



UNIVERSITÀ DI PERUGIA
Dipartimento di Matematica e Informatica



Appunti *Cybersecurity*

Bistarelli

Anno Accademico 2021-2022

Last Update: 9 agosto 2023

Indice

1	Introduzione	6
1.1	La Sicurezza Informatica	6
1.2	C.I.A.	8
1.3	AAA	10
1.4	Concetti Correlati	11
1.4.1	Assurance	11
1.4.2	Minacce	11
1.4.3	Policy e Meccanismi	13
2	Autenticazione	14
2.1	Tipi di Autenticazione	14
2.1.1	Autenticazione utente-computer	15
2.1.2	Autenticazione su conoscenza	15
2.1.3	Gestione delle password	15
2.1.3.1	Vulnerabilità delle password	16
2.1.3.2	Come difendersi ?	17
2.1.4	Autenticazione su possesso	17
2.2	Autenticazione su caratteristiche	18
2.3	Kerberos	19
2.3.1	Come funziona ?	20
2.3.2	Nel dettaglio	22
2.3.3	Gestione delle Chiavi	24
2.4	Intruder	25
3	Crittografia	28
3.1	Introduzione	28
3.2	Crittografia a chiave Privata (simmetrica)	28
3.3	Crittografia a chiave Pubblica (asimmetrica)	29
3.4	Funzioni Hash	30
3.5	Firma Digitale	31
3.5.1	Creazione della Firma	32



3.5.2	Verifica della Firma	33
3.6	Certificato Digitale	33
3.6.1	Formato x.509	34
3.6.2	Ottener un certificato Digitale	35
3.6.3	Revoca del certificato	36
3.6.4	Certificate Revocation List	36
3.7	Protocolli di Sicurezza	36
3.8	Autenticazione di utenti remoti con Needham-Schröder (1978)	37
3.8.1	Attacco di Lowe (1995)	39
3.9	Protocollo di sicurezza di Woo-Lam (anni 80)	40
3.9.1	Attacco su Woo-Lam	41
4	Policy	43
4.1	Confidentiality Policies	48
4.1.1	Modello Bell-LaPadula	50
4.1.2	Multilevel Security	51
4.1.2.1	Channel Cascade Attacks	52
4.1.3	Secure Interoperation	54
4.1.4	Access Interoperation	54
4.1.5	Secure Reconfiguration	55
4.2	Integrity Policies	56
4.2.1	Modello BIBA	57
4.2.2	Modello Clark-Wilson	58
4.2.2.1	Come funziona?	60
4.2.2.2	Regole di certificazione	63
4.2.2.3	Regole di rinforzo	64
4.2.2.4	Utenti e regole	64
4.2.2.5	Uso dei Log	64
4.2.2.6	Trattamento di input non fidato	65
4.2.2.7	Separazione dei compiti	65
4.2.2.8	I principali limiti	65
4.2.2.9	Biba vs. Clark-Wilson	66
4.3	Hybrid Policies	66
4.3.1	Chinese Wall	66
4.3.1.1	Chinese Wall vs Bell-LaPadula	68
4.3.1.2	Chinese Wall vs Clark-Wilson	69
4.3.2	ORCON 	69
4.3.3	RBAC	70
4.3.3.1	Modello RBAC-NIST	70

5 Risk Management Process	73
5.1 Risk Assessment	73
5.2 Risk Mitigation	74
5.2.1 Attack Tree	74
5.2.2 Cp-nets	76
5.2.3 Indici Economici	78
5.3 Risk Monitoring	80
6 Security Design	81
6.1 Principi Fondamentali	81
6.2 Trust Network Brainstorming	86
7 Database & Data Center Security	87
7.1 Introduzione	87
7.2 SQL Injection Attack	88
7.2.1 Attack Avenues	90
7.2.2 Attack Types	91
7.3 SQL Injection Mitigation	92
8 Blockchain & Bitcoin	94
8.1 Introduzione	94
8.1.1 Blockchain	94
8.1.2 Bitcoin	95
8.1.2.1 Bitcoin Core	97
8.1.2.2 Bitcoin Actors	97
8.1.2.3 Miner	98
8.1.2.4 Escrow Contracts	98
8.2 Sicurezza e Blockchain	99
8.2.1 Double Spending	99
8.2.1.1 51% Attack	99
8.2.2 Wallet attack	100
8.2.3 Transaction Malleability	100
8.2.4 Sybil Attack	100

*“Oي, con quanto sentimento
defeco sul tuo naso,
così che ti coli sul mento.”*

Wolfgang Amadeus Mozart

Capitolo 1

Introduzione

1.1 La Sicurezza Informatica

La sicurezza non è un prodotto ma un processo da compiere e realizzare. La sicurezza di un intero sistema è data in realtà dalla sicurezza dell'anello più debole che ne fa parte. Esistono diversi livelli di sicurezza, ma non ne esiste una assoluta, in quanto questa dipende dal contesto specifico nel quale il sistema opera. Un sistema è **sicuro** se, una volta entrato in uno stato sicuro, non attraversa più uno stato non sicuro. Crittografia, firewall, antivirus, utilizzo di password e smart card sono solo dei meccanismi, degli strumenti e delle tecniche usati per ottenere una maggiore sicurezza. Il loro utilizzo deve essere ponderato in relazione al loro costo: bisogna sempre chiedersi *“conviene fare l’investimento per acquistare un firewall, ecc... o conviene risparmiare nell’ambito della sicurezza?”*. In sostanza, va sempre valutato il rapporto costo/benefici.

Un **piano di sicurezza** è composto da diverse fasi:

- **evitare** i rischi: per esempio, collegarsi ad internet solo quando necessario;
- **deterrenza**: pubblicizzare strumenti di difesa e di punizione;
- **prevenzione**: impiego di strumenti per prevenire gli attacchi, ad esempio i firewall;
- **individuazione**: utilizzo di strumenti per IDS;
- **reazione**: un esempio può essere il ripristino del sistema, ma occorre aver prima programmato delle procedure di backup;



- eventuale **denuncia** presso un tribunale per risarcimento danni.

Alcune soluzioni contro gli attacchi informatici potrebbero essere:

- una buona pianificazione della rete con hardware adeguato (router, switch ecc.) insieme alla divisione della rete in aree a livello di sicurezza variabile;
- controllo dell'integrità delle applicazioni (bugs free) e verifica della correttezza delle configurazioni;
- utilizzo di software che controllino e limitino il traffico di rete dall'esterno verso l'interno e viceversa (es. firewall, router screening...);
- utilizzo di applicazioni che integrino algoritmi di crittografia in grado di codificare i dati prima della loro trasmissione in rete (es. PGP, SSH, SSL, ecc.).

La sicurezza:

- richiederebbe spesso il ridisegno del sistema, il che non è sempre possibile, ovviamente, perché è assai costoso farlo;
- è una proprietà di vari livelli architetturali (OS, rete, ecc.);
- non deve essere sicura solo la rete o solo il sistema operativo, ma tutto nel complesso;
- non è un semplice predicato booleano, infatti al quesito “il sistema è sicuro?” non si può rispondere semplicemente sì o no;
- è costosa nel senso di risorse computazionali, gestione, mentalità, utilizzo;
- rimane un campo aperto anche per i colossi dell'Informatica.

In generale per sicurezza si intende l'assenza di rischio o pericolo. La **Sicurezza Informatica** è, quindi, l'insieme delle azioni di prevenzione o protezione contro accesso, distruzione o alterazione di risorse/informazioni da parte di utenti non autorizzati. Quale di questi attacchi è più pericoloso dipende chiaramente dal contesto.

Sistema Sicuro: possiamo infine definire un *Sistema Sicuro* come un sistema che opera e lavora per garantire:

- Riservatezza,
- Integrità,
- Disponibilità,
- Autenticità,
- Non ripudio.

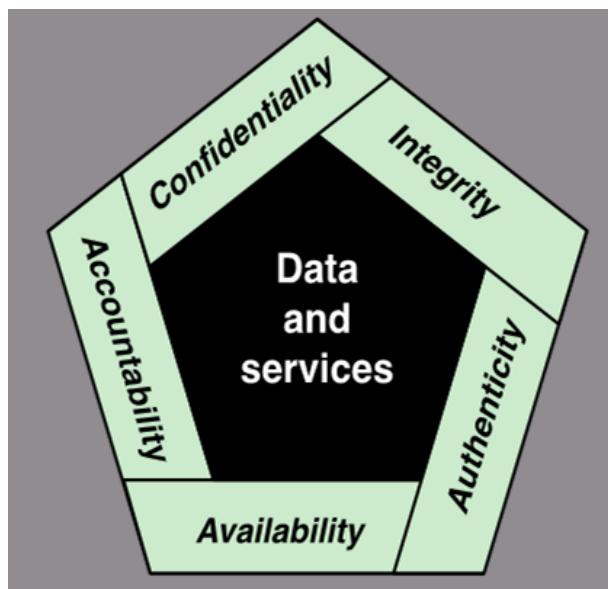


Figura 1.1: Requisiti minimi per la Sicurezza Informatica.

1.2 C.I.A.

Quando si parla di protezione e sicurezza dei dati informatici, esistono **tre principi fondamentali** su cui focalizzare l'attenzione, in nome di una corretta gestione: vale a dire **Confidentiality** (*Riservatezza*), **Integrity** (*Integrità*) e **Availability** (*Disponibilità*) (detta anche “*Triade C.I.A.*”). Tali principi devono essere ricercati in ogni soluzione di sicurezza, tenendo conto anche delle implicazioni introdotte dalle vulnerabilità e dai rischi. Per questo motivo vengono detti “componenti base”:

Riservatezza dei Dati: Una strategia volta alla privacy informatica deve in prima battuta offrire riservatezza, ovvero garantire che i dati e le risorse siano preservati dal possibile utilizzo o accesso da parte di soggetti non autorizzati. Ciò deve valere per tutte le fasi di vita del dato. Le cause di violazione della riservatezza possono essere imputabili ad un attacco malevolo oppure ad un errore umano. Le modalità di attacco possono essere molteplici: si va per esempio dalla sottrazione di password all’intercettazione di dati su una rete. Meccanismi per garantire la riservatezza possono essere:

- controllare l’accesso ai dati cifrandoli: solo chi ha la chiave riesce ad accedervi;
- meccanismi dipendenti dal sistema (sono a più basso livello).

Se viene usato un certo meccanismo, significa che si sono fatte delle assunzioni di affidabilità, quindi si presuppone che esso sia funzionante e perciò posso fidarmi. Si potrebbe argomentare, in realtà, che la riservatezza debba anche affrontare, non solo la necessità di voler nascondere il contenuto di un documento da occhi indesiderati, ma anche la sua esistenza. Se consideriamo, ad esempio, l’analisi di un sistema di comunicazione, potremmo certamente non essere in grado di leggere i messaggi tra due utenti, ma allo stesso tempo potremmo derivare informazioni importanti in base alla frequenza di scambio di messaggi. Pensiamo, ad esempio, alla frequenza di messaggi scambiati da un Server bancario rispetto a quella di un semplice utente. La definizione di riservatezza dovrebbe essere in grado quindi di coprire un altro aspetto, chiamato *Unlinkability*: due o più oggetti di interesse (messaggi, azioni, eventi, utenti) non sono correlabili se un attaccante non è in grado di distinguere la loro relazione. Non di meno, la definizione di riservatezza dovrebbe includere anche la possibilità che un utente non voglia poter essere riconosciuto. Vogliamo cioè l’*Anonimato*: un utente è Anonimo se non può essere identificato in un insieme di soggetti anonimi.

Integrità: intesa come capacità di mantenere la veridicità dei dati e delle risorse e garantire che non siano in alcun modo modificati o cancellati, se non ad opera di soggetti autorizzati. Consiste quindi nella prevenzione verso modifiche improvvise o non autorizzate dei dati, al fine di garantire una fiducia sugli stessi. Per assicurare l’integrità è necessario mettere in atto policy di autenticazione chiare e monitorare costantemente l’effettivo accesso ed utilizzo delle risorse, con strumenti in grado di creare log di controllo. Le violazioni all’integrità dei dati possono avvenire a diversi livelli. Ad essa è spesso affiancata l’autenticazione: infatti, oltre a concentrarsi sui cambiamenti del



contenuto, occorre salvaguardarsi anche dalle modifiche che possono cambiare l'origine dei dati. Si può quindi spesso effettuare un attacco sia di autenticazione che di integrità: per esempio, quando in un messaggio viene cambiato il mittente. I possibili meccanismi per garantire l'integrità sono:

- la **prevenzione**: che consiste nel vietare le modifiche a chiunque o ad utenti specificatamente non autorizzati (ciò è però difficile da ottenere);
- la **scoperta**: cioè accorgersi che ci sono state modifiche nell'origine o nel contenuto dei dati e reagire in modo opportuno.

Disponibilità: si riferisce alla possibilità, per i soggetti autorizzati, di poter accedere alle risorse di cui hanno bisogno per un tempo stabilito ed in modo ininterrotto. Rendere un servizio disponibile significa anche garantire che le risorse infrastrutturali siano pronte per la corretta erogazione di quanto richiesto. Devono quindi essere messi in atto meccanismi in grado di mantenere i livelli di servizio definiti. È più difficile effettuare un attacco alla disponibilità della risorsa piuttosto che alla sua affidabilità (reliability). Va anche detto che la disponibilità è più facile da garantire rispetto all'affidabilità. I possibili attacchi in questo campo sono per esempio:

- la manipolazione dell'utilizzo di dati e risorse;
- Denial of Service (DoS).

1.3 AAA

I servizi di **AAA**¹ rappresentano uno strumento fondamentale per la messa in sicurezza dell'infrastruttura di rete. AAA è l'acronimo di **Authentication, Authorization and Accounting** e in generale descrive un protocollo (o una famiglia di protocolli) software che implementa le funzionalità di **Autenticazione, Autorizzazione e Tracciabilità** degli utenti: ad esempio all'interno di una infrastruttura di rete aziendale. Questi elementi sono fondamentali per poter controllare l'accesso alle risorse ed a eventuali dati confidenziali.

Authentication. L'autenticazione (o non ripudio, *non-repudiation*) ha lo scopo di riconoscere l'utente. Normalmente si fa ricorso ad uno *username*, in genere pubblico, e di una *password*, che invece è segreta. Per garantire sicurezza e segretezza, la password deve rispondere a specifiche policy rinnovate periodicamente.

¹AAAAAAAAAAAAAAAAAAAAAAAAAAAAAH



Authorization. L'autorizzazione, seppur integrata con l'autenticazione, è un processo molto differente. Una volta riconosciuto l'utente, bisogna accertarsi che questo possa accedere alle sole risorse che gli competono. Le informazioni necessarie per questo processo sono strettamente legate all'applicativo e alle prestazioni dello stesso: ogni volta che si desidera accedere ad un dato occorre verificare l'autorizzazione dell'utente.

Accounting. L'accounting (tracciabilità) consente di tracciare l'utilizzo delle risorse. È fondamentale per scoprire se un account ha compiuto accessi in orari o da luoghi che risultano insoliti o per verificare azioni sospette, come un accesso da una rete pubblica mentre l'utente risulta, dalla timbratura cartellini, ancora presente in azienda.

1.4 Concetti Correlati

1.4.1 Assurance

Il concetto di “*Garanzia*” non è formalmente legato al sistema su cui si effettua un’analisi di sicurezza informatica. Spesso, infatti, una minaccia alla sicurezza è dovuta dall’utenza stessa del sistema considerato. Pensiamo per esempio ad un impiegato negligente che involontariamente permette ad un virus di infettare il proprio computer, dopo aver inserito una USB. La sicurezza informatica deve tener conto anche di queste situazioni che potrebbero di fatto rendere inutili gli sforzi effettuati dall’organizzazione per evitare che minacce esterne compromettano il proprio sistema informatico. Un comportamento sbagliato da parte del personale infatti è spesso più rischioso rispetto ai tentativi di intrusione da parte di attaccanti esterni all’organizzazione. La natura umana non permette di creare dei protocolli di sicurezza in grado di difendere i propri sistemi da cattive abitudini o da un atteggiamento noncurante degli operatori. Solitamente ogni organizzazione utilizza delle policy aziendali che stabiliscono il comportamento del proprio personale. Una policy non garantisce una protezione totale, ma permette di ridurre il numero di comportamenti sbagliati e di conseguenza rende più difficile gli attacchi interni.

1.4.2 Minacce

Una minaccia (**threat**) è una potenziale violazione della sicurezza. In realtà non è necessario che avvenga concretamente una violazione; per essere considerata una minaccia basta il solo fatto che ci sia la possibilità che il sistema



debba essere protetto. Le azioni che causano una minaccia sono chiamate **attacchi**. Coloro che mettono in pratica queste azioni, o permettono che esse siano eseguite, sono invece chiamati **attaccanti**. Le *minacce* possono essere classificate in:

- *Disclosure*: accesso non autorizzato alle informazioni (violazione della riservatezza);
- *Deception*: accettazione di dati falsi (violazione dell'integrità e dell'autenticazione);
- *Disruption*: corrisponde al DoS (violazione della disponibilità);
- *Usurpation*: controllo non autorizzato di un sistema o parte di esso.

Tra gli *attacchi* più diffusi troviamo:

- **Snooping**: È una minaccia di tipo *Disclosure*. Consiste nell'ascolto o nella lettura passiva (intercettazione) di informazioni. Si parla di *Wiretapping* passivo se l'oggetto di ascolto è la rete stessa. La riservatezza si occupa di evitare che questa minaccia possa essere applicata;
- **Modification**: Consiste nella modifica non autorizzata delle informazioni. Copre tre classi di minacce. L'obiettivo potrebbe essere una *Deception* se l'intenzione è fare in modo che i dati siano accettati in modo alterato. Potrebbe essere una *Disruption* ed una *Usurpation* se la modifica dei dati consente l'esecuzione di azioni non legittime. Il *Wiretapping* attivo se i dati che transitano nella rete sono soggetti ad una modifica non autorizzata. Un esempio di attacco è il “**man-in-the-middle**”, nel quale un intruso legge i messaggi dal mittente e li invia modificati al destinatario senza farsi notare. L'integrità si occupa di evitare che questa minaccia possa essere applicata;
- **Masquerading o Spoofing**: Si tratta di un furto di identità. È un attacco di tipo *Deception* e *Usurpation*. Si basa sul far credere ad una vittima che l'entità con cui sta comunicando è un'altra. L'integrità si occupa di evitare che questa minaccia possa essere applicata;
- **Ripudio dell'origine**: Si tratta di una forma di *Deception* che consiste nel negare l'origine (invio o creazione) di un determinato elemento (ad esempio di un messaggio). L'integrità si occupa di questo attacco;
- **Ripudio della ricezione**: Si tratta di una forma di *Deception* che consiste nel negare di aver ricevuto un determinato elemento. L'integrità si occupa di questo attacco;



- **Denial of Service:** Consiste in una negazione a lungo termine di un servizio. È una forma di *Disruption*. L'attaccante non permette ad un server di fornire un servizio.

