in gioco in presenza di due processi, uno di alto e uno di basso livello; essi non potrebbero comunicare tra loro ma attraverso questi covert channels sono in grado di attuare meccanismi di segnalazione anche a livello di singolo bit.
Questo risulta essere molto dannoso anche se i due processi non possono comunicare tra loro, per esempio è possibile posizionare al tempo t la testina del disco su una certa posizione.
Per esempio passando il bit 1 nel covert channel allora sarà messa in una certa posizione dal processo più basso, se 0 in un'altra: il processo di più alto livello anche se non può scrivere può causare comunque il posizionamento della testina.
Ripetendo questo processo per n bit allora si aprirà un vero e proprio canale di comunicazione.

BLP in Multics

Nonostante ciò il modello di Bell - La Padula venne comunque utilizzato per lo sviluppo di un SO predecessore di Unix chiamato **Multics**, che doveva essere sicuro, affidabile e multi-utente.

In questo caso i soggetti sono i processi del sistema, ciascuno di essi ha associato un suo descrittore con tutte le informazioni che lo identificano.
Gli oggetti sono invece segmenti di memoria o device I/O e a loro volta hanno associati dei descrittori chiamati Segment Descriptor Word (SDW) i quali contengono nome oggetto, puntatore e un flag per i vari permessi di accesso (r, w, e).
A livello dei processi sono inoltre presenti la Process Level Table e la Current Level Table, che definiscono i livelli di sicurezza dei soggetti.
Infine la Active Segment Table è la tabella che contiene i processi attivi.

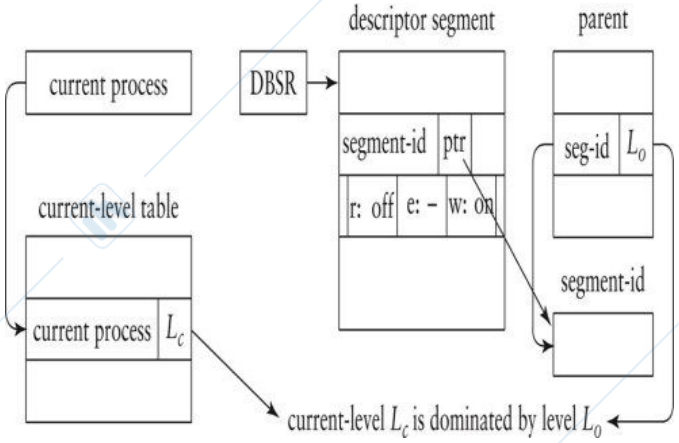
Gli oggetti anche in Multics sono organizzati in alberi di directory, le informazioni sugli oggetti stanno quindi nella directory genitore.
Per accedere a un oggetto un processo deve attraversare tutto l'albero analizzando i permessi: essere ad un livello più alto significa essere accessibili per esempio solo a livello di root, a livello più basso invece anche a livello utente; se una directory non risulta accessibile allora non lo sarà nemmeno l'oggetto in questione.
Le regole di Bell - La Padula sono state adattate in Multics, in particolare considerando ogni SDW di un processo attivo il livello corrente del processo:

- domina il livello del segmento se r è settato e w no;
- è dominato dal segmento se r è off e w on;
- è uguale se r e w sono on.

Come detto in precedenza tutti i processi attivi sono contenuti nella Active Segment Table e, di volta in volta, delle strutture contenute nel SO definiscono quale è il processo corrente il quale viene salvato all'interno del SDW, nel segmento descrittore relativo proprio ai processi attivi.
E' inoltre presente anche una matrice di controllo degli accessi che per ogni oggetto definisce quali sono i relativi permessi.
Infine è possibile valutare quale è il livello di sicurezza di un certo soggetto prendendolo dalla Process Level Table e al contempo si può risalire al livello di sicurezza di un oggetto prendendone i permessi definiti su di esso o ereditati dalla directory parent.
Si può quindi capire che il SO fornisca tutto il necessario per valutare se un soggetto possa avere accesso ad un oggetto o meno.

Multics security

Come è possibile valutare la * - property in Multics?
Il livello di sicurezza del processo corrente che chiameremo L(c) è presente all'interno della current-level table; il puntatore presente all'interno della struttura chiamata Descriptor Segment Base Register (DSBR) punta invece all'inizio del descriptor segment del processo corrente.
Supponiamo che questo descriptor segment contenga l'SDW di un oggetto il cui accesso è solo in scrittura: il flag read è quindi off mentre quello write è on.
Per verificare la * - property il livello di sicurezza dell'oggetto L(o) viene estratto dalla parent directory e comparato con L(c) per verificare che L(c) >= L(o).