1.4.3 Policy e Meccanismi

Le **Policy** di sicurezza identificano le minacce e definiscono quali sono le assunzioni da fare ed i requisiti di cui disporre. Stabiliscono ciò che può o non può essere fatto e definiscono quindi il livello di sicurezza del sistema. Esse si basano su delle assunzioni che consistono nel definire degli insiemi di stati sicuri ed altri insicuri. I **Meccanismi** sono i metodi che ci consentono di individuare, prevenire e ripristinare le minacce. Una volta individuati i rischi in termini di sicurezza ed i loro effetti, si stabiliscono quali contromisure adottare. L'affidabilità di un meccanismo di sicurezza richiede diverse assunzioni:

1. Ogni meccanismo è stato ideato per implementare una o più richieste di una politica di sicurezza;
2. L'unione di tutti i meccanismi copre tutti gli aspetti della policy di sicurezza;
3. Il meccanismo è stato implementato correttamente;
4. Il meccanismo è installato e amministrato correttamente.

Un meccanismo di sicurezza tipico è dato dalla verifica dell'identità prima di cambiare una password. Esistono meccanismi di:

- *Prevenzione:* prevengono attacchi derivati dalla violazione delle politiche di sicurezza;
- *Individuazione:* individuano attacchi di violazione alle politiche di sicurezza;
- *Ripristino:* rappresentano le azioni da eseguire a fronte di un attacco. Le possibili alternative sono:
 - il blocco dell'attacco e la riparazione dei danni causati.
 - non si agisce sull'attacco ma si immagazzinano informazioni sull'attaccante;
 - ritorsione (retaliation) eseguendo un attacco verso l'attaccante.

Capitolo 2

Autenticazione

2.1 Tipi di Autenticazione

L'**autenticazione** è il processo attraverso il quale viene verificata l'identità di un utente che vuole accedere ad un computer o ad una rete. È il sistema che verifica, effettivamente, che un individuo è chi sostiene di essere. Stabilisce l'identità di una parte ad un'altra. Le parti possono essere utenti o computer:

- **computer-computer** (stampa in rete, delega,...)
- **utente-utente** (protocolli di sicurezza, ...)
- **computer-utente** (autenticare un server web,...)
- **utente-computer** (per accedere a un sistema...)

In realtà sono spesso richieste varie combinazioni di queste. L'autenticazione è una proprietà primaria, viene richiesta da un corretto controllo d'accesso. Garantire l'autenticazione significa fare in modo che un sistema sia in grado di associare con certezza un'identità ad una persona.

Esistono vari tipi di autenticazione:

- **Locale**: funziona anche offline ed è la singola macchina che autentica l'utente (desktop systems);
- **Diretta**: l'autenticazione avviene su una macchina diversa da quella dove è avvenuto il collegamento; sul server digitando le proprie credenziali (per aprire file, fare login..);

- **Indiretta:** l'autenticazione non avviene sulla macchina ma su un server di logging remoto (Windows domain, radius, kerberos, nis). Un esempio è l'autenticazione che facciamo in laboratorio. L'utente richiede al server l'accesso ad una risorsa; il server invia la richiesta di accesso al programma gestore degli accessi, il quale ritorna l'esito dell'autenticazione al server. Quest'ultimo infine rigira l'esito all'utente iniziale;
- **Offline:** controllo con chiave pubblica e privata. Grazie al certificato associato alla chiave pubblica, siamo ricondotti al destinatario (PKI...).

2.1.1 Autenticazione utente-computer

L'autenticazione tra un computer ed un utente solitamente avviene tramite l'utilizzo di **certificati** (come nel caso bancario); quella tra l'utente ed un computer avviene sulla base di qualcosa che l'utente:

- *Conosce:* informazioni quali password, PIN, ecc.
- *Possiede:* cose fisiche o elettroniche che solo l'utente ha, come chiavi convenzionali, carte magnetiche o smart, ecc.
- *È:* caratteristiche biometriche quali impronte digitali, l'iride, tono di voce, ecc

2.1.2 Autenticazione su conoscenza

Questo metodo si basa sulla conoscenza di una coppia di elementi, **userID** e **password**, che forniscono una prova dell'identità. Esiste un database locale dove si ricerca la coppia di informazioni: se viene trovata una corrispondenza, l'utente viene riconosciuto. Questo metodo è *antico* ma *estremamente diffuso* per via della sua economicità e *semplicità* di gestione ed implementazione. Risulta però essere debole e poco resistente una volta scoperta una password, in quanto la vulnerabilità agli attacchi sarà molto alta.

2.1.3 Gestione delle password

L'utente fornisce userID e password; questi vengono ricercati nel database e se sono presenti è consentito l'accesso. Questo sistema però presenta alcune debolezze. In primo luogo, nel database userID e password sono in chiaro: poteva capitare che queste informazioni venissero utilizzate per entrare illecitamente. Poteva anche presentarsi il rischio che il database stesso venisse

rubato, è vero che il file era protetto, ma il suo contenuto era in chiaro, per cui non vi era alcuna protezione qualora qualcuno se ne impossessasse. Per questo motivo dal 1960 dal MIT sono state adottate delle strategie per migliorare tale implementazione. Le password, infatti, non venivano più ricercate nel database ma venivano memorizzate in chiaro su un file di sistema in RAM, protetto da politica di sicurezza. In un secondo momento il file veniva letto dal database e veniva effettuato il controllo. Dal 1967, ad opera di Cambridge, è stata introdotta la criptazione delle password al momento del loro inserimento tramite una specifica funzione, per cui venivano memorizzati solo gli hash e non le password stesse. Il vantaggio stava nel fatto che le informazioni nel database non fossero più in chiaro e non vi era più la necessità di utilizzare il file memorizzato in RAM. Allo stesso tempo però il file degli hash poteva essere aperto in lettura da chiunque e per due password uguali si aveva il medesimo hash. Il problema delle password uguali viene risolto con l'aggiunta del *sale* (**salt**), ossia una sequenza di bit generata dal sistema. La password, quindi, risulta diversa dalla sua omonima grazie a questa informazione casuale in più. Nel file viene memorizzato non solo la password, ma anche il sale corrispondente. Questo viene posizionato in una directory nascosta (**shadow**), in nessun modo accessibile da alcun utente ad eccezione di root. Per quanto riguarda l'autenticazione, l'utente inserisce la password che conosce: dal file delle password viene preso il sale dalla riga che corrisponde all'userID e si codificano le due informazioni, *salt + password*. Se il risultato è uguale alla riga memorizzata nel database, allora l'utente può entrare. Anche questo file viene aperto in lettura da tutti. Tale problema verrà risolto solo nel 1987.

2.1.3.1 Vulnerabilità delle password

Di seguito un elenco delle vulnerabilità più comuni a cui sono soggette le password:

- **Guessing**: indovinate. Mai usare le singole parole;
- **Snooping**: sbirciate mentre vengono inserite. Per questo vengono sostituite dagli asterischi (Shoulder surfing);
- **Sniffing**: intercettate durante la trasmissione in rete (Keystroke sniffing);
- **Spoofing**¹: acquisite da terze parti che impersonano l'interfaccia di login (Trojan login);

¹Ricordarsi che il termine corretto è *Spoofing* e non *Spooping*, qui lo SCAT non è bene accetto 



2.1.3.2 Come difendersi ?

Ecco alcune contromisure utili per difendersi e prevenire i principali attacchi alle password:

- *Guessing attack*: utilizzare un **Audit-log** ed introdurre un **limite massimo** per il numero di sbagli;
- *Social engineering*: il cambio password è abilitato solo in specifiche circostanze. Può anche esserci una policy dedicata;
- *Shoulder surfing*: si utilizza la tecnica del **Password Blinding**: vedo gli asterischi al posto della password oppure nessun carattere;
- *Keystroke sniffing*; utilizzo tecniche di **Memory Protection**;
- *Trojan login*: chiave speciale per fare login;
- *Offline dictionary attack*: utilizzo uno **Shadow Password** per evitare che il file delle password venga rubato.

2.1.4 Autenticazione su possesso

In questo caso il **possesso** di un **token** fornisce la prova per il riconoscimento. Chi possiede un oggetto come una carta magnetica, una smart card o smart token, è autorizzato a compiere determinate azioni. Tale tipo di autenticazione solitamente non è nominale, ma può essere accompagnata dal riconoscimento dell'utente. Rubando il token, infatti, non si fa altro che impersonare l'utente. Ogni token memorizza una chiave che deve essere inserita per sbloccarlo. Nel caso del bancomat, per esempio, viene richiesto un PIN per utilizzare la carta. Il vantaggio di questo strumento è che è molto complesso estrarre un'informazione dal token.

Tipi di Token:

- Carte magnetiche, ormai obsolete;
- Smart card per memorizzare una password robusta:
 - Memory card: ha una memoria ma non ha capacità computazionali. È impossibile controllare o codificare il PIN, ma essendo trasmesso in chiaro può essere soggetto a sniffing;
 - Microprocessor card: sono più evoluti. Ha una memoria e un microprocessore, ma può esserci un controllo o la codifica del PIN;

- Smart token:
 - Protetto da PIN;
 - Microprocessor card + tastierino e display
 - Vero e proprio computer;
 - Svantaggi: costoso e fragile
 - Funzionamento: Hanno una Chiave segreta (seme o seed) memorizzata dalla fabbrica, condivisa col server. Preleva le informazioni esterne, per esempio il PIN inserito oppure l'ora, per generare una one-time password. La Password compare poi sul display e viene rinnovata ogni 30-90 secondi. La Sincronizzazione col server avviene grazie al seme ed un algoritmo comune;

2.2 Autenticazione su caratteristiche

Il possesso da parte dell'utente di alcune caratteristiche univoche fornisce prova della sua identità. Le caratteristiche possono essere:

- **Fisiche:** impronte digitali, forma della mano, impronta della retina o del viso, ecc.
- **Comportamentali:** firma in riferimento alle sue caratteristiche quali pressione della penna o inclinazione delle lettere, timbro della voce, velocità nella scrittura, ecc.

Lo schema di funzionamento di tale tecnica prevede una fase iniziale di **campionamento** in cui vengono eseguite più misurazioni sulla caratteristica d'interesse, in modo da stabilire un margine d'errore e viene definito un modello (**template**). Durante la fase di **autenticazione**, viene confrontata la caratteristica appena misurata rispetto al template: c'è successo se i due valori corrispondono a meno di una *tolleranza*, che va definita attentamente. Se è troppo alta consento l'accesso ad altri utenti, se è troppo bassa potrei non consentire l'accesso neanche all'utente a cui magari appartiene la caratteristica misurata. La tolleranza va decisa in base al livello di sicurezza che vogliamo adottare e quindi dall'ambiente di utilizzo. Ad esempio se utilizzo l'impronta digitale per accedere al conto bancario allora il margine di errore dovrà essere minimo, con il rischio di creare dei falsi negativi. Se invece uso l'impronta digitale per fare accedere degli utenti in piscina allora posso permettermi di impostare un margine elevato ed avere a volte dei falsi positivi, così che i clienti non si lamentino. Ottenere una perfetta uguaglianza è



tecnicamente un’operazione impossibile. Il problema fondamentale di questa tecnica sta nell’identificare il giusto template da associare all’individuo in analisi. Va fatto notare che l’autenticazione biometrica è sempre probabilistica, ovvero dipende da come è settato il parametro soglia: se la soglia è **molto bassa** risulterà difficile che l’intruso si autentichi al posto dell’utente, ma il numero di falsi negativi sarà alto; se invece la soglia è **molto alta**, non avrò mai falsi negativi in quanto l’utente reale verrà sempre autenticato ma potrebbero esserci falsi positivi da parte di un attaccante.

Osservazioni: è necessario scegliere un tipo di autenticazione biometrica adatta alla circostanza. Occorre che il sistema non sia troppo invasivo, poiché ciò potrebbe non essere accettato dall’utente. Inoltre, non possono essere utilizzate informazioni che metterebbero a repentaglio la sua privacy. Per esempio, l’autenticazione tramite retina non viene quasi mai impiegata in quanto da essa si potrebbe rinvenire alla presenza di alcune malattie. L’impronta digitale rimane il meccanismo più usato, anche se è il meno pratico ed efficace.

2.3 Kerberos

In un sistema si può aggiungere una procedura di autenticazione per ogni servizio, ad esempio una password per accedere al sistema, una per accedere al file system, ecc. Questa tecnica risulta essere molto scomoda, soprattutto se si utilizzano i token. Un’alternativa consiste nell’utilizzare un’unica credenziale di autenticazione per accedere a tutti i servizi; avere un’unica password è una soluzione comoda ma poco robusta. Kerberos è un **protocollo** reale che si occupa di questo problema (e non solo), ha come obiettivo quello di garantire la segretezza, l’autenticità (ad accesso singolo), la temporalità (le chiavi usate hanno validità limitata per prevenire *replay attack*). Quest’ultima caratteristica viene gestita tramite i **timestamp**. In pratica, l’accesso ad un servizio avviene tramite “biglietto” che vale per un periodo limitato; anche se tale biglietto venisse intercettato, può essere usato solo una volta e per un breve tempo.

Kerberos è un protocollo di autenticazione dei servizi di rete creato dal MIT che si serve della crittografia a chiave simmetrica per autenticare gli utenti per i servizi di rete, eliminando così la necessità di inviare le password attraverso la rete. Ricorre ad un’unica credenziale di autenticazione per accedere a tutti i servizi. L’autenticazione mediante Kerberos impedisce agli utenti non autorizzati di intercettare le password inviate attraverso la rete.

Il nome Kerberos deriva da Cerbero, il cane a tre teste che sorveglia le porte dell'Ade.

La maggior parte dei sistemi di rete convenzionali usa uno schema di autenticazione basato sulle password. Quando un utente effettua una autenticazione per accedere ad un server di rete deve fornire le credenziali. Sfortunatamente, la trasmissione delle informazioni di autenticazione spesso non è criptata. Per essere sicuri, la rete non deve essere accessibile dall'esterno e tutti i computer e gli utente sulla rete devono essere fidati. Anche una volta che una rete è collegata a Internet, non si potrà più assumere che la rete sia sicura, in quanto qualunque aggressore che ha accesso alla rete può intercettare le password e i nomi utente che la attraversano. Lo scopo principale di Kerberos è quello di eliminare la trasmissione delle informazioni di autenticazione attraverso la rete. Il suo corretto utilizzo permette di ridurre drasticamente la possibilità di intercettazione. Per funzionare, Kerberos deve avere possedere due requisiti:

- un **timeserver** in ogni sistema dove viene utilizzato. I timestamp permettono di sincronizzare le macchine;
- ogni utente deve avere una **chiave pubblica** e una **privata** (una password a lungo termine).

2.3.1 Come funziona ?

L'utente, una volta entrato nella propria macchina, vuole accedere ai servizi messi a disposizione nella rete. Supponiamo voglia raggiungere il server in basso. Effettuato il primo accesso alla macchina, le credenziali dell'utente vengono inviate ad uno speciale servizio centrale, il Kerberos, che ricorre essenzialmente a due tipi di server:

- *Authentication Server (AS)*², che ha lo scopo di autenticare l'user;
- *Ticket-Granting Server (TGS)*, che a seconda dell'user, gli assegna i diritti per compiere determinate azioni;

²Ricordarsi che AS non sta per *Autonomous System* !



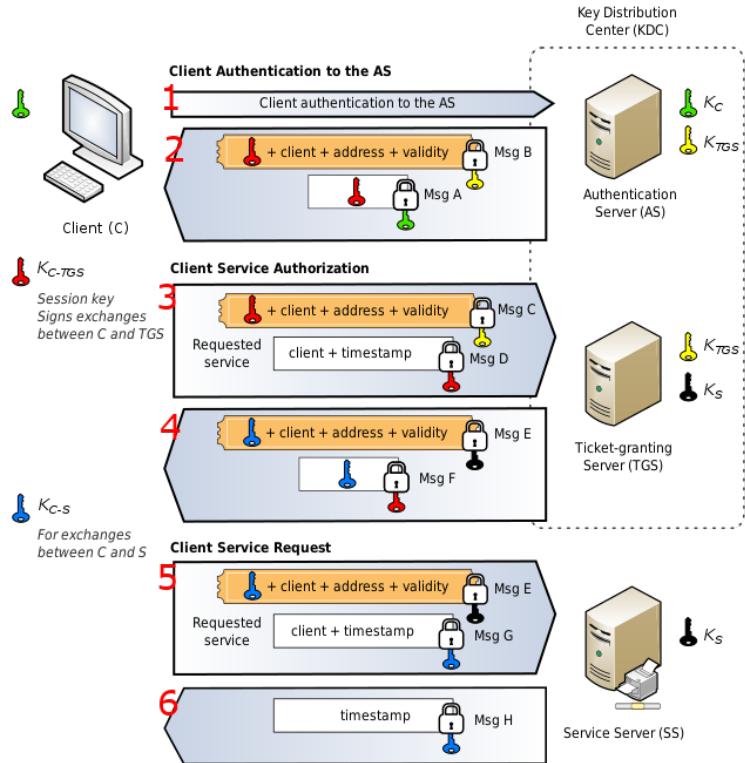


Figura 2.1: Schema di funzionamento del protocollo Kerberos.

1. L'utente si connette con le proprie credenziali alla workstation e queste vengono passate al Kerberos;
2. AS verifica il diritto di accesso dell'utente, ricercando una corrispondenza nel database. Se il confronto ha esito positivo, AS assegna all'user un ticket e una session key. Il ticket verrà poi passato al TGS per richiedere i servizi, mentre la chiave permette di codificare le sue richieste, sempre poste al TGS. Entrambe le informazioni vengono criptate;
3. L'utente a questo punto vuole utilizzare un certo servizio. La workstation richiede all'utente la password e la utilizza per decrittografare il messaggio in arrivo, quindi invia ticket e autenticatore (che contiene il nome, la rete, l'indirizzo e l'ora dell'utente) a TGS;
4. TGS decripta il ticket e l'autenticatore, verifica se esso ha il diritto o meno ad utilizzare una certa risorsa e crea il ticket specifico per il servizio richiesto. La comunicazione tra utente e TGS avviene solo grazie alla session key che la codifica;

5. La workstation manda ticket e autenticatore al server;
6. Il server verifica il match tra ticket ed autenticatore e permette l'accesso al servizio. Se viene richiesta una mutua autenticazione, il server ritorna un autenticatore.

È importante notare che l'utente può utilizzare il ticket finché esso risulta ancora valido, il che può accadere anche per più tentativi di richiesta al servizio. Per un servizio diverso mai utilizzato, deve ripercorrere il medesimo processo.

Kerberos opera fondamentalmente in **3 fasi** e ognuna di queste fornisce le credenziali per la successiva:

I. AUTENTICAZIONE

1. $A \rightarrow AS : A, TGS, T1$
2. $AS \rightarrow A : \{authK, TGS, Ta, \{A, TGS, authK, Ta\}K_{tgs}\}K_a$

II. AUTORIZZAZIONE

3. $A \rightarrow TGS : \{A, TGS, authK, Ta\}K_{tgs}, \{A, T2\}_{authK}, B$
4. $TGS \rightarrow A : \{servK, B, Ts, \{A, B, servK, Ts\}K_b\}_{authK}$

III. SERVIZIO

5. $A \rightarrow B : \{A, B, servK, Ts\}K_b, \{A, T3\}_{servK}$
6. $B \rightarrow A : \{T3+1\}servK$

1. All'autenticazione si ottiene **authK** e **authTicket**. Il primo per autenticarsi e rendere riservata la trasmissione con il TGS, il secondo per ottenere il servizio del TGS;
2. Nell'autorizzazione da parte del TGS, si ottiene **servK** e **servTicket**, da presentare per il server richiesto;
3. Il servizio conferma poi la richiesta.

authK e **servK** sono chiavi simmetriche, in quanto possono essere usate sia per criptare che per decriptare. Ogni chiave di sessione ha una propria durata. Chiaramente quella della **servK** è minore rispetto a quella della **authK**.

2.3.2 Nel dettaglio

A è l'utente che richiede il servizio *B*. *A* si autentica all'AS ad un certo tempo *T1* (*messaggio 1*). Se l'utente non è registrato tra quelli del dominio,



non viene preso in considerazione. Se invece è autorizzato, si verifica grazie al ticket a quale TGS vuole rivolgersi. A per autenticarsi ha inviato solo il nome dell’utente, però l’AS fa una challenge: invia ad A un’informazione e se l’utente è effettivamente chi dice di essere, allora saprà decifrarla con la sua chiave. A apre il pacchetto con la chiave privata (*messaggio 2*). All’interno trova:

- un `authTicket` codificato con la chiave di TGS e quindi non può aprirlo;
- la conferma del fatto che il TGS è disponibile a ricevere le sue richieste;
- il tempo T_a in cui AS ha inviato la risposta;
- `authK`, cioè una chiave di sessione che vale per un certo periodo di tempo (stabilito dalla configurazione) a partire da T_a ;

All’interno del *messaggio 3* abbiamo:

- ticket ricevuto dall’AS, anche se non sa cos’è;
- il suo nome e T_2 che corrisponde all’istante in cui viene inviato il pacchetto. Il tutto è codificato da `authK`;
- nome del servizio a cui vuole accedere, cioè B ;

Il fatto di inviare A e `authK` permette di autenticarsi con il TGS. `authK` è stata inviata da AS e poiché TGS si fida di AS, riesco ad aprire il messaggio $\{A, T_2\}$ dell’autenticatore 1. Il messaggio è chiaramente scritto da A e la conferma arriva dal fatto che A è scritto anche fuori in `authTicket` e l’`authK` è stato condiviso da AS con A . A questo punto A è autenticato. Se il *messaggio 3* fosse stato inviato al TGS sbagliato, al richiesta non verrebbe presa in considerazione. T_a permette di capire se `authK` è ancora valida. Il TGS verifica se A ha il diritto di utilizzare B . Se così è, viene autorizzato con l’invio del *messaggio 4*:

- È criptato con `authK` e solo A può aprirlo. Si usa `authK` perché altrimenti si potrebbe usare il ticket al posto di A ;
- grazie ad `authK` siamo sicuri che il messaggio viene aperto da TGS ed A perché la chiave è condivisa (ci sono sia confidenzialità che autenticazione);



- TGS verifica che ci sia scritto B , altrimenti il messaggio viene eliminato. A è autorizzato ad utilizzare B a partire dal momento T_s (momento in cui TGS invia il messaggio ad A);
- la chiave di sessione servK ha una certa durata a partire da T_s . È la session key da utilizzare con il servizio;

Il *messaggio 5* che va da A a B invece:

- prende il ticket servTicket che gli ha dato il server precedente;
- A per autenticarsi con B scrive $\{\text{A}, T_3\}$ codificato con servK . Viene decodificato autenticatore 2 e si ha la certezza che A ha inviato il messaggio. Se T_3 è troppo distante da T_s , il messaggio perde di validità. Anche in questo caso A , B e TGS conoscono la durata delle chiavi di sessione;
- K_b è una chiave condivisa tra B , TGS e AS

Il *messaggio 6* viene inviato come risposta al fatto che la richiesta è stata accettata:

- $\{T_3+1\}$ è criptato con servK , questo per evitare dei replay attack;

È doveroso notare che non sono solo le chiavi ad essere simmetriche, anche lo stesso protocollo lo è. Questa versione di Kerberos è la 4, che attualmente non viene utilizzata.

2.3.3 Gestione delle Chiavi

AS genera authK al tempo T_a , TGS genera servK al tempo T_s . Validità:

- di authK (ossia di T_a) in ore, diciamo L_a
- di servK (ossia di T_s) in minuti, diciamo L_s
- di un autenticatore (ossia di T_1 , T_2 e T_3) in secondi.

TGS può generare servK solo qualora sia $T_s + L_s \leq T_a + L_a$, altrimenti si verifica il problema di **cascata dell'attacco**. Si hanno quindi attacchi consequenziali: un attacco ne provoca altri direttamente. Supponiamo che C abbia violato una chiave di sessione (di autorizzazione) scaduta authK che B aveva condiviso con A . Con una semplice decodifica C ottiene servK ancora valida se non si impone $T_s + L_s \leq T_a + L_a$. C può accedere a B per la durata residua di L_s .



Autenticazione tra domini: fino ad ora abbiamo parlato di Kerberos in relazione ad un unico dominio, ma le versioni successive alla 4 ne ammettono anche di più. Supponiamo ci siano due aziende connesse in extranet, ovvero due intranet connesse tramite VPN. Si potrebbero avere due server Kerberos e far sì che i servizi delle due sedi accettino utenti di domini diversi.

2.4 Intruder

Abbiamo tre classi di intruders (intrusi):

- **Masquerader:** Individuo che finge di essere qualcun altro per ottenere un accesso non autorizzato a informazioni o servizi.
- **Misfeasor³:** Un utente legittimo che accede a dati, programmi o risorse per i quali tale accesso non è autorizzato o che è autorizzato a tale accesso ma abusa dei suoi privilegi per eseguire azioni non autorizzate. Un esempio può essere il dipendente con accesso a dati aziendali sensibili che utilizza i propri privilegi per rubare o divulgare i dati;
- **Clandestine User:** Una persona che prende il controllo di supervisione per eludere l'auditing e i controlli di accesso o sopprimere la raccolta di audit;

Intrusion Detection System: L’Intrusion Detection System o **IDS** è un dispositivo software o hardware utilizzato per identificare accessi non autorizzati ai computer o alle reti locali. Lo scopo generale di un IDS è informare gli amministratori di sistema che potrebbe esserci un’intrusione nel sistema. Lo fa andando a lanciare degli allarmi che poi gli amministratori di sistema dovranno andare a controllare manualmente. Gli avvisi includono generalmente informazioni sull’indirizzo di origine dell’intrusione, l’indirizzo di destinazione/vittima e il tipo di attacco “sospetto”.

Un IDS può avere un’installazione o di tipo **HOST based** o di tipo **NETWORK based**. Nella *Host based* l’IDS è installato nelle singole macchine utente ed ha quindi una vista ristretta alla singola macchina e non all’intero sistema. In questo caso lavora più a *livello applicazione* per rilevare le intrusioni. Nel *Network based* invece l’IDS è installato a livello di rete e quindi ha una visione più ampia e generale del sistema. In questo caso lavora appunto a livello di rete e quindi riesce a rilevare se un nuovo dispositivo si collega alla rete o se un dispositivo invia strane richieste.

³Ricordarsi che non è il nome di un pokémon !



Un IDS può individuare una intrusione tramite 3 tipologie di rilevamento differenti:

- **Threshold based detection:** si conta il numero di occorrenze di un determinato evento e se tale numero supera una determinata soglia allora viene considerato come “attacco in corso”. E’ una tecnica di rilevamento basata su anomalie (*anomaly detection*).
- **Profile based detection:** si traccia ciò che un account fa abitualmente e si guarda se in una giornata sta assumendo il solito comportamento o se si discosta molto dalle abitudini. Se si discosta troppo allora si considera come “attacco in corso”. E’ una tecnica di rilevamento basata su anomalie (*anomaly detection*).
- **Signature based detection:** si usano modelli di attacchi già noti e si mette a confronto il comportamento rilevato con i modelli conosciuti per vedere se corrispondono a una minaccia nota.

È importante sapere che un IDS non può bloccare o filtrare i pacchetti in ingresso ed in uscita, né tanto meno può modificarli. Un IDS può essere paragonato ad un antifurto mentre il firewall alla porta blindata. L’IDS non cerca di bloccare le eventuali intrusioni, cosa che spetta al firewall, ma cerca di rilevarle laddove si verifichino. Per ogni rete è necessario un IDS che agisce solo sulla stessa.

Intrusion Prevention System : L’Intrusion Prevention System o **IPS** è un dispositivo software o hardware che ha le stesse funzionalità di un IDS ma in più può fermare un attacco in **real time** andando a compiere alcune azioni quali comunicare con un firewall e aggiungere delle regole per bloccare determinate connessioni malevoli o mandare down l’intero network (solo nei casi peggiori) così da evitare ulteriori danni al sistema.

Honeypot 🍷: Un honeypot rappresenta una strategica misura di sicurezza con la quale gli amministratori di un server ingannano gli hacker e gli impediscono di colpire. Un “barattolo di miele” simula i servizi di rete o programmi per attirare i malintenzionati e proteggere il sistema da eventuali attacchi. In pratica gli utenti configurano gli honeypot, utilizzando delle tecnologie lato server e lato client. Solitamente la trappola consiste in un computer o un sito che sembra essere parte della rete e contenere informazioni preziose, ma che in realtà è ben isolato e non ha contenuti sensibili o critici; potrebbe anche essere un file, un record, o un indirizzo IP non utilizzato. Il valore primario di un honeypot è l’informazione che esso dà sulla



natura e la frequenza di eventuali attacchi subiti dalla rete. Gli honeypot possono portare dei rischi ad una rete e devono essere maneggiati con cura. Se non sono ben protetti, un attaccante potrebbe usarli per entrare in altri sistemi.

Gli honeypot possono essere di due tipi:

- **Bassa Interazione:** sono macchine facili da gestire. Emulano perfettamente i servizi, ma l'intruder non riesce a prendere completamente il controllo in quanto la macchina è vuota;
- **Alta Interazione:** macchine dove vi è effettivamente un servizio e l'utente può utilizzarlo. Sono la tipologia migliore perché più realistiche ma sono più complesse da gestire;

Capitolo 3

Crittografia

3.1 Introduzione

La crittografia è una scienza antichissima che consiste nel codificare e decodificare l'informazione. L'operazione di codifica permette di ottenere un testo codificato a partire da un “testo in chiaro” che può essere letto da tutti. L'operazione di decodifica invece, consiste nel ricavare un testo in chiaro partendo da un testo cifrato. Ambedue le operazioni si basano su un algoritmo e sulla chiave; l'algoritmo è certamente pubblico. La sicurezza del sistema è data dalla segretezza della chiave e dalla robustezza dell'algoritmo. Esistono due tecniche di crittografia: la crittografia *simmetrica* e la crittografia *asimmetrica*, quest'ultima più recente.

3.2 Crittografia a chiave Privata (simmetrica)



Figura 3.1: Esempio del funzionamento della Crittografia Simmetrica.

La codifica e la decodifica sono eseguite dagli algoritmi crittografici assieme ad una chiave che è la stessa per entrambi i procedimenti. Tale chiave pertanto dovrà essere condivisa tra le parti della comunicazione. La segretezza,

l'autenticazione e l'integrità dipendono dalla segretezza della chiave. Un sistema crittografico a chiave simmetrica molto conosciuto è il **DES** ideato nel 1976 dalla IBM, tuttora usato per cifrare files nei personal computer. Questo sistema utilizza una chiave di 56 bit (256 possibili chiavi).

3.3 Crittografia a chiave Pubblica (asimmetrica)

La crittografia asimmetrica, conosciuta anche come crittografia a coppia di chiavi, crittografia a chiave pubblica/privata o anche solo crittografia a chiave pubblica, è un tipo di crittografia dove ad ogni attore coinvolto nella comunicazione è associata una coppia di chiavi:

- La chiave pubblica, che deve essere distribuita;
- La chiave privata, appunto personale e segreta;

Fra due interlocutori, non vi è dunque la necessità di scambiarsi le chiavi. Se con una delle due chiavi si cifra (codifica) un messaggio, allora quest'ultimo sarà decifrato solo con l'altra.

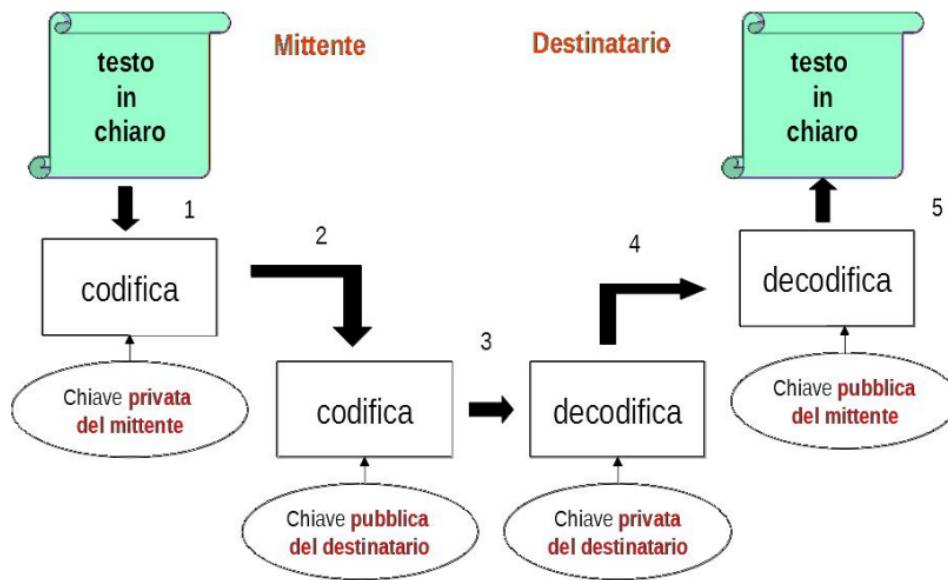


Figura 3.2: Esempio del funzionamento della Crittografia Asimmetrica.

In Figura 3.2 va fatto notare come:

- il mittente cifra il messaggio con la sua chiave privata così che tutti possano decifrarlo con la sua chiave pubblica. La cifratura in questo caso ha il solo scopo di garantire l'**integrità** del messaggio.
- il mittente cifra il messaggio con la chiave pubblica del destinatario così che solo il destinatario, con la propria chiave privata, potrà decifrarlo, garantendo la **confidenzialità** del messaggio.

Gli attacchi a sistemi di crittografici sono detti attacchi di **Crittoanalisi** e come obiettivo hanno quello di provare a dedurre la chiave che gli permetterebbe di decriptare il testo. I principali attacchi crittografici si basano sulle seguenti caratteristiche:

- **CypherText only**: è noto solo il testo codificato;
- **Known plaintext**: il messaggio è cifrato ma il testo in chiaro è noto;
- **Chosen plaintext**: l'attaccante può cifrare messaggi in chiaro da lui scelti per ottenere il ciphertext ed eseguirne l'analisi;
- **Brute-force**: forza bruta, tentativi di attacco alla chiave finché non si trova quella giusta;

Crittografia Perfetta: la crittografia si dice “Perfetta” quando nessun testo codificato rilascia informazione alcuna né sulla chiave usata per la codifica, né sul testo in chiaro, il quale può essere recuperato se e solo se la chiave è disponibile. Si tratta di una situazione ideale, in quanto ogni tipo di crittoanalisi sarebbe reso inutile e la probabilità di ricavare informazioni supplementari da un testo codificato sarebbe piuttosto nulla. Purtroppo però, la crittografia in pratica non è quasi mai perfetta.