The Data General B2 UNIX System

Come nel caso precedente il data general B2 UNIX system implementa una politica di controllo dell'accesso di tipo mandatoria, in particolare basandosi sul modello Bell - La Padula in cui un livello alto non può scrivere in uno più basso.
Nei livelli più bassi come VP - 2 saranno presenti i dati trusted come le chiamate di sistema, in particolare questa regione è chiamata **Virus Prevention Region** appunto perchè si vuole evitare che malware o virus possano andare a modificare questa regione di codice.
Inoltre nessun processo utente potrà andare a scrivere in tali livelli.
In questo caso il livello di sicurezza viene identificato attraverso un'etichetta MAC e inizialmente ad ogni soggetto viene assegnata un'etichetta derivata dalla directory parent o comunque contenuta nell'authorization and authentication (A&A) database definito dal sistema operativo stesso.
Gli oggetti invece possono avere delle etichette di due tipologie:

- **esplicite** quando sono attribute da qualche soggetto;
- **implicite** quando sono derivate dalla directory parent.

Questo sistema applica la nozione di dominanza del modello BLP.

Integrity policies

Una seconda categoria di politiche di sicurezza è quella che mira a preservare l'integrità, i dati possono quindi essere letti ma non possono essere modificati da utenti che non hanno le autorizzazioni.
I principi di base sono:

- **separazione dei doveri,** ossia se per eseguire un processo ho due fasi queste devono essere fatte da due soggetti diversi;
- **separazione delle funzioni**, ossia si vuole che lo sviluppo e il testing siano due operazioni separate per evitare che il secondo venga influenzato dallo primo;
- **auditing,** ossia avere delle nozioni che mi permettano di tenere traccia delle responsabilità (chi ha modificato un file e quando) e possibilità di recovery.

Biba Integrity Model

Uno dei modelli più importanti per quanto concerne le politiche mandatorie per l'integrità è il modello di Biba.
L'obiettivo primario è quello di bloccare i flussi di informazione verso classi di integrità alte o incomparabili. Il modello di Biba, o più in generale, i modelli basati su politiche mandatorie per l'integrità applicano dei principi che sono duali a quelli delle politiche mandatorie per la segretezza. Di conseguenza i principi applicati nel modello Biba sono duali rispetto ai principi applicati nel modello di Bell e La Padula.
Anche qui le due proprietà che sono modellate all'interno del modello Biba sono:

1. **simple property,** modellizzazione del principio di NO READ DOWN. Un soggetto s può eseguire un'operazione di lettura su o solo nel caso in cui la classe di integrità dell'oggetto o domina, o al più uguale, la classe di integrità del soggetto s;
2. **star property,** modellizzazione del principio di NO WRITE UP. Un soggetto s può eseguire un'operazione di scrittura su o solo nel caso in cui la classe di integrità del soggetto s domina, o al più uguale, la classe di integrità dell'oggetto o.

Tali proprietà modellano i principi di NO READ DOWN e NO WRITE UP la cui applicazione mi garantisce il blocco di flussi di informazione verso livelli di integrità più alti o incomparabili.
Esempi a pag. 8

Windows and Biba

A partire da Windows Vista è stato adottato un modello simile a quello di Biba.
Ad ogni file viene associato un livello di integrità: low, medium, high o system e viene adottato il principio di NO WRITE UP.

Clark-Wilson Integrity Model

Altro esempio di modello di sicurezza per l'integrità e in cui il sistema si evolve facendo delle transazioni che possono essere eseguite se e solo se si finisce in uno stato ancora valido.
L'integrità è definita secondo un certo insieme di vincoli:

- i dati risultano validi se soddisfano tali vincoli;
- l'integrità del sistema si mantiene attraverso l'esecuzione di una transazione (valida);
- una transazione ben formata sposta il sistema da uno stato consistente ad un altro.

Entities and rules

Nel modello Clark-Wilson è possibile definire:

- **Constrained Data Items** (CDI), tutti quegli oggetti all'interno del sistema ai quali deve essere applicato il modello d'integrità;
- **Unconstrained Data Items** (UDI), oggetti che non sono sottoposti ai vincoli di integrità;
- **Integrity Verification Procedures** (IVP), procedure che permettono la verifica dell'integrità. Il loro obiettivo è quello di confermare che tutti i CDI siano conformi alle specifiche di integrità ogni volta che una IVP è eseguita;
- **Transformation Procedures** (TP), sono tutte quelle procedure che possono modificare i CDI o possono prendere un input arbitrario dagli utenti e creare proprio i CDI. Esse corrispondono proprio a transazioni ben-formate, in quanto portano un set di CDI da uno stato valido ad un altro.