3.4 Funzioni Hash

Una funzione hash è una funzione matematica che permette di ridurre una qualunque stringa di testo (indipendentemente dalla sua lunghezza) in una nuova stringa avente precise caratteristiche tra cui un numero di caratteri predefinito. A partire da un input X, in altre parole, sarà possibile generare un output Y che avrà delle caratteristiche ben definite. A prescindere dal tipo di funzione di hashing tutte hanno dei punti in comune:

- **Costanza:** a parità di input la stessa funzione di hash restituirà sempre la stessa stringa alfanumerica; ad ogni input, in altre parole, corrisponde sempre e inevitabilmente lo stesso output;
- **Irreversibilità:** mentre è sempre possibile riprodurre l'output conoscendo l'input originario col quale l'output è stato generato, non è però possibile fare il percorso inverso. Non si può quindi, a partire da una stringa alfanumerica (output), risalire al contenuto iniziale dell'input;
- **Determinismo:** indipendentemente da quanto sia lungo l'input, la funzione di hash restituirà sempre una stringa alfanumerica di un numero determinato di caratteri;
- **Effetto valanga:** non importa quanto l'input sia lungo e complesso, è sufficiente una variazione infinitesimale dell'input per generare un output completamente differente;

Alcune funzioni hash (come MD5 e SHA1) sono notoriamente conosciute a causa della loro vulnerabilità ai “Rainbow Table Attack” , ovvero attacchi in cui sono stati creati dizionari che riportano la corrispondenza tra una parola e relativo hash. Così facendo un attaccante può comodamente usare uno dei molteplici siti web¹ o scaricare direttamente un dizionario² per verificare se l'hash di una password che ha trovato è stato già craccato, senza doversi preoccupare di dover effettuare un *bruteforce* sull'hash.

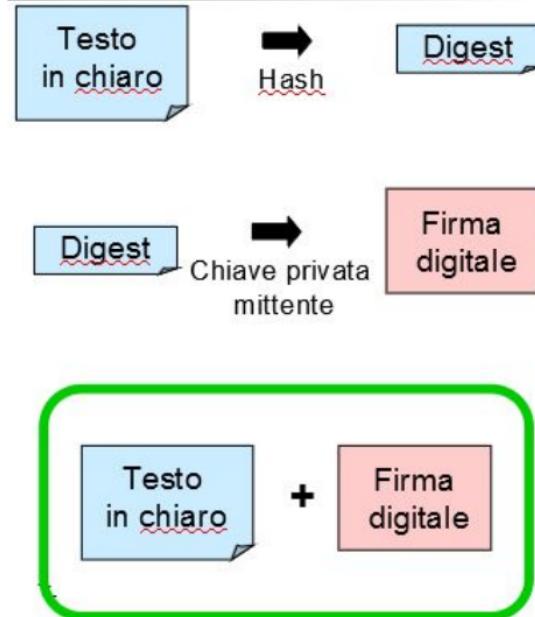
3.5 Firma Digitale

La firma digitale è una tecnologia con cui possono essere effettivamente soddisfatti tutti i requisiti richiesti per dare validità legale ad un documento elettronico firmato digitalmente; garantisce i servizi di integrità, autenticazione e non ripudio. Firmare non è esattamente codificare: firmare digitalmente un documento è spesso utile perché evita di codificare l'intero file in quanto ciò può richiedere un tempo elevato. Verificare una firma, quindi, non significa decodificare.

¹Uno dei più famosi è sicuramente CrackStation.

²Esistono rainbow table da svariati Tb scaricabili gratuitamente su Free Rainbow Tables.





3.5.1 Creazione della Firma

La firma digitale viene realizzata tramite tecniche crittografiche a chiave pubblica insieme all'utilizzo di particolari funzioni matematiche, chiamate funzioni hash unidirezionali. Il processo di firma digitale passa attraverso tre fasi:

1. Generazione dell'impronta digitale.
2. Generazione della firma.
3. Apposizione della firma.

Nella prima fase viene applicata al documento in chiaro una funzione di hash appositamente studiata che produce una stringa binaria di lunghezza costante e piccola, normalmente 128 o 160 bit, chiamata “digest message”, ossia impronta digitale. Poiché la dimensione del digest message è fissa, e molto più piccola di quella del messaggio originale, la generazione della firma risulta estremamente rapida. Utilizzare le funzioni hash consente di evitare che per la generazione della firma sia necessario applicare l'algoritmo di cifratura all'intero testo che può essere molto lungo. Mediante un software adatto si genera una coppia di chiavi da utilizzare: una che verrà mantenuta segreta per l'apposizione della firma; l'altra, destinata alla verifica, che verrà resa

pubblica. Quindi la seconda fase, la generazione della firma, consiste semplicemente nella cifratura con la propria chiave privata dell'impronta digitale generata in precedenza. Nell'ultima fase, la firma digitale generata precedentemente viene aggiunta in una posizione predefinita, normalmente alla fine del testo del documento. E' da tenere presente che l'apposizione della firma digitale non garantisce la confidenzialità del testo, perché questo viene inviato in chiaro. Serve solo a garantirne integrità e autenticità:

- **Autenticità:** Il messaggio arriva proprio da chi dice di essere il mittente;
- **Integrità:** Il messaggio non ha subito modifiche o manomissioni;

3.5.2 Verifica della Firma

Il destinatario ottiene testo in chiaro con apposta la firma digitale. Per comodità viene inviata anche la chiave pubblica del mittente. A questo punto gli step sono:

1. Separare il testo dalla firma;
2. Decodificare la firma con la chiave pubblica del mittente;
3. Calcolare il digest del testo tramite la funzione di hash;
4. Verificare che i due digest coincidano. In caso positivo si ha la conferma del fatto che il documento è integro e non è stato sottoposto a modifiche.

3.6 Certificato Digitale

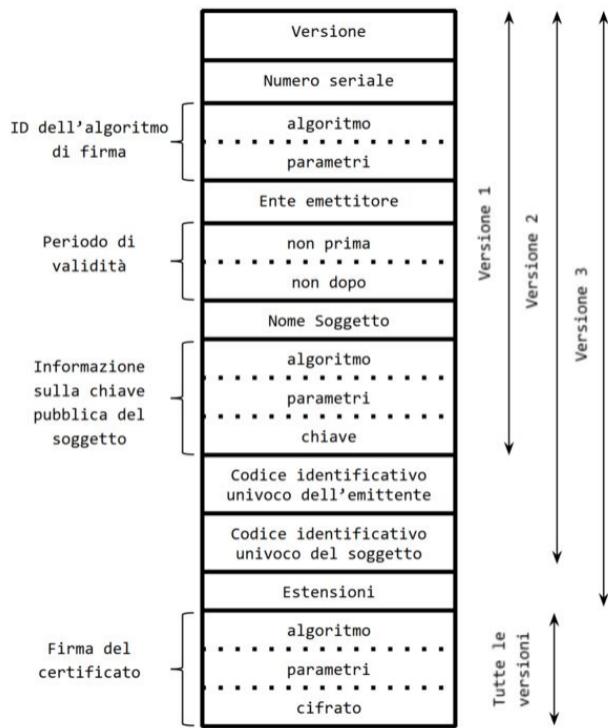
Nella crittografia asimmetrica un **certificato digitale** è un documento elettronico che attesta l'associazione univoca tra una chiave pubblica e l'identità di un soggetto. Il certificato digitale contiene informazioni sulla chiave, sull'identità del proprietario (denominato soggetto) e sulla firma digitale di un'entità che ha verificato i contenuti del certificato (denominato emittente) e riconosciuta come **CA** (*Certification Authority*). Tale certificato, infatti, è a sua volta autenticato per evitarne la falsificazione, sempre attraverso firma digitale, ovvero viene cifrato con la chiave privata dell'associazione, la quale fornisce poi la rispettiva chiave pubblica per verificarlo. Ad una CA sono assegnati 10 compiti:

1. Identificare con certezza la persona che fa richiesta della certificazione della chiave pubblica;

2. Rilasciare e rendere pubblico il certificato;
3. Garantire l'accesso telematico al registro delle chiavi pubbliche;
4. Informare i richiedenti sulla procedura di certificazione;
5. Dichiарare la propria politica di sicurezza;
6. Attenersi alle norme sul trattamento di dati personali;
7. Non rendersi depositario delle chiavi private;
8. Procedere alla revoca o alla sospensione dei certificati in caso di richiesta dell'interessato o venendo a conoscenza di abusi o falsificazioni, ecc;
9. Rendere pubblica la revoca o la sospensione delle chiavi;
10. Assicurare la corretta manutenzione del sistema di certificazione.

3.6.1 Formato x.509

Il formato più comune per i certificati di chiave pubblica è definito da X.509, raccomandato dalla ITU (International Telecommunication Union). Il certificato viene firmato da chi lo emette. “Codice identificativo dell'emittente” è univoco per ogni CA esistente. La firma garantisce il fatto che il documento è autentico, cioè che è stato scritto dal “nome soggetto” e che è stato verificato da “ente emittitore”. E’ importante specificare il “periodo di validità” poiché la stessa firma (e la coppia chiave pubblica-privata) ha una validità temporale. Se questa dovesse scadere, il certificato va riemesso. Ciò viene fatto anche nel momento in cui i ruoli (specificati nel campo “estensioni”) di chi partecipa alla sottoscrizione cambiano. Viene anche specificato l’elenco degli algoritmi utilizzati per l’apposizione della firma con i relativi parametri.



3.6.2 Ottenerne un certificato Digitale

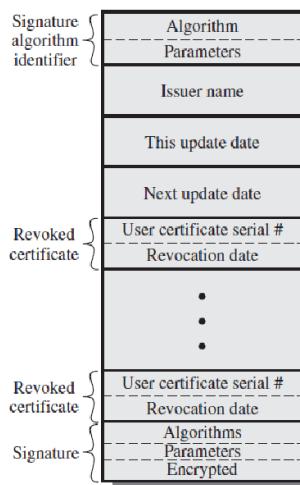
Sono richieste le seguenti fasi: l'utente genera sul proprio PC una coppia di chiavi. I browser comuni offrono tale il servizio; la chiave privata è memorizzata localmente in un file nascosto. Per avere una maggiore sicurezza si potrebbero memorizzare le chiavi su una SmartCard protetta da un PIN. L'utente invia alla CA una richiesta di certificato, insieme alla chiave pubblica generata; la CA autentica il richiedente, di solito chiedendogli di recarsi di persona ad uno sportello di **LVP** (*Local Validation Point*) collegato con la CA. A questo punto, verificata l'identità, la CA emette il certificato, lo invia al richiedente tramite posta elettronica ed inserisce la chiave certificata nel registro delle chiavi pubbliche. L'intera procedura sopra descritta accade nell'ambito di una **PKI** (*Public Key Infrastructure*), la cui struttura minima prevede la presenza di una CA ed un LVP; in generale sono ammessi più LVP. Una PKI può avere una struttura gerarchica in cui alcune CA certificano altre CA, ottenendo una “catena di fiducia”. Secondo tale struttura: la “Root CA” certifica le CA di primo livello; le CA di primo livello certificano le CA di secondo livello; le CA di ultimo livello certificano il singolo utente.

3.6.3 Revoca del certificato

Il certificato può essere revocato per varie ragioni: cambio dei dati personali (email, recapito, ecc), licenziamento, dimissioni, Compromissione della chiave privata, ecc. Può avvenire anche su richiesta da parte dell'utente o dell'emittitore.

3.6.4 Certificate Revocation List

Un **CRL** è un elenco di certificati digitali che sono stati revocati dalla CA, prima della data di scadenza pianificata e non dovrebbero più essere considerati attendibili. Viene generato e pubblicato periodicamente, spesso ad un intervallo di tempo definito. Ad ogni pubblicazione, infatti, viene anche comunicata la data del successivo aggiornamento. Viene rilasciato da un'entità autorizzata che è tipicamente la CA che ha anche rilasciato i certificati corrispondenti.



3.7 Protocolli di Sicurezza

Un protocollo di sicurezza è una sequenza di azioni (passi) che coinvolge due o più parti, finalizzata all'instaurazione di una comunicazione sicura tra di esse, al sicuro dalle azioni di un possibile intruso. L'insieme dei passi specificati costituisce la sessione del protocollo. Servono per garantire l'autenticità dell'utente.

Nonce: In crittografia il termine nonce indica un numero, generalmente casuale o pseudo-casuale, che ha un utilizzo unico. Nonce deriva infatti dal-



l'espressione inglese “for the nonce”, che significa appunto “per l'occasione”. Un nonce viene utilizzato spesso nei protocolli di autenticazione per assicurare che i dati scambiati nelle vecchie comunicazioni non possano essere utilizzati in attacchi di tipo replay attack.

La spia DY: In tutti i protocolli di autenticazione che analizzeremo di seguito, viene identificata la possibilità che vi sia una “spia” (intruder) all'interno del sistema che può effettuare una serie di operazioni, quali:

- Intercettare messaggi e prevenirne il recapito;
- Far rimbalzare a piacere i messaggi intercettati;
- Imparare i testi in chiaro e i testi codificati;
- Tentare di decriptare con tutte le chiavi note;
- Utilizzare le proprie credenziali legali;
- Ottenere certe credenziali illegalmente;
- Creare messaggi fasulli da componenti già note;

3.8 Autenticazione di utenti remoti con Needham-Schröder (1978)

Sono stati definiti vari protocolli a riguardo. L'obiettivo è quello di garantire l'autenticazione dell'utente da remoto appunto, ottenuta dallo scambio segreto dei nonce. Needham-Schröder è basato sulla crittografia asimmetrica e permette di assicurare la mutua autenticazione tra due entità di rete. Nella sua forma proposta non è sicuro. Utilizzeremo dei messaggi fatti così:

- Nomi di utenti: A, B, C, \dots
- Chiavi crittografiche
 - a lungo termine: K_a, K_b, \dots
 - a breve termine: K_{ab}, \dots (chiavi di sessione)
- Nonce: N_a, N_b, \dots
- Timestamp: T_a, T_b, \dots



- Digest
- Label: “trasferisci denaro”, “collegati alla porta xy”, ...

I messaggi base possono essere a loro volta composti:

- Concatenati: m, m', \dots
- Criptati: $m_K, \{m, m'\}_K, \dots$

Il protocollo presuppone una PKI (infrastruttura a chiave pubblica) con crittografia perfetta.

1. **Alice → Bob : $\{Alice, N_a\}K_{bob}$**
2. **Bob → Alice : $\{N_a, N_b\}K_{alice}$**
3. **Alice → Bob : $\{N_b\}K_{bob}$**

Alice vuole identificarsi con Bob, quindi Bob alla fine dell'applicazione del protocollo dovrà essere sicuro di aver comunicato con Alice.

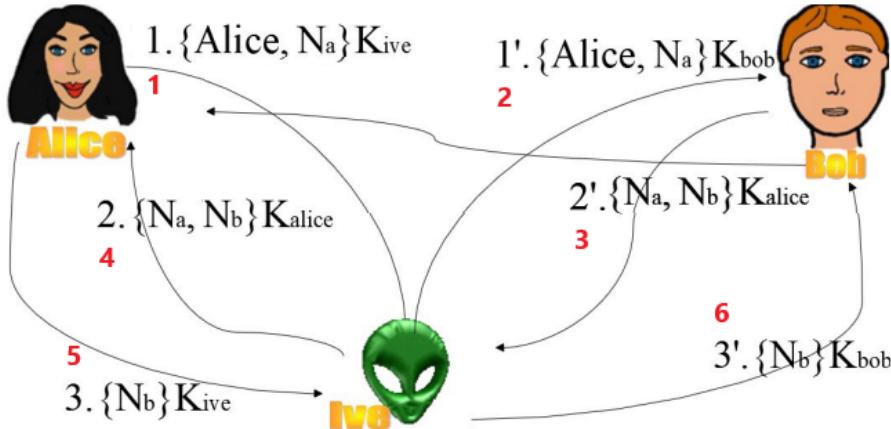
1. Alice invia un messaggio a Bob, criptato con la chiave pubblica di quest'ultimo e che quindi solo lui potrà aprire, in cui inserisce chi lo sta contattando ed un nonce. Il nonce N_a permette di capire se il messaggio che è stato inviato è nuovo. Si analizzano tutti i nonce già utilizzati per verificare che esso sia appunto un valore mai visto;
2. Bob riceve il messaggio e lo apre. Una volta letto, risponde ad Alice: invia indietro il nonce N_a ricevuto, più ne aggiunge uno nuovo, N_b , questa volta per fare in modo che sia lui ad autenticarsi con Alice. Il messaggio è codificato con la chiave pubblica di Alice, K_{alice} ;
3. Alice riceve la risposta e invia a sua volta N_b a Bob. A questo punto Bob è sicuro di poter parlare con Alice.

L'autenticazione è garantita dal fatto che i nonce non possono necessariamente avere lo stesso numero e che le chiavi a lungo termine non possono essere compromesse. La prova del fatto che il messaggio 1 sia stato inviato da Alice ci viene data dalla chiave K_{alice} nel messaggio 2 e quindi sappiamo che solo lei lo può aprire.



3.8.1 Attacco di Lowe (1995)

Uno studente di Cambridge, Lowe, dimostra che il protocollo appena visto non funziona perché è possibile attaccarlo. Ciò significa che qualcun altro potrebbe riuscire a spacciarsi per Alice con Bob (in riferimento all'esempio precedente).



Nel Dettaglio: Ive è uno dei partecipanti al protocollo:

1. Alice vuole comunicare con lei. Ive (che però è l'intruder) apre il messaggio inviato da Alice (grazie alla sua chiave privata) e lo invia a Bob;
- 1'. Ive, che vuole spacciarsi per Alice nei confronti di Bob, ricodifica il messaggio prima di inoltrarlo a Bob;
- 2'. Bob segue le regole del protocollo N-S ed invia ad Alice $\{N_a, N_b\}$ criptato con K_{alice} . Ive potrebbe lasciare andare il messaggio oppure
2. potrebbe intercettarlo e rispedirlo ad Alice.

Tutti i partecipanti, nel momento in cui ricevono dei messaggi, devono verificare che gli invii corrispondano a quanto specificato nel protocollo. Se così non fosse, tutta l'esecuzione cadrebbe e verrebbe riavviata da zero.