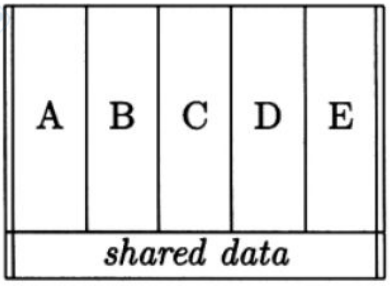
In questo contesto sono definite delle regole di certificazione che permettono di dire quando una transazione verifica i vincoli d'integrità.
Per prima cosa è necessario verificare che le procedure siano certificate, ossia che i vincoli di integrità non vengano modificati, se ciò è vero allora si può dire che quel metodo è certificato e non provocherà danni al sistema stesso.
Ci sono inoltre anche delle regole che garantiscono la separazione dei doveri tra esecuzione e certificazione delle transazioni.

Confronto con Biba

Quali sono i fattori comuni e non tra i due modelli?
Nel modello di Clark-WilsonRispetto a Biba si perde la nozione di multilevel, il quale si focalizza sulle transazioni, sul rispetto dei vincoli d'integrità e sulla separazione dei doveri.
E' però possibile dimostrare che CW possa implementare Biba, ma non il viceversa.

Multilateral Security

Finora sono stati analizzati casi relativi alla sicurezza multilivello, in cui sono appunto presenti con livelli crescenti di sicurezza.
Nel caso della **sicurezza multilaterale** invece non sono i flussi di dati da un livello all'altro che si vogliono bloccare, ma bensì quelli tra dipartimenti che operano allo stesso livello come nel caso sotto riportato.
In questo esempio i dipartimenti lavorano sugli stessi dati ma non si vogliono flussi tra A,B,C,D,E.



Multilateral Security Model

Esistono tre principali modelli che permettono di implementare la sicurezza multilaterale:

- **compartimentazione;**
- **chinese wall;**
- **BMA mode**l.

Compartmentation and Lattice Model

Consiste nella formazione di reticoli in cui c'è l'utilizzo di parole chiave e in cui è presente il concetto di dominanza.
Due oggetti A e B si trovano in una **relazione di dominanza** se posso definire che A > B o viceversa; nell'esempio riportato a pag. 11 si può dire ad esempio che SECRET > UNCLASSIFIED.
Può però succedere che certe volte la relazione di dominanza non sia definita, come per esempio tra (SECRET, [CRYPTO, FOREIGN]) e (TOP SECRET), in quanto stanno su due

rami diversi del reticolo e quindi non sono confrontabili in quanto non connessi da un arco: si dicono incomparabili.

Chinese wall model

Una seconda tipologia di applicazione della sicurezza multilaterale è il **chinese wall**, il quale si basa sulla separazione dinamica dei doveri per proteggere la segretezza dei dati.
L'obiettivo è quello di prevenire flussi di informazioni che possano causare conflitti di interessi per consulenti individuali (per esempio uno di essi non dovrebbe avere informazioni su due banche diverse ma solo di una).
In questo modello gli oggetti vengono organizzati in maniera gerarchica secondo tre livelli:

- **oggetti base**, per esempio i file;
- **datasets aziendali**, ossia gruppi di oggetti che fanno riferimento ad una stessa azienda;
- **classi di conflitto di interesse**, gruppi di datasets appartenenti a compagnie che sono in competizione tra loro.

Esempio a pag. 12

Come vengono regolati in questo contesto gli accessi in lettura?
Vengono regolati attraverso la cosiddetta simple security rule, mediante la quale un soggetto *s* può accedere ad un oggetto *o* solo se:

- *o* è nello stesso dataset aziendale di tutti gli oggetti a cui *s* ha già acceduto;
- *o* appartiene ad una differente classe di conflitto di interesse.

La simple security rule in certi contesti risulta però un po' stringente, in quando gli utenti potrebbero aver bisogno di comparare informazioni provenienti da diverse compagnie: questa problematica è risolvibile attraverso la sanitizzazione.
Questo processo permette di mascherare un'informazione aziendale, in particolare per prevenire la scoperta dell'identità della società.

Come vengono regolati invece gli accessi in scrittura?
Vengono regolati attraverso la cosiddetta *-property, mediante la quale un soggetto *s* può scrivere un oggetto *o* solo se:

- l'accesso è consentito dalla simple security rule;
- nessun oggetto può essere letto da *s* (in accordo con le autorizzazioni) se è in un diverso dataset di *o* e contiene informazioni non sanitizzate.

Esempio a pag. 14

The BMA Model

Il modello BMA è specializzato invece sui dati sanitari, all'interno di tale modello vengono inoltre definiti dei **ruoli** (medico responsabile, paziente…) e delle regole per la gestione dei flussi informativi relativi ai dati sanitari stessi, per far si che il sistema evolva il modo sicuro.
Avendo definito dei ruoli risulta essere più espressivo del modello Bell - La Padula.