2. Il messaggio inviato da Ive ad Alice ha come primo elemento N_a (il nonce di Alice), e come secondo elemento N_b (il nonce di Bob) che Alice crede sia stato generato da Ive;
3. Alice inoltra N_b ad Ive criptando con la chiave di Ive, K_{ive} ;

3'. Ive apre il messaggio e lo invia a Bob, criptato con K_{bob}

Bob, che è l'utente di cui il protocollo si approfitta, riceve N_b . Controlla che il nonce sia quella che ha effettivamente inventato e a quel punto è sicuro di parlare con Alice. Ive ha quindi usato il nonce che ha inventato Alice per approfittarsi di Bob. Ha effettuato un *attacco di autenticazione*, in quanto Bob pensa di comunicare con Alice, ma in realtà sta parlando con Ive. Ive si spaccerà per Alice con Bob visto che si è autenticata al suo posto, quindi può chiedergli dei servizi. Per esempio, se Bob rappresentasse una banca, Ive potrebbe richiedere un trasferimento di denaro dal conto di Alice al proprio. Lo stesso tipo di attacco può essere studiato nella tassonomia BUG. Tale soluzione è deprecata ma continua ad essere utilizzata. Sappiamo che Alice ha condiviso con Ive il nonce N_a da lei generato, ma questo in realtà è stato inoltrato anche a Bob. Bob, nel caso in cui avesse delle intenzioni malevoli, potrebbe utilizzare N_a ai danni di Alice: se Alice vedesse arrivare un messaggio contenente N_a e N_b , infatti, penserebbe che questo sia stato inviato da Ive. Bob però, conoscendo l'importanza di N_a , potrebbe agire, stavolta vendicandosi su Ive. Il protocollo presenta delle vulnerabilità, quindi non è del tutto sicuro. Per ovviare a questo problema, si potrebbe aggiungere una comunicazione fuori banda. Ogni qual volta un utente voglia autenticarsi, prima di eseguire una sua qualunque istruzione, è necessario verificarne l'effettiva identità.

3.9 Protocollo di sicurezza di Woo-Lam (anni 80)

Usa la crittografia simmetrica e usa un TTP (Trusted Third Party), che possiede un database di tutte le chiavi. Un TTP è un'entità che facilita le interazioni tra due parti che si fidano entrambe della “terza parte”. L'obiettivo del protocollo è che Alice (A) riesca ad autenticarsi con Bob (B).

1. $A \rightarrow B : A$
2. $B \rightarrow A : N_b$
3. $A \rightarrow B : \{N_b\}K_a$
4. $B \rightarrow TTP : \{A, \{N_b\}K_a\}K_b$
5. $TTP \rightarrow B : \{N_b\}K_b$



1. A invia un messaggio pubblico a B;
2. B invia ad A un nonce N_b per verificare che stia effettivamente comunicando con lei. Anche questo messaggio è pubblico;
3. A riceve N_b e lo inoltra di risposta a Bob, ma criptata con K_a . K_a è una chiave simmetrica che ha in condivisione con il TTP e che B non può aprire. Bob deve verificare che K_a appartenga davvero ad A;
4. B invia la richiesta al TTP di aprire il messaggio per verificare l'appartenenza di K_a . Il messaggio è criptato con K_b e lo può aprire solo il TTP poiché condivide le chiavi simmetriche con B;
5. Il TTP, una volta decifrato il messaggio e ottenuto N_b , lo inoltra a B. B apre il messaggio finale perché possiede K_b e verifica che N_b sia il nonce che ha inviato ad A. Se è così, significa che B ha effettivamente comunicato con A. La decodifica di $\{N_b\}_{K_a}$ funziona solo se K_a appartiene ad A.

3.9.1 Attacco su Woo-Lam

Anche detto “interlacciamento di sessioni”.

1. C → B : A
- 1'. C → B : C
2. B → A : N_b
- 2'. B → C : N_b'
3. C → B : $\{N_b\}K_c$
- 3'. C → B : $\{N_b\}K_c$
4. B → TTP : {A, $\{N_b\}K_c\}K_b$
- 4'. B → TTP : {C, $\{N_b\}K_c\}K_b$
5. TTP → B : $\{N_b''\}K_b$
- 5'. TTP → B : $\{N_b\}K_b$

1. C'è l'attaccante C che si vuole spacciare per A con B e invia un messaggio a B confermando ciò;
- 1'. C invia anche un altro messaggio a B dove afferma invece di essere C; Alla ricezione dei due messaggi, B segue il protocollo e genera due nonce;
2. In risposta ad A genera N_b ;
- 2'. In risposta a C genera N'_b ; C intercetta i messaggi e quindi anche i due nonce.
- 3/3'. C invia N_b criptato con K_c a B due volte; B quindi riceve lo stesso messaggio due volte.
4. Supponendo che un messaggio sia stato inviato da A, B invia al TTP il nonce N_b che ha ricevuto. Il messaggio è poi criptato con K_b ;
- 4'. B invia un altro messaggio al TTP ma questa volta con C all'interno.
5. TTP invia a B il nonce N''_b ottenuto a partire dalla decifratura del messaggio 4 (nonce completamente sbagliato);
- 5'. TTP invia a B il nonce N_b ottenuto a partire dalla decifratura del messaggio 4'

B aveva associato ad A il nonce N_b quindi quando gli verrà ritornato penserà che è stato inviato da A. B andrà allora ad autenticare C al posto di A. Inoltre, se B attiva due comunicazioni contemporaneamente non può sapere in che ordine arriveranno i messaggi a causa del funzionamento di HTTP, quindi si potrebbe vedere ricevere prima il nonce "corretto" (quello inviato da TTP per C nel messaggio 4') invece che quello "corrotto" (inviato da TTP per A nel messaggio 4).

Il protocollo Woo-Lam è quindi vulnerabile. Per risolvere teoricamente basta che il TTP inserisca nel messaggio di ritorno quello che è l'utente per il quale ha decifrato il nonce.

Si porta all'attenzione dell'egregio lettore che esiste anche la versione simmetrica di Needham-Schröder (questa è un informazione dalla dubbia utilità).



Capitolo 4

Policy

La Policy sulla Sicurezza Informatica è quel documento nel quale sono contenute tutte le disposizioni, comportamenti e misure organizzative richieste ai dipendenti e/o collaboratori aziendali per contrastare i rischi informatici. Si va a identificare tutte quelle regole che possono essere stabilite su un Server così che le workstation collegate vengano “controllate” nella stessa maniera e per fare in modo che su di esse siano presenti le stesse caratteristiche. L’obiettivo è quello di garantire i tre goal della Sicurezza: riservatezza, integrità e disponibilità. Il compito della policy è quello di stabilire cosa è permesso e cosa non lo è, cioè distinguere cosa è reputato sicuro (e quindi autorizzato) da cosa invece può portare ad una violazione del sistema. Il sistema si muove da uno stato ad un altro. Ciò che deve fare la policy è fare in modo che questo non assuma mai stati non sicuri. Un sistema si dice “sicuro” quando ciò non accade mai. Quando un sistema permette ad un utente o ad un processo di entrare in uno stato non autorizzato si dice che si è verificato un “security breach”. Sia X un set di entità e I un’informazione:

- I ha riservatezza con rispetto a X se nessun membro di X può ottenere alcuna informazione su I;
- I ha integrità con rispetto a X se tutti i membri di X si fidano di I. Le nozioni di “fiducia” e “integrità” sono collegate. Se un sistema ha integrità, dobbiamo fidarci del fatto che il suo comportamento è corretto. Se I è una risorsa, la sua integrità implica che essa funzioni come dovrebbe (assurance);
- I ha disponibilità con rispetto a X se tutti i membri di X hanno accesso ad I.

Confidentiality Policy. Il suo scopo è garantire che tutto lo staff comprenda i requisiti dell'organizzazione in relazione alla divulgazione di dati personali e informazioni riservate. Riguardo al flusso delle informazioni è possibile possedere diversi diritti. Si può, infatti:

- Trasferire i diritti di accesso;
- Trasferire le informazioni senza però trasferire i diritti;
- Avere diritto di accesso alle informazioni solo per un certo periodo di tempo.

Il modello della policy spesso dipende dalla fiducia.

Integrity Policy. Definisce come l'informazione può essere modificata o alterata. Specifica:

- Chi può effettivamente compiere queste operazioni;
- Sotto quali condizioni i dati possono essere alterati;
- Eventuali limiti sulle modifiche dei dati.

Un mezzo per ottenere l'integrità è la separazione dei compiti. Si deve fare in modo che tutte le operazioni compiute (transazioni) mantengano il sistema in uno stato “consistente”.

Availability. Tipi di disponibilità:

- Tradizionale: x ha o no l'accesso;
- Quality of Service: viene promesso un certo livello di accesso che però non è garantito (ad esempio, un livello specifico di larghezza di banda).

Elementi base di una policy: Gli elementi base di una policy sono 3, ovvero:

- Subject: entità in grado di accedere ad una risorsa;
- Object: risorsa a cui vuole accedere l'entità e il cui accesso è regolamentato;
- Access right: descrive le modalità in cui una entità può accedere una risorsa;

La policy e i suoi meccanismi: La policy descrive ciò che è permesso e cosa no, i meccanismi, invece, controllano come questa viene implementata. Chiaramente gli utenti devono verificare sia policy che meccanismi utilizzati. Le policy prese in considerazione per il controllo sugli accessi sono:

- Discretionary Access Control (**DAC**): il proprietario determina i diritti di accesso. Solitamente sono identity-based access control (IBAC), ovvero il proprietario indica anche quali altri utenti possono avere l'accesso. È definita “discrezionale” perché un utente può avere diritti di accesso che gli consentono, di sua spontanea volontà, di consentire a un altro utente l'accesso alla risorsa;
- Mandatory Access Control (**MAC**): policy più restrittive. Stabiliscono a priori quali sono i comportamenti da evitare e quelli permessi. Ciò non viene specificato da chi crea la risorse, ma dalle regole generali che vengono messe in atto, per esempio quelle aziendali. In particolare controlla l'accesso in base al confronto tra le etichette di sicurezza (che indicano quanto sono sensibili o critiche le risorse del sistema) e i permessi di sicurezza (che indicano a quali determinate risorse possono accedere gli utenti del sistema). Questa politica è definita obbligatoria (“mandatory”) perché un'entità che ha il permesso per accedere a una risorsa NON può, di sua spontanea volontà, consentire a un altro utente di accedere a quella risorsa;
- Originator Controlled Access Control (**ORCON**): policy dove colui che assegna i diritti è il creatore. I possessori dei file non detengono diritti e non possono cederli a loro volta. Il creatore permette che il file sia diffuso a soggetti terzi con le seguenti restrizioni:
 - il file non può essere rilasciato ad altri soggetti senza il permesso del creatore;
 - ogni copia del file deve avere le stesse restrizioni;
- Role Based Access Control (**RBAC**): controlla l'accesso in base ai ruoli che gli utenti hanno all'interno del sistema e alle regole che stabiliscono quali accessi sono consentiti agli utenti in determinati ruoli. Quando si definisce una policy si devono sempre tenere presenti:
 - Gli utenti;
 - I ruoli che essi ricoprono: utente, utente segreto, sistemista, utente negligente..



- Le operazioni che possono essere compiute o no: leggere, scrivere, “downgrade”, cambio password...
- Le modalità con cui vietare o consentire determinate operazioni: obbligo, permesso, divieto, discrezionalità..
- Attribute-based access control (**ABAC**): controlla l’accesso in base sia agli attributi dell’utente che a quelli della risorsa a cui vuole accedere.

Matrice di Controllo degli Accessi: Questa matrice è principalmente utilizzata negli schemi di controllo degli accessi discrezionali (DAC), ovvero quegli schemi in cui un’entità può ricevere diritti di accesso che le permettono, di sua spontanea volontà, di consentire ad un’altra entità di accedere a qualche risorsa. Un approccio al DAC (ad esempio da parte di un sistema operativo o da un sistema di gestione di database) è quello di una **Matrice di Accesso** (concetto inizialmente formulato da Lampson).

Una tipica implementazione di questo schema consiste nell’avere:

- in una dimensione della matrice **i soggetti identificati** che possono tentare l’accesso ai dati e alle risorse,
- e nell’altra dimensione **gli oggetti** a cui si può accedere (al massimo livello di dettaglio, gli oggetti possono essere singoli campi di dati).

Raggruppamenti più aggregati, come record, file o anche l’intero database, possono anche essere oggetti nella matrice.

Ogni voce nella matrice indica i diritti di accesso (permessi) di un particolare soggetto per un particolare oggetto.

Alcuni esempi di permessi che possono essere presenti nelle voci della matrice sono:

- Lettura
- Scrittura
- Esecuzione
- Cancellazione
- Creazione
- Ricerche

		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

(a) Access matrix

Figure 4.2 Example of Access Control Structures

Figura 4.1: Esempio di Matrice di Controllo degli Accessi.

Nel caso in cui volessi avere accesso ad un file che non c'è, ci sarebbe il file safe default di sistema, il default deve essere sempre safe.

Esistono anche altri metodi per mappare gli accessi (**più efficienti rispetto alle matrici**), come le **Access Control List** e le **Capability List**. Rispetto alle Access Control Matrix sono meno costose perché non contengono coppie riga/colonna vuote e di conseguenza inutili. Sono utilizzate in genere per mappare appunto l'Access Control Matrix.

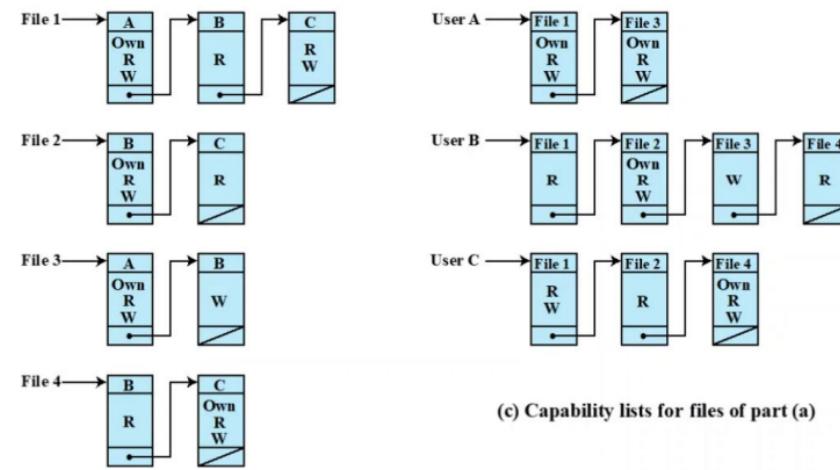


Figura 4.2: Access Control List e Capability List.



Soltamente può accadere che, quando la matrice di accesso è sparsa, viene implementata tramite decomposizione in uno dei due modi.

- La matrice può essere decomposta per colonne, ottenendo liste di controllo degli accessi (ACL). Per ogni oggetto, una ACL elenca gli utenti e i loro diritti di accesso consentiti. L'ACL può contenere una voce di default, o pubblica. Questo permette agli utenti che non sono esplicitamente elencati come aventi diritti speciali di avere un insieme predefinito di diritti. L'insieme predefinito di diritti dovrebbe sempre seguire la regola del minimo privilegio o dell'accesso in sola lettura, a seconda del caso. Gli elementi della lista possono includere sia utenti individuali che gruppi di utenti. Quando si vuole determinare quali soggetti hanno quali diritti di accesso a una particolare risorsa, le ACL sono convenienti, perché ogni ACL fornisce le informazioni per una data risorsa. Tuttavia, questa struttura di dati non è conveniente per determinare i diritti di accesso disponibili per un utente specifico.
- La decomposizione per righe produce i capability ticket. Un capability specifica gli oggetti e le operazioni autorizzate per un particolare utente. Ogni utente ha un certo numero di ticket e può essere autorizzato a prestarli o darli ad altri. Poiché i ticket possono essere dispersi nel sistema, essi presentano un maggiore problema di sicurezza rispetto alle liste di controllo degli accessi. L'integrità del biglietto deve essere protetta e garantita (di solito dal sistema operativo). In particolare, il biglietto deve essere non falsificabile. Questi biglietti dovrebbero essere tenuti in una regione di memoria inaccessibile agli utenti. Gli aspetti convenienti e scomodi dei biglietti di capacità sono l'opposto di quelli delle ACL:
 - è facile determinare l'insieme dei diritti di accesso che un dato utente ha,
 - ma è più difficile determinare l'elenco degli utenti con diritti di accesso specifici per una risorsa specifica.

4.1 Confidentiality Policies

Una politica di riservatezza, chiamata anche “information flow policy”, impedisce la divulgazione non autorizzata di informazioni. Il suo obiettivo è quello di specificare quali dati devono essere protetti e da chi o da che cosa



vanno protetti. In poche parole, indicano un insieme di principi e regole che definiscono i possibili accessi al sistema.

Distinguiamo tra:

- **Oggetti:** una qualunque entità passiva che necessita di essere protetta;
- **Soggetti:** una qualunque entità attiva che può manipolare gli oggetti (persone o processi).

Un modello di sicurezza definisce i soggetti e gli oggetti ai quali i soggetti hanno accesso ed i diritti di accesso, che non sono altro che le operazioni con le quali è possibile operare. Un soggetto può avere diritti di accesso sia per gli oggetti che per altri soggetti. Esistono due tipi di modelli di sicurezza:

- *Discretionary access control (DAC)*: meccanismo attraverso il quale i proprietari di un file possono liberamente decidere di garantire o revocare l'accesso ad altri utenti per quel file. In particolare si ha che:
 - Gli utenti amministrano i dati che possiedono (vengono detti proprietari);
 - Il proprietario può autorizzare altri utenti all'accesso;
 - Il proprietario può definire il tipo di accesso da concedere agli altri;
 - Accessi selettivi.
- *Mandatory access control (MAC)*: meccanismo attraverso il quale le decisioni di accesso sono basate su delle etichette che contengono informazioni rilevanti alla sicurezza di un oggetto. Questo metodo fornisce l'accesso alla risorsa in base al livello di autorizzazione dell'utente.
 - Classificazione dei dati (livelli di sensibilità);
 - Classificazione dei soggetti (autorizzazione);
 - Classe di sicurezza <componente gerarchica, insiemi di categoria>;
 - Ordinamento (parziale) tra le classi di sicurezza;
 - I meccanismi di sicurezza devono garantire che tutti i soggetti abbiano accesso solo ai dati per cui possiedono le autorizzazioni appropriate;
 - Non si possono propagare privilegi.



4.1.1 Modello Bell-LaPadula

È un modello tipico MAC. Fu proposto da Bell-LaPadula nel 1976 con lo scopo di rafforzare i controlli ai possibili accessi alle applicazioni militari (corrisponde, infatti, alle classificazioni stile-militare). Viene anche chiamato modello “*a multi-livelli*”. Nelle applicazioni i soggetti e gli oggetti vengono partizionati in differenti livelli di sicurezza. Un soggetto può solo accedere ad oggetti a certi livelli, i quali dipendono strettamente dal loro livello di sicurezza. Per esempio, i seguenti sono due tipici specificazioni di accessi: “una persona UNCLASSIFIED non può leggere informazioni a livello CONFIDENTIAL” e “informazioni TOP SECRET non possono essere scritte in files di livello UNCLASSIFIED”. Gli oggetti sono classificati in 4 diversi livelli di sensibilità, cioè di confidentiality. Ciascun oggetto può essere associato a uno o più livelli, detti compartments. Ad ogni soggetto, invece, viene associata una “*clearance*” ovvero un’autorizzazione. Una clearance non è altro che una coppia del tipo `<rank, compartments>` dove rank è il massimo livello di sensibilità dell’informazione a cui il soggetto ha accesso; compartment indica i compatti a cui il soggetto può accedere. A tutte le entità vengono assegnati dei livelli di sicurezza. In particolare:

- Un soggetto S possiede “*security clearance*” $L(S) = l_s$;
- Un oggetto O possiede “*security classification*” $L(O) = l_o$;

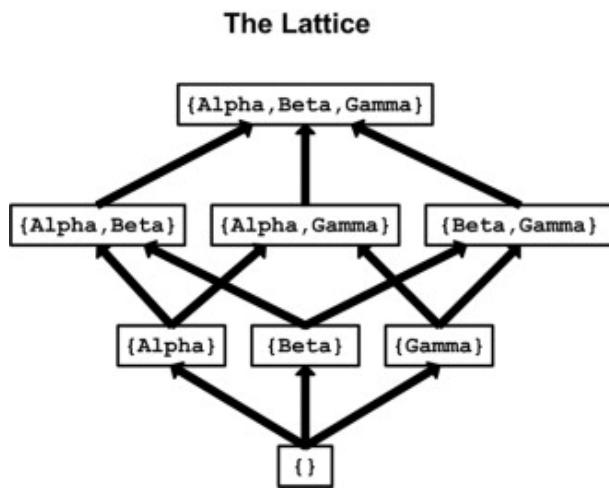
Per ogni classificazione di sicurezza l_i , $i = 0, \dots, k - 1$ con $l_i < l_{i+1}$, quindi i livelli di sicurezza sono disposti in ordine lineare. Il tipo più semplice di classificazione della confidenzialità è un insieme di autorizzazioni poste in gerarchia: Top secret > Secret > Confidential > Unclassified. Il modello definisce due regole obbligatorie del controllo accessi (MAC):

- **Simple Security Condition:** S può leggere O se e solo se $l_o \leq l_s$ e S ha discrezione nell’accesso in lettura a O. Questa regola viene anche detta “No Read Up” e impedisce ai soggetti di leggere oggetti a livelli superiori. Il proprietario può aggiungere anche dei diritti di tipo DAC, ovvero può restringere ulteriormente l’accesso.
- *** Property (Star property):** S può scrivere su O se e solo se $l_s \leq l_o$ e S ha discrezione nell’accesso alla scrittura su O. La regola è anche detta “No Write Down” e impedisce ai soggetti di scrivere su oggetti di livello inferiore. Questo perché se no un utente potrebbe declassificare informazioni o oggetti scrivendoli all’interno di oggetti con grado di classificazione inferiore.



Si potrebbe estendere il modello addizionando un gruppo di categorie per ogni classificazione di sicurezza. Ogni categoria descrive un tipo di informazioni. Tali categorie risultano dal principio del “Need To know”: limito l’accesso alle sole informazioni per le quali l’utente ha necessità di accedere.

Esempio: CATEGORIE → Alpha, Beta e Gamma. Ogni soggetto ha un’autorizzazione ad un determinato livello di sicurezza e può accedere anche ad alcuni dei livelli inferiori.



4.1.2 Multilevel Security

Il concetto viene trattato per la prima volta nel “Red Book” nel 1987. La sicurezza multilivello o più livelli di sicurezza (**MLS**) è l’applicazione di un sistema di computer per elaborare le informazioni con incompatibili classificazioni (cioè, a diversi livelli di sicurezza), consentire l’accesso da parte di utenti con diversi spazi di sicurezza e le esigenze di sapere (need to know) e impedire agli utenti di ottenere l’accesso alle informazioni per le quali non hanno l’autorizzazione. La sicurezza multilivello viene implementata in ambiti particolari, dove è necessario un controllo obbligatorio degli accessi, per assicurare che le politiche di sicurezza siano garantite dal sistema. L’ente definisce delle regole in merito a chi può accedere e anche a che cosa; queste regole non possono essere modificate dai singoli utenti. I sistemi MLS ci assicurano diversi livelli di sicurezza, L, che definiscono la classificazione dei soggetti (processi) e degli oggetti. I livelli di affidabilità, cioè di “assurance”, vengono stabiliti in base a vari criteri di valutazione:

$$(\text{worst}) \ D < C1 < C2 < C3 < B1 < B2 < B3 < A1 \ (\text{best})$$



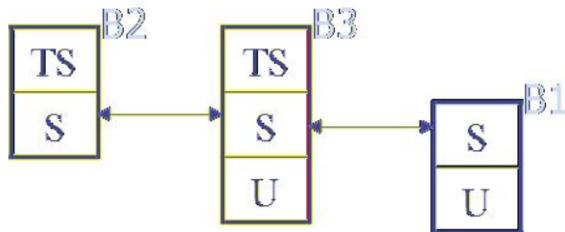
In base ai dati e alle circostanze possono essere richiesti livelli più o meno elevati. Solitamente, in presenza di un solo livello, è sufficiente una macchina del tipo C2 o C3. Da B1 in poi, invece, si possono gestire efficacemente informazioni su livelli multipli.

System Stores	Minimum Assurance
TopSecret + Unclassified	B3
TopSecret + Secret	B2
Secret + Unclassified	B1

TopSecret+Unclassified rappresentano in realtà 3 livelli, insieme al Secret. Attaccare una macchina B3 costa certamente di più che attaccare B1 o B2.

4.1.2.1 Channel Cascade Attacks

Prendiamo in esame la seguente immagine:



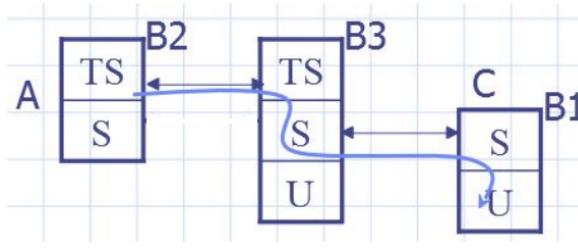
I livelli di sicurezza delle macchine sono B1, B2 e B3. Lo sforzo (l'effort) viene effettuato da un attaccante, il quale cerca di penetrare una macchina con un determinato livello di sicurezza. Il **“cascade attack”** si ha quando una sola macchina viene attaccata ad un certo livello, ma anche le altre vengono coinvolte conseguentemente. L'informazione tra i sistemi è condivisa, quindi può fluire tra le macchine. Nell'esempio, infatti, sono tutte collegate a livello S. Grazie all'effort su B2 si riesce a portare l'informazione da TS a U, che equivale a dire di aver buttato giù un sistema B3. Quindi, a partire da B2, si riesce a buttar giù anche B1 e B3. Si dice “attacco a cascata” perché grazie ad uno sforzo relativamente piccolo (nel nostro caso effettuato su B2) si possono attaccare anche sistemi più grandi e sicuri, poiché sono interconnessi. In generale quindi, se l'attacco ha successo, il livello di segretezza può essere abbassato da TopSecret a Secret, fino a Unclassified. Si ha un “cascading attack” se esiste un “cascading path”, ovvero un cammino tra sistemi che

costa meno dello sforzo necessario a rompere i singoli sistemi. Ricordiamo che eliminare i cascade path è un problema che rientra in NP.

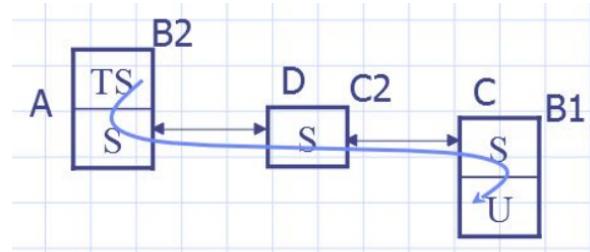
Come eliminare il Cascade Path

- Si possono disconnettere le macchine. Più c'è condivisione, più si alza la possibilità di avere cascading path;
- Mettere in comunicazione le macchine a livelli diversi;
- Definire i flussi consentiti (ognuno avrà uno specifico costo).

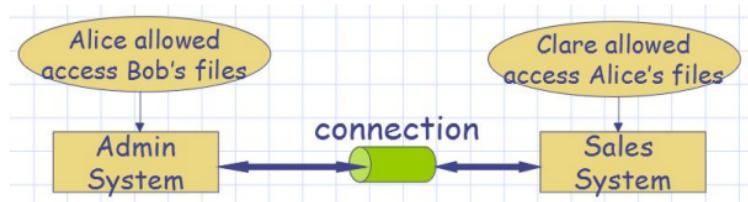
Il seguente non è un cascading path (non va bene perché dovremmo rompere un sistema B3. L'effort è uguale all'attacco, non c'è il guadagno).



Questo invece è un cascading path perchè, nonostante sembra che paghiamo il prezzo massimo dovendo rompere B2 per avere le informazioni TS, in realtà portiamo le informazioni non solo da TS a S ma poi, sempre al prezzo di B2, riusciamo a portarle fino ad U in B1. Quindi effettivamente guadagniamo qualcosa perchè il livello dell'attacco è più grande dello sforzo.



4.1.3 Secure Interoperation



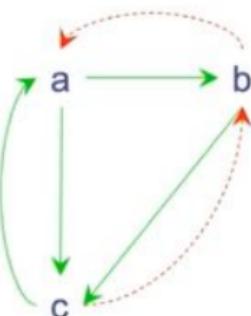
Supponiamo di avere due sistemi all'interno ad un'azienda, uno per le Vendite e uno per l'Amministrazione. Ognuno ha la propria policy che chiaramente definisce i diritti degli utenti. Alice e Claire lavorano su entrambi i sistemi ma Alice ha anche accesso ai file di Bob (Clare no). Entrambe vorrebbero che i filesystem venissero connessi (i sistemi sono sicuri individualmente). Se ciò avvenisse però, allora anche Clare avrebbe accesso ai file di Bob (transitività, passando per Alice) per i quali non è autorizzata. Ciò rappresenta a tutti gli effetti un attacco al sistema.

Le possibili soluzioni sono:

- Togliere a Clare l'accesso ai file di Alice;
- togliere ad Alice l'accesso ai file di Bob.

In generale quindi è necessario riconfigurare il sistema in modo da eliminare alcune connessioni e di conseguenza ridurre i diritti degli utenti.

4.1.4 Access Interoperation



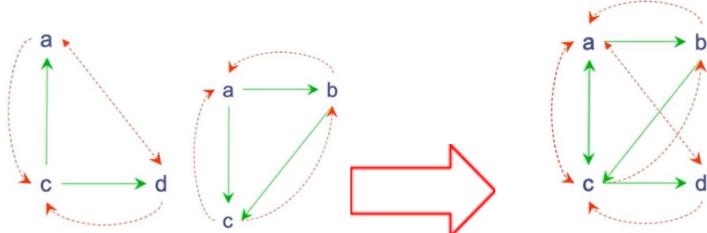
Consideriamo la figura precedente che ci mostra una policy (rappresentata dalle linee verdi) la quale definisce i seguenti accessi. L'accesso da A a C potrebbe essere ottenuto per transitività passando per B. Stesso discorso vale per C e B: da C si può arrivare a B tramite A. Ma il fatto che da C si possa raggiungere B viola la policy, perché questo accesso non è definito in realtà. È necessario quindi riconfigurare il sistema. Eventuali soluzioni potrebbero essere:

- Esplicitare i flussi, quindi ciò che è effettivamente permesso e cosa non lo è (ovvero aggiungere anche le linee rosse);
- Esplicitare soltanto ciò che è vietato (linee rosse) e quindi il resto è default permit;
- Esplicitare soltanto ciò che è permesso (linee verdi) e quindi il resto è default deny.

Va ricordato che la riconfigurazione deve essere safe. Ciò che viene stabilito con la nuova configurazione deve essere un sottoinsieme di ciò che era permesso anche prima. Non dobbiamo andare ad escludere per errore flussi già esistenti. La policy deve essere rispettata sempre, anche dopo la riconfigurazione.

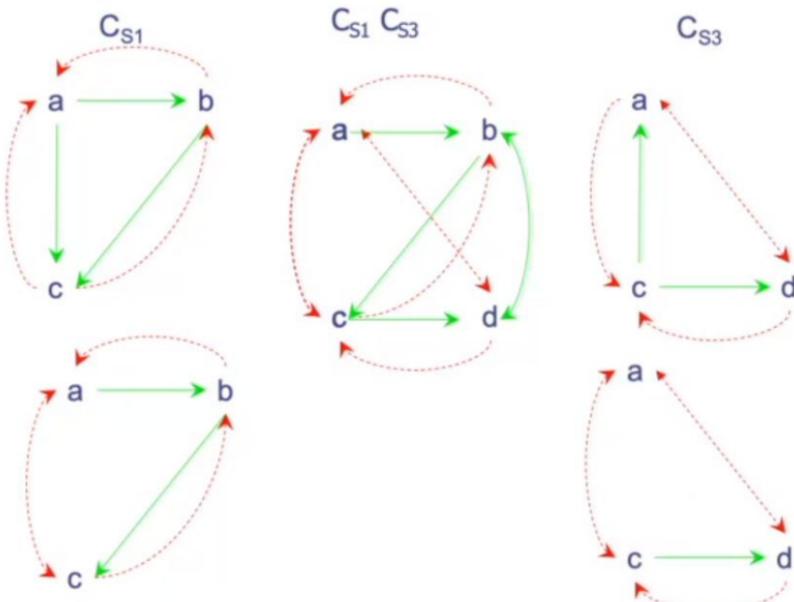
4.1.5 Secure Reconfiguration

Supponiamo di avere due differenti policy, corrispondenti a due diversi sistemi che devono però interoperare. Unire le due policy potrebbe non essere così semplice, in quanto potrebbero verificarsi dei conflitti. A questo punto è necessario capire a cosa dare priorità, se a ciò che è permesso o a ciò che non lo è. Solitamente si dà priorità al default deny, altrimenti non ci sarebbe garanzia del fatto che il sistema sia ancora safe.



Una volta esplicitato cosa è vietato, si può definire cosa è invece permesso, assicurando il fatto che non ci sia alcuna transitività. Il nuovo schema della

policy deve essere progettato sui sistemi a cui è destinato per verificare che non si presentino comunicazioni non corrette.



4.2 Integrity Policies

Possiamo distinguere due diversi tipi di integrità:

- **Il livello di integrità di un soggetto** è una misura della fiducia che si ripone nella sua capacità di produrre o gestire informazioni. Per esempio, un'applicazione certificata può avere una maggiore integrità rispetto al freeware scaricato da internet.
- **Il livello di integrità di un oggetto** descrive il grado di “affidabilità” dell’informazione contenuta in quell’oggetto.

Una meta-policy che i modelli di integrità devono soddisfare è necessariamente quella di non permettere che informazioni meno affidabili corrompano quelle più affidabili. Le informazioni con bassa integrità non dovrebbero fluire in domini ad alta integrità. Più alto è il livello, maggiore è la fiducia (trust). Quindi è più probabile che:

- Un programma venga eseguito correttamente;
- I dati siano precisi e/o affidabili.

È importante precisare che avere un alto livello di integrità non significa avere un alto livello di confidenzialità e viceversa. In generale se P è un processo con un grado X di “affidabilità” (**trustworthiness**) non può modificare dati ad un livello più alto di lui perché è certificato solo fino a quel punto; quindi può modificare o scrivere informazioni nei livelli sottostanti. I requisiti che una policy di integrità deve rispettare sono:

- Gli utenti non possono usare programmi scritti da loro stessi ma devono usare programmi e DB già esistenti;
- Separazione delle funzioni: i programmatore sviluppano e testano programmi su un sistema diverso da quello di produzione. Quando sono richiesti dei dati in fase di testing, non vengono presi i dati reali ma dei dati opportunamente modificati tramite un processo speciale. Tali dati vengono usati solo nella fase di sviluppo e non in quella di produzione;
- Separazione dei compiti;
- Logging and auditing: il processo deve essere controllato, verificato (audited), tracciato (logging) e deve offrire la possibilità di effettuare un’operazione di rollback;
- I manager e coloro che eseguono il processo di verifica devono avere accesso ad entrambi gli stati dei sistemi ed ai log generati da essi.

4.2.1 Modello BIBA

Il modello di integrità Biba venne sviluppato per aggirare le debolezze del modello di protezione Bell-LaPadula, il quale non prevedeva la possibilità di eliminazione implicita degli oggetti di sicurezza scritti da loro. Il modello di Biba in generale ha l’obiettivo di preservare l’integrità, ma in particolare vuole:

- Prevenire la modifica dei dati da soggetti non autorizzati;
- Prevenire modifiche non autorizzate ai dati da soggetti autorizzati;
- Mantenere la consistenza interna ed esterna (es.: dati che rispecchino il mondo reale).



Implementa le protezioni definendo una serie ordinata di livelli di integrità per i soggetti e gli oggetti, rispettando le regole:

- **No Read Down:** Questo significa che un soggetto che si trova al livello di integrità X può leggere solo oggetti allo stesso livello o superiore (“assioma di integrità semplice”);
- **No Write Up** un soggetto al livello di integrità X può scrivere solo oggetti allo stesso livello o di livello più basso (assioma di integrità * o Proprietà di semplice sicurezza);

Un esempio è quello della gerarchia militare: un generale può solo ricevere ordini dai suoi superiori o parigrado e può imporre ordini solo ai suoi sottoposti o parigrado per evitare un cambiamento/danneggiamento nella missione da svolgere. Da notare che *No Read Down* e *No Write Up* sono l’opposto delle regole *No Read Up*, *No Write Down* del modello Bell-LaPadula.

È impossibile trasformare informazioni meno trusted in più trusted. L’obiettivo di tale modello è quello di evitare che il grado di fiducia di un’informazione aumenti. Se ciò si verificasse, si potrebbe considerare fidata un’informazione che in realtà non lo è. Ad ogni risorsa del sistema viene assegnata un’etichetta indicante il livello di integrità minimo richiesto per l’accesso da parte dei soggetti. Le etichette di integrità, non coincidono con quelle di confidenzialità: le prime limitano le modifiche che si possono fare alle informazioni, mentre le seconde limitano il flusso delle informazioni. Maggiore è il livello di integrità e maggiore è la certezza che un programma sia eseguito correttamente e che i dati siano accurati e veritieri. Esistono delle relazioni tra integrità e la sua affidabilità. I livelli di integrità sono un punto importante per la sicurezza ma non coincidono con i livelli di sicurezza.

Ricordarsi che i modelli Bell-LaPadula e BIBA sono in conflitto tra loro e non si possono utilizzare contemporaneamente.

4.2.2 Modello Clark-Wilson

Rappresenta un’alternativa al modello di Biba ed utilizza le transazioni come operazioni base per garantire integrità prima e dopo le operazioni. Si basa su questi tre principi:

- **Separazione dei compiti:** Se sono necessarie due o più fasi per eseguire una funzione, almeno due persone devono eseguire le fasi separandosi adeguatamente i compiti;



- **Separazione delle funzioni:**

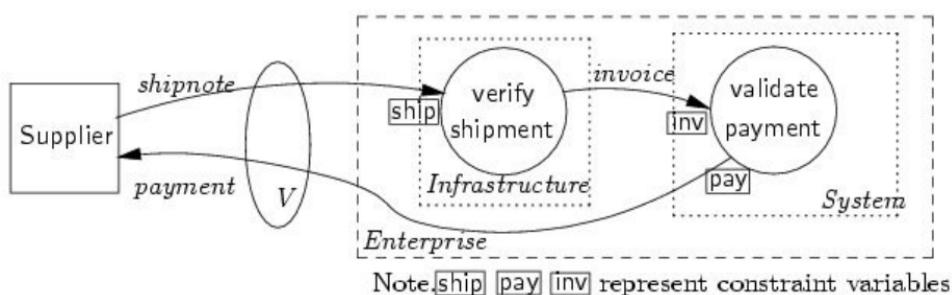
- Una singola persona non può svolgere ruoli complementari in un processo critico;
- Gli sviluppatori non sviluppano nuovi programmi sui sistemi di produzione;
- Gli sviluppatori non elaborano i dati di produzione sui sistemi di produzione;

- **Auditing:** I sistemi commerciali sottolineano l'importanza del recupero e della responsabilità (recoverability e accountability). I sistemi vengono analizzati per determinare quali azioni hanno avuto luogo e chi è stato coinvolto.

In generale un dato si può trovare in uno stato consistente o non consistente, in particolare si troverà nello stato consistente o valido se soddisfa un insieme di vincoli (definizione di integrità secondo Clark Wilson). I tasselli alla base del modello CW sono quindi:

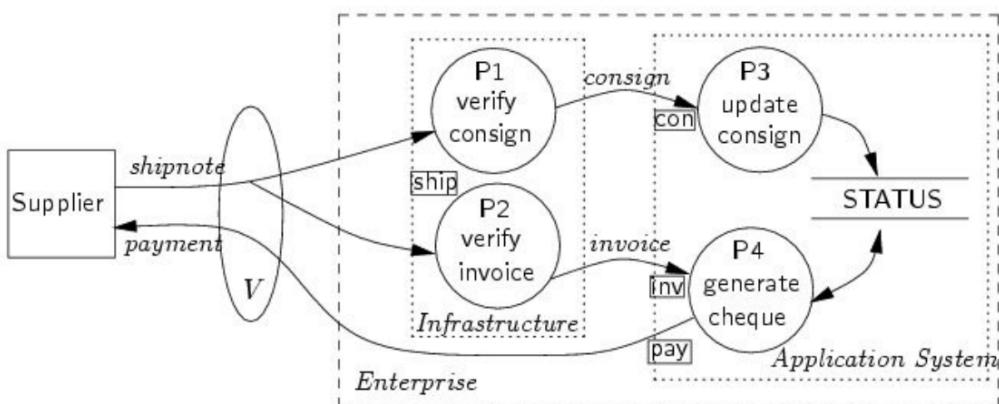
- **Autenticazione:** l'identità di tutti gli utenti deve essere autenticata correttamente;
- **Audit:** le modifiche devono essere tracciate (logging) per registrare ogni programma eseguito e da chi, in modo che non possa essere annullato;
- **Transazioni ben formate:** gli utenti manipolano i dati solo in modo limitato. Sono consentiti solo accessi legittimi;
- **Separazione dei compiti:** il sistema associa ad ogni utente un insieme di programmi che può eseguire. Impedisce le modifiche non autorizzate, preservando così l'integrità;

Esempio.



Consideriamo il momento in cui il fornitore consegna la merce ad una azienda e ciò è testimoniato dalla shipnote, ovvero la bolla di consegna. Una possibile organizzazione dell'impresa prevede che vi sia una persona addetta al magazzino che verifica se la merce ricevuta corrisponde a quella specificata nella bolla. Se così è, viene generata la fattura e avviene il pagamento per la merce. Un vincolo di integrità che vogliamo avere (dal lato dell'azienda) è che venga pagato solo quello che è stato consegnato, quindi che il valore del pagamento sia minore o uguale a quello indicato nella bolla di consegna. Il discorso è inverso per il lato del fornitore.

C'è poi una *Seconda Organizzazione* in cui la bolla viene affidata a due persone (separazione dei compiti): entrambe verificano che ciò che è stato consegnato corrisponda a quanto indicato dalla bolla ed eventualmente la inoltrano all'ufficio pagamenti. L'ufficio deve a questo punto constatare che le due comunicazioni siano coerenti tra loro.



Questo secondo sistema è più forte e ciò può essere verificato formalmente:

- funziona lo stesso anche se interviene un utente disonesto;
- può essere attaccato, ma occorrono due operai disonesti che scrivono bolle con importi più alti rispetto al valore reale;
- se un attacco dovesse avere successo, il sistema non risulta essere più sicuro dal lato integrità.

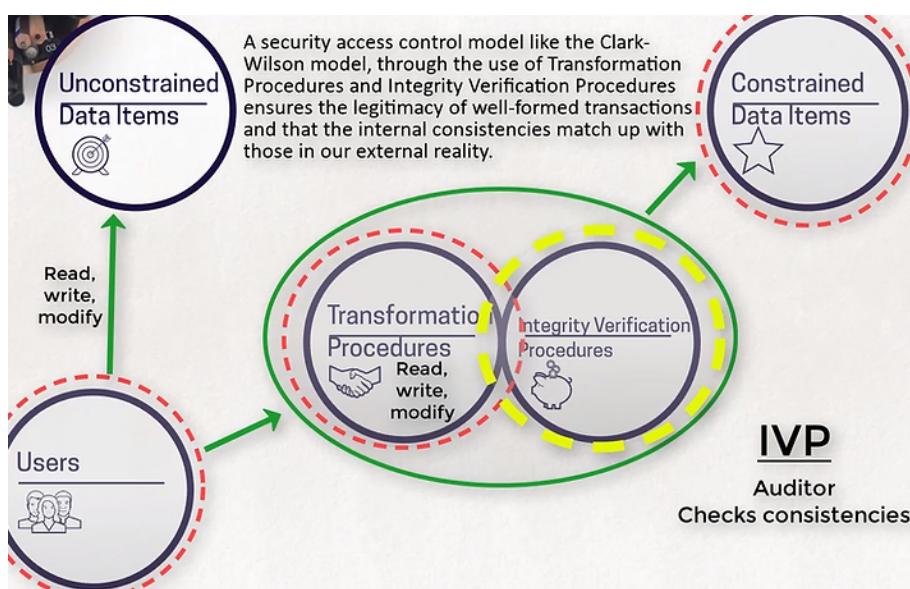
4.2.2.1 Come funziona?

Il modello Clark-Wilson è di tipo MAC e le regole sono decise dal sistema e non dagli utenti. Si basa su altri due concetti chiave:

- Una transazione deve essere ben formata (WFT), cioè deve far passare il sistema da uno stato consistente verso altri stati consistenti. E' pertanto importante stabilire un meccanismo che permetta di esaminare e certificare che le transazioni siano eseguite correttamente;
- Viene rispettato il principio della separazione dei compiti: l'operazione viene divisa in sottoparti che devono essere eseguite da soggetti differenti. In questo modo, per attuare una frode, ad esempio, è necessario che tutti i soggetti delle transazioni ne siano consapevoli. In pratica vi è una distinzione tra colui che esegue una transazione e colui che la certifica.

Il modello di Clark-Wilson richiede che il Sistema Informativo, nel quale viene applicato, disponga delle seguenti caratteristiche essenziali:

- Autenticazione da parte degli utenti;
- I dati possono essere manipolati da uno specifico set di procedure, che a loro volta possono essere eseguite solo da un utente autorizzato;
- Vengono mantenuti i log (contenenti programmi, dati e nomi degli utenti);
- I meccanismi di protezione non possono essere cambiati (staticità);
- Il security officer è il responsabile degli assegnamenti;
- Specifiche regole di sicurezza per la modifica dei dati.



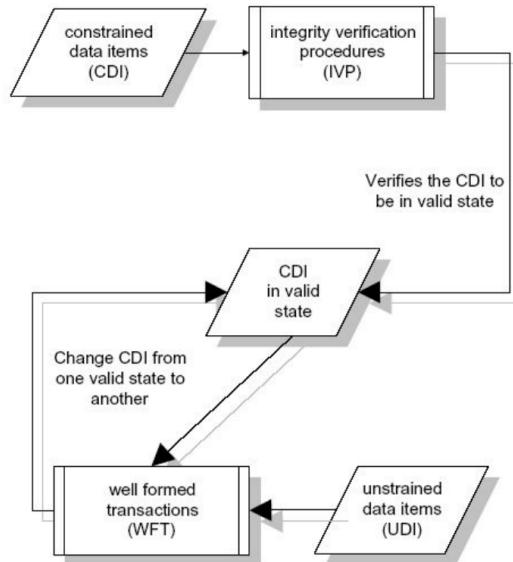
Formalmente il modello suddivide i dati in due set:

- **CDI** (*Constrained Data Item*): contiene gli elementi del sistema da proteggere, ossia i dati soggetti al controllo di integrità. Ad esempio in una banca, il CDI potrebbe essere composto dal saldo dei conti;
- **UDI** (*Unconstrained Data Item*): contiene l'insieme dei dati non soggetti ai vincoli di integrità.

L'appartenenza dei dati ad una determinata classe è esclusiva. Il modello prevede due tipi di procedure:

- **IVP** (*Integrity Verification Procedures*): è l'insieme delle funzioni che verificano se un determinato set di dati, appartenenti alla classe CDI, soddisfa determinati vincoli di integrità, i quali sono implementati come vincoli nel linguaggio SQL;
- **TP** (*Transformation Procedures*): è l'insieme delle procedure di trasformazione che hanno in input ed in output dei dati appartenenti alla classe CDI. Se un set dati CDI di partenza soddisfa un determinato IVP, allora lo soddisferà anche il set CDI ottenuto dopo la trasformazione; questo risulterà vero solo se la procedura è ben formata.

Le procedure tipo IVP permettono di garantire che il sistema parta da uno stato consistente, mentre le procedure TP assicurano che lo stato sarà consistente anche in futuro. Una tipica transazione nel database è composta da TP, le quali trasformano dati UDI in CDI oppure aggiornano dati CDI. Una IVP quindi garantisce che tutti i CDI nel sistema siano validi in un determinato stato. Un TP, invece, prende in input un CDI o un UDI e produce sempre un CDI. Il modello utilizza le transazioni come operazioni di base. L'integrità deve essere garantita prima e dopo le operazioni; di conseguenza i dati possono trovarsi in uno stato coerente oppure inconsistente. L'integrità viene definita mediante vincoli che i dati devono rispettare. Definire le caratteristiche di una transazione ben formata è un lavoro manuale eseguito da un security officer, il quale è l'unico che può specificare il set di procedure che possono modificare certi CDI.



Nel caso debbano entrare nel sistema degli UDI questi devono essere processati solo da WFT per risultare CDI. Il minimo di sicurezza richiesta comprende:

- Integrità: i vincoli d'integrità esistono per proteggere IS da modifiche maligne o accidentali dei dati. Le regole possono essere definite su stati statici del db o su transazioni (es. prima di poter effettuare un modifica);
- Identificazione, autenticazione: prima di accedere a un sistema ogni utente deve essere identificato e autenticato per mantenerne traccia e per dargli l'accesso;
- Autorizzazioni (cioè controllo sugli accessi)

4.2.2.2 Regole di certificazione

Al fine di garantire la validità dei dati nel sistema, il modello di Clark-Wilson prevede le seguenti regole di certificazione:

- **CR1** se un IVP è eseguito, deve assicurarsi che tutti i CDI siano in uno stato valido.
- **CR2** una TP deve trasformare un insieme di CDI da uno stato valido ad uno valido
 - CR2 definisce come “certificata” una relazione che associa un insieme di CDI ad una TP;

- CR2 implica che una TP può corrompere una CDI se non è certificata per lavorare con quel CDI
 - * *Esempio:* la TP che investe denaro nel portafoglio azionario della banca corromperebbe i bilanci anche se la TP fosse certificata a lavorare sul portafoglio, poiché le azioni della TP potrebbero non avere senso sui conti bancari
 - * Da ciò nasce la prima regola di rinforzo

4.2.2.3 Regole di rinforzo

Tutte le TP devono essere certificate per essere valide. Un TP può manipolare in modo errato (cioè non conforme ad un IVP) un CDI se non è stato certificato a lavorare per questo. Per evitare il problema, esistono delle regole di rinforzo (enforcement):

- **ER1** Il sistema deve garantire che solo le TP certificate ad operare su determinati CDI svolgano operazioni su di essi.
- **ER2** Il sistema deve associare a ciascun TP un utente e un insieme di CDI su cui l'utente e la TP possono lavorare. Il TP può accedere a quei CDI per conto dell'utente, ma non può accedere per conto di utenti non autorizzati.
 - Il sistema deve mantenere le relazioni certificate;
 - Il sistema deve restringere l'accesso degli utenti alle TP mediante una relazione di permesso che specifica quali utenti possono eseguire certi TP e su quali CDI.

4.2.2.4 Utenti e regole

- **CR3** Le relazioni di permesso devono essere basate sulla separazione dei compiti.
- **ER3** Il sistema deve autenticare ogni utente che utilizza una TP.

4.2.2.5 Uso dei Log

- **C4** Tutte le TP devono scrivere in CDI di solo accodamento (append) tutte le informazioni sufficienti a ricostruire le operazioni svolte.
 - Questo CDI è il log;
 - Gli amministratori devono essere in grado di capire cosa ha fatto una determinata transazione.

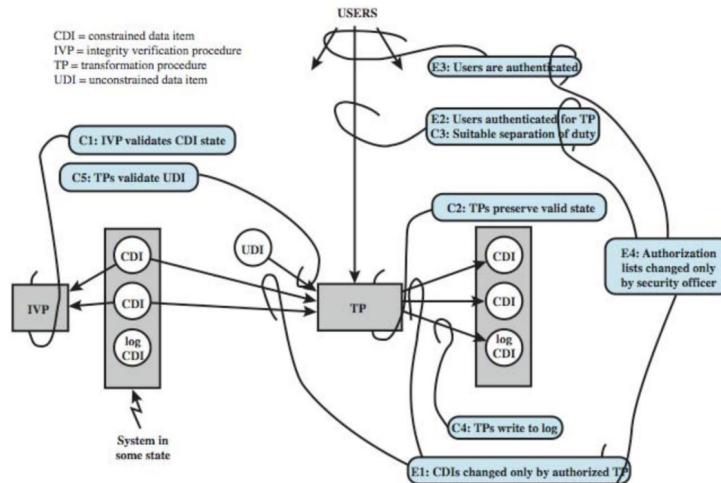


4.2.2.6 Trattamento di input non fidato

- **CR5** Ogni TP che prende in input un UDI può eseguire solo trasformazioni valide, oppure nessuna trasformazione, per ogni possibile valore dell'UDI. La trasformazione può quindi rifiutare l'UDI o trasformarlo in un CDI.
 - Un possibile esempio: in un banca i numeri inseriti da tastiera sono UDI, i quali non possono essere dati in input ad una TP. Un'apposita TP deve validare tali numeri (rendendoli un CDI) prima di usarli; se la validazione fallisce, la TP rigetta l'UDI.

4.2.2.7 Separazione dei compiti

- **ER4** Solo il certificatore di un TP può cambiare la lista di entità associate al TP. Nessun certificatore di un TP, o di un'entità associata al TP, potranno mai farlo.



4.2.2.8 I principali limiti

- Staticità delle relazioni di autorizzazione,
- Staticità della separazione dei compiti,
- Concessione centralizzata delle autorizzazioni assegnata all' amministratore della sicurezza.

4.2.2.9 Biba vs. Clark-Wilson

In Biba la fiducia nei soggetti viene valutata in base alle azioni che soddisfano le regole e non in base alla nozione di regola di certificazione. I dati non fidati vengono esaminati prima di essere dichiarati fidati. In Clark-Wilson ci sono degli esplicativi requisiti che regolano le azioni che possono essere fatte; inoltre le entità fidate devono certificare il metodo per aggiornare il dato da non fidato a fidato (e non il dato stesso). Biba si basa sull'integrità multi-livello, mentre Clark-Wilson focalizza la separazione dei compiti e delle transazioni.

4.3 Hybrid Policies

Le policy ibride servono per garantire sia la confidenzialità che l'integrità. Le principali sono: modello a Muraglia Cinese (**Chinese Wall**) basato sul concetto di conflitto di interesse; **ORCON** che combina le regole di controllo sugli accessi di tipo mandatory con quelle di tipo discretionary; **RBAC** dove l'accesso avviene tramite regole di controllo basate sui ruoli di appartenenza (usato in ambienti commerciali come SAP).

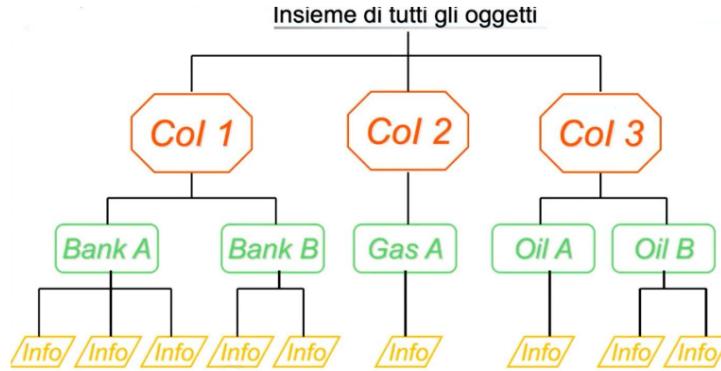
4.3.1 Chinese Wall

La politica della “*muraglia cinese*” è stata introdotta da Brewer e Nash nel 1989 per evitare che informazioni sensibili, concernenti una certa compagnia, vengano passate a compagnie rivali per mezzo di consulenti finanziari. Si stabiliscono dinamicamente i diritti di accesso degli utenti in base alle risorse a cui l'utente stesso ha già avuto accesso. Tale modello organizza le entità in classi basate su *Conflitti di Interesse*. Il controllo sull'utente viene effettuato prima dell'accesso ad ogni classe e al momento della scrittura, in questo ultimo caso il controllo viene fatto su tutte le classi per avere la certezza che l'informazione non violi le regole sui conflitti di interesse. Per ogni classe di conflitto di interesse saranno definite delle informazioni visibili a tutti chiamate “**Sanitized Data**” (informazioni prive di dati sensibili). Occorre definire tre elementi:

- *Oggetti (O)*: sono gli elementi di informazione relativi ad un'azienda;
- *Company Dataset (CD)*: contiene l'insieme degli oggetti relativi ad una singola azienda (l'operazione di inserimento di un oggetto è chiamata $CD(o)$);

- *Conflict of Interest Class (CoI)*: contiene l'insieme delle aziende in concorrenza tra di loro (l'operazione di inserimento di un'azienda è detta $CoI(o)$); ogni oggetto appartiene esattamente a una classe CoI.

Prendiamo in esame la seguente immagine:



Gli oggetti O sono le “Info”. I CD sono Bank A e Bank B, Gas A, Oil A e Oil B. Le banche A e B sono in conflitto tra loro e per questo nella medesima CoI indicata con 1. Gli altri CD appartengono alle CoI 2 e 3. Il protocollo vuole evitare che se un consulente accede ai dati della banca A, non può accedere a quelli della banca B perché esse sono chiaramente in conflitto. Quindi, se un consulente legge un oggetto appartenente ad un CD in una data CoI , non può più leggere oggetti di altri CD in quella CoI . È possibile che le informazioni apprese prima possano consentirgli di prendere decisioni “migliori” dopo (ovviamente in modo sleale). Indichiamo con $PR(S)$ l'insieme degli oggetti che S ha già letto. Una semplice condizione di sicurezza in fase di lettura, prevede che un soggetto S può leggere un oggetto O se e solo se:

- Esiste un oggetto o' a cui S ha avuto accesso e $CD(o') = CD(o)$ oppure
- Per ogni $o' \in O$, $o' \in PR(S) \Rightarrow CoI(o') \neq CoI(o)$.

o (minuscolo) è un oggetto sanitized e perciò non genererà conflitti di interesse. In questa politica non vengono presi in considerazione i dati sanitized ed inizialmente $PR(S) = \emptyset$. In altri termini, un soggetto può leggere un oggetto se l'oggetto è in un dataset di cui il soggetto ha già letto qualcosa oppure l'oggetto appartiene a una CoI di cui il soggetto non ha letto ancora niente. Nel nostro esempio, se il consulente legge dalla banca A non potrà leggere dalla B. Non appena viene compiuta un'azione di lettura viene negato



l'accesso ad altri dati che potrebbero generare conflitti di interessi. La politica **Chinese Wall** è una combinazione di *libera scelta* e *MAC*. Inizialmente un soggetto è libero di accedere a ciò che vuole ma, una volta effettuata la scelta, per quell'utente viene creata una *Muraglia Cinese* attorno al dataset a cui l'oggetto appartiene. Si noti che la Chinese Wall può essere combinata con le politiche DAC.

Esempio. Anthony e Susan lavorano per la stessa azienda di consulenza. Anthony può leggere i *CD* di **Bank A** e di **Gas A** e Susan può leggere i *CD* di **Bank B** e di **Gas A**. Se Anthony potesse scrivere sul *CD* di **Gas A**, Susan potrebbe leggerlo. Perciò se i due si coalizzano per condividere i dati dall'azienda, Antony può scrivere su **Gas A** (che ha in comune con Susan) i dati della **Bank A**. Così Susan, che può leggere **Gas A** leggerà invece i dati di **Bank A** nonostante sia un rivale perché lavora con **Bank B**, un chiaro conflitto di interesse. La regola così descritta per la lettura non è in grado di prevenire “fughe di notizie”.

Quindi, in fase di scrittura, il modello deve stabilire che un soggetto S può scrivere un oggetto o se e solo se:

1. La condizione di sicurezza in lettura permette a S di leggere o ,
2. Per ogni oggetto non *sanitized* (quindi sensibile) o' , se S può leggere o' , allora $CD(o') = CD(o)$.

In pratica S può scrivere un oggetto se tutti gli oggetti sensibili che può leggere appartengono allo stesso dataset. Un utente che ha letto più *CD* non potrà scrivere nessun oggetto; inoltre, una volta che ha scritto in un *CD*, potrà scrivere soltanto lì. Così il flusso di informazioni è destinato a restare all'interno dell'azienda. In altri termini, un soggetto può scrivere un oggetto se lo può anche leggere e non può leggere dati di altre compagnie.

4.3.1.1 Chinese Wall vs Bell-LaPadula

Chinese Wall non assegna etichette di sicurezza ma si basa sugli accessi passati. Bell-LaPadula può simulare Chinese Wall istante per istante, in questo caso: ogni coppia di (*CoI*, *CD*) costituirà una categoria, ci saranno due livelli di sicurezza (S (*sanitized*), U (*non sanitized*), in cui S domina U), ad ogni soggetto deve essere associata al massimo una sola categoria per ogni classe *CoI*. Però, Bell-LaPadula non sarà in grado di gestire gli accessi passati e di adattare i vincoli relativi ai permessi di accesso con il variare del tempo (in Chinese Wall infatti, inizialmente si ha accesso a tutti gli oggetti, poi no).



4.3.1.2 Chinese Wall vs Clark-Wilson

Se non è prevista una distinzione tra i soggetti ed i processi, allora si rispetterebbe il modello di Clark-Wilson ma si violerebbe il modello Chinese Wall (una singola persona può accedere a più processi). Se però è prevista una separazione tra soggetto e processo allora si dovrebbero rispettare entrambi i modelli in termini di sicurezza di integrità.

4.3.2 ORCON

Il modello ORCON stabilisce che i diritti di accesso vengono definiti dall'utente che ha creato l'oggetto. Il diritto di lettura di un dato non è concesso dal possessore del dato ma da colui che l'ha originato.

1. Il soggetto $s \in S$ marca l'oggetto $o \in O$ come ORCON per conto dell'organizzazione X .
2. X permette che o sia diffuso ai soggetti che lavorano per conto dell'organizzazione Y con le seguenti restrizioni:
 - o non può essere rilasciato a soggetti che lavorano per conto di altre organizzazioni senza il permesso di X ;
 - Ogni copia di o deve avere le stesse restrizioni.

Per realizzare questa politica non sono idonei né *MAC* né *DAC*. *DAC* non può essere utilizzato poiché il possessore potrebbe concedere qualsiasi diritto violando la seconda proprietà. *MAC* è inadeguato poiché:

- può causare un'esplosione del numero di categorie;
- la gestione degli accessi avviene in maniera centralizzata, mentre in ORCON è richiesta una gestione locale.

La soluzione prevede la combinazione di *MAC* e *DAC* generando la seguente tecnica di gestione degli accessi: il possessore dell'oggetto non può cambiare i controlli di accesso. Quando l'oggetto viene copiato, vengono copiate anche le restrizioni di accesso dalla sorgente (cioè è *MAC*, il possessore non può controllarlo); il creatore del documento può alterare le restrizioni di accesso in base al soggetto e all'oggetto (cioè è *DAC*, il creatore può controllarlo).

4.3.3 RBAC

Il modello **RBAC** (*Role Based Access Control*) nasce poiché un problema importante nell’organizzazione di grandi sistemi è la complessità dell’amministrazione della sicurezza. Quando il numero dei soggetti e degli oggetti è alto, il numero di autorizzazioni può diventare molto grande. Inoltre, se l’utenza è dinamica, il numero di concessioni e di revoca di permessi diventa davvero elevato. Gli utenti finali spesso non “possiedono” le informazioni a cui hanno accesso. Le aziende o gli enti sono i reali possessori degli oggetti. Il controllo di accesso è quindi basato sulle mansioni delle persone e non sul semplice possesso. RBAC è stato quindi proposto come alternativa al DAC e al MAC per semplificare la gestione degli accessi e per supportare direttamente il controllo basato sui ruoli. I diritti di accesso dipendono dal ruolo del soggetto ma non dalla sua identità. I ruoli rappresentano le mansioni all’interno di un’organizzazione; le autorizzazioni sono concesse ai ruoli anziché ai singoli utenti. Gli utenti sono perciò autorizzati semplicemente ad assumere ruoli appropriati, acquisendo le autorizzazioni assegnate a quei ruoli. Poiché i ruoli rappresentano le funzioni dell’organizzazione, un modello RBAC può direttamente supportare le politiche di sicurezza proprie dell’organizzazione. La concessione e la revoca delle autorizzazioni alle categorie di utenti è estremamente semplificata. I modelli RBAC sono anche detti “**policy-neutral**”.

4.3.3.1 Modello RBAC-NIST

Il modello RBAC NIST è una definizione standard di controllo di accesso basato sui ruoli. Nel 2000, il NIST ha richiesto uno standard unificato per RBAC, integrando il modello RBAC pubblicato nel 1992. Il modello dispone di tre livelli con capacità funzionali crescenti: **Core RBAC** (o Flat RBAC), **RBAC Gerarchico**, **RBAC Vincolato**. Formalmente, RBAC definisce:

- **Utente**: un essere umano, una macchina, un processo, o un agente intelligente autonomo, ecc.;
- **Ruolo**: una funzione nel contesto di un’organizzazione con una semantica associata secondo l’autorità e la responsabilità del ruolo;
- **Permesso**: modo di accesso che può essere esercitato sugli oggetti nel sistema.
- Sia gli oggetti e i modi di accesso sono dipendenti dal dominio;
- **Sessione**: è una particolare istanza di una connessione di un utente al sistema e definisce un sottoinsieme di ruoli attivati.



- Ad ogni istante, possono essere attive più sessioni differenti per ciascun utente.
- Quando un utente entra nel sistema, stabilisce una sessione e durante tale sessione può attivare un sottoinsieme dei ruoli che è autorizzato ad assumere;
- L'utente ottiene i permessi associati al ruolo che ha attivato nella sessione

RBAC Flat. In questo primo modello vengono definiti solo gli elementi essenziali senza i quali non è possibile implementare un controllo d'accesso.

- il **ruolo** è un insieme di funzioni; $\text{trans}(r)$ indica le transazioni autorizzate per il ruolo r ;
- I ruoli attualmente attivi per il soggetto s sono dati da $\text{actr}(s)$;
- il set di ruoli autorizzati per il soggetto s si calcolano con $\text{authr}(s)$;
- le azioni che il soggetto s può fare al tempo t si ricavano da $\text{canexec}(s, t)$ il cui risultato dipende dal ruolo di s .

Sia S l'insieme dei soggetti e T quello delle transazioni: le transazioni che il soggetto può fare dipendono dal ruolo di s ; i ruoli che s ha attivi sono un sottoinsieme dei ruoli autorizzati per s . Se t è una transazione che il soggetto s può eseguire, allora essa è anche una di quelle transazioni permesse a tutti i soggetti del ruolo ricoperto da s .

RBAC Gerarchico. La gerarchia tra i ruoli è un modo naturale per strutturarli, riflettendo le linee di autorità e responsabilità di un'organizzazione. Ad un ruolo gerarchicamente più importante, saranno assegnate le stesse azioni di un ruolo meno importante più alcune specifiche, ovvero:

$$(\forall s \in S)[r' \in \text{authr}(s) \wedge r' > r \rightarrow r \in \text{authr}(S)]^1$$

In tale modello è applicabile il principio di ereditarietà:

- *Ereditarietà di utente (UI)*: Tutti gli utenti autorizzati ad un ruolo r sono anche autorizzati ad un ruolo r' , ove $r > r'$

¹È inutile ed incomprensibile, ma era carina da vedere.



- *Eredità di permessi (PI)*: Un ruolo r è autorizzato a tutti i permessi per i quali ogni ruolo r' , tale che $r > r'$, è autorizzato
- *Eredità di attivazione (AI)*: Attivando un ruolo r automaticamente si attivano tutti i ruoli r' , tali che $r > r'$.

RBAC Vincolato. È un modello RBAC con la capacità di supportare le politiche di *Separazione dei compiti (Separation of Duty)*. Evita conflitti di interesse e collisioni tra mansioni. Sia r un ruolo e s un soggetto tale che r appartiene a $\text{auth}(s)$, allora il predicato $\text{meauth}(r)$ (usato per assegnare autorizzazioni in mutua esclusione) è l'insieme dei ruoli che s non può assumere a causa dei requisiti di separazione dei compiti.

$$(\forall r_1, r_2 \in R)[r_2 \in \text{meauth}(r_1) \rightarrow [(\forall s \in S)[r_1 \in \text{authr}(s) \rightarrow r_2 \notin \text{authr}(s)]]]^2$$

In pratica, la *Separation of Duty* stabilisce che se un ruolo r_2 è in conflitto con un ruolo r_1 , allora ogni soggetto s autorizzato al ruolo r_1 non può essere autorizzato al ruolo r_2 . Esistono due tipi di categorie di separazione dei compiti (Separation of Duty):

- **Statici**: restringono l'insieme dei ruoli, ed in particolare la possibilità di entrare nella relazione RA. Per evitare che una persona sia autorizzata ad assumere troppi ruoli, nessun utente può assumere più di un certo numero di ruoli in un determinato insieme di ruoli;
- **Dinamici**: limitano il numero di ruoli che un utente può attivare in una singola sessione. Ad esempio si può stabilire un massimo numero di ruoli in una sessione, oppure si può stabilire che se assunto un ruolo in una sessione non si possono assumere altri ruoli nella stessa sessione. Ovviamente per garantire questa tecnica di assegnamento dei compiti, è importante mantenere una storia dei ruoli attivati da ciascun utente nell'arco della sessione.

In conclusione, le politiche ibride riguardano sia la riservatezza che l'integrità. Diverse combinazioni di questi modelli ORCON non sono né MAC né DAC. In realtà, un modello RBAC combinato controlla l'accesso in base alle funzionalità.

²Anche questa è inutile ed incomprensibile, ma carina da vedere.



Capitolo 5

Risk Management Process

Il processo di “*Risk Management*” è l’insieme di attività coordinate per gestire un’organizzazione con riferimento ai rischi. Tipicamente include l’identificazione, la misurazione e la mitigazione delle varie esposizioni al rischio. Il rischio è l’incertezza che eventi inaspettati possano manifestarsi producendo effetti negativi per l’organizzazione. Il rischio di Information Technology è definito come il pericolo di interruzione di servizio (*availability*), diffusione di informazioni riservate (*riservatezza*) o di perdita di dati rilevanti archiviati tramite mezzi computerizzati (*integrità*). Per operare su questi rischi è stata introdotta la “*Information Security Risk Management*”. Va ricordato che Identificazione e Misurazione sono racchiusi nella Risk Assessment.



5.1 Risk Assessment

Il processo di “*Risk Assessment*” è usato per determinare l’ampiezza delle potenziali minacce ad un sistema IT ed identificare tutte le possibili contromisure per ridurre o eliminare tali voci di rischio. Vengono identificati:

- **asset**: qualsiasi bene di proprietà di un’azienda (macchinari, merci, ma anche il database o la rete) che deve essere protetto;
- **minacce**: possibili attacchi, quindi non ancora avvenuti;

- **vulnerabilità**: falle di sicurezza che portano al bypass delle policy;
- **contromisure**: da implementare per evitare le vulnerabilità;

Di conseguenza vengono determinati:

- impatto prodotto dalle minacce: per capire quale vulnerabilità andare a coprire rispetto ai mezzi finanziari a disposizione;
- fattibilità delle minacce: probabilità con cui può verificarsi un attacco;
- complessivo livello di rischio (che decido di correre come azienda nel globale).

5.2 Risk Mitigation

Nel processo di “*Risk Mitigation*” vengono analizzate le contromisure raccomandate dal team di assessment e poi selezionate e implementate quelle che presentano il miglior rapporto costi/benefici. Solitamente si adotta la modalità degli “**Attack Tree**” (approcci qualitativi). Più nel dettaglio distinguiamo:

- Approcci **qualitativi**: Analisi degli scenari che possono realizzarsi all’interno di un sistema. Lo scopo è quello di individuare le possibili minacce e il livello di rischio associato ad ogni risorsa che compone il sistema. Vengono chiamati “Attack Tree” perché l’analisi dei possibili attacchi viene effettuata per mezzo della costruzione di alberi.
- Approcci **quantitativi** (Indici economici): Quantificazione di tutte le grandezze necessarie per una valutazione dei rischi con l’obiettivo di determinare, attraverso l’uso di una serie d’indici, la convenienza economica di un investimento in sicurezza.

5.2.1 Attack Tree

Gli *Attack Trees* sono diagrammi concettuali che mostrano come una risorsa o un obiettivo, potrebbe essere attaccata. Gli Attack Trees sono diagrammi multilivello costituiti da una radice, foglie e figli. Dal basso verso l’alto, i nodi figli sono condizioni che devono essere soddisfatte per rendere vero il nodo genitore diretto; quando la radice è soddisfatta, l’attacco è completo. Ogni nodo può essere soddisfatto solo dai suoi nodi figli diretti. Un nodo può essere figlio di un altro nodo; in tal caso, diventa logico che sia necessario



eseguire più passaggi per attuare un attacco. Un attacco descritto in un nodo può richiedere che uno o più dei molti attacchi