Other policy models

Altri esempi di politiche di sicurezza sono:

- CISS policy (Clinical Information Systems Security Policy), sempre relativo ai dati sanitari ma che rispetto al BMA combina sia integrità che confidenzialità;

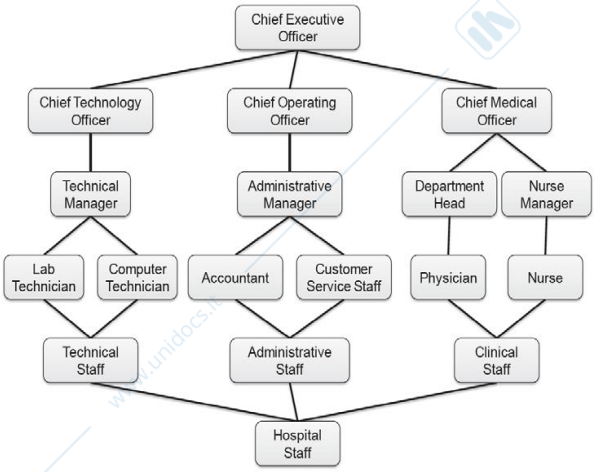
- ORCON (originator controlled), in questo caso l'organizzazione di un certo documento è colei che ne controlla la disseminazione.
  Per esempio un certo soggetto s definisce che l'organizzazione X è l'originataria di un certo oggetto o, allora X definirà le regole secondo cui l'oggetto potrà essere rilasciato ad altre organizzazioni.

Role-Based Access Control

Queste politiche si basano sulla definizione di **ruolo**, cioè un insieme di privilegi necessari per eseguire un determinato compito in un'organizzazione. L'accesso degli utenti agli oggetti del sistema è proprio mediato dai ruoli. Le autorizzazioni non vengono più quindi concesse agli utenti ma ai ruoli: agli utenti viene concessa l'autorizzazione di attivare uno o più ruoli. A seguito dell'attivazione di un determinato ruolo *r,* l'utente acquisisce tutti i privilegi di accesso che sono stati concessi al ruolo stesso. I privilegi non sono più validi quando il ruolo è disattivato.
Esiste una differenza tra il concetto di gruppo, cioè insieme di utenti, e ruolo, cioè insieme di privilegi.
Esempio a pag. 16

Hierarchical RBAC

All'interno di un sistema i ruoli possono essere definiti in diversi modi. Una tipologia che ha preso molto piede consiste nell'introdurre una **gerarchia di ruoli**, che definisce delle specializzazioni secondo ordinamento parziale. Il fatto di avere definito questa gerarchia permette la propagazione delle autorizzazioni: se ad un ruolo r viene concessa l'autorizzazione di eseguire una certa azione su un certo oggetto allora anche tutti i ruoli che sono specializzazione di r potranno eseguire quell'azione su quell'oggetto. Se poi ad un utente u viene concessa l'autorizzazione ad attivare un ruolo r allora u potrà attivare anche ogni generalizzazione di r.



Role of "trust"

Le politiche di confidenzialità viste in precedenza si focalizzano su chi può avere accesso alle informazioni, mentre quelle di integrità su chi può modificare i dati o meglio che ciò avvenga da una parte trusted.
Gli oggetti a livello più alto sono considerati più trusted e in particolare le politiche di integrità sono più dipendenti da questo concetto di **fiducia**.
Esempio software a pag. 17

Decoupling Policy, Mechanism

Le politiche di sicurezza per essere implementate hanno bisogno di **meccanismi**, i quali devono essere forti come le politiche stesse per garantire la sicurezza.

Ciò che accade però è che il passaggio dalla politica all'implementazione è problematico, poiché il software è molto complesso e risulta essere complicato effettuare verifiche formali a causa dei numerosi aspetti da tenere in considerazione.

Security and Trust

Il passaggio tra politica, implementazione software e la conseguente implementazione hardware tiene conto di numerose **assunzioni** riguardanti la sicurezza che devono essere valide quando si vuole dimostrare la sicurezza del sistema.
Ad esempio siamo sicuri che il compilatore compili e basta?
Una violazione di questa sicurezza fu dimostrata da Ken Thompson nella nella sua “ACM Turing Award Lecture”, in cui esso presentò un malware ibrido il quale consisteva in uno speciale compilatore C eseguibile.
Questo ibrido, oltre che a compilare appunto codice C, aveva due funzionalità aggiuntive:

- quando veniva compilato il codice sorgente del login, il compilatore inse-riva una backdoor per bypassare l’autenticazione tramite password;
- andando poi a compilare il codice sorgente del compilatore stesso, non consapevolmente gli utenti creavano una sorta di replica eseguibile di esso.

Questo speciale compilatore era così un Trojan horse che si auto-replicava come un virus e che creava back doors.
Questo dimostra che non ci si può fidare completamente di codice non scritto da noi stessi.

PARTE LABORATORIO DA 1H:54M VIDEOLEZIONE 22/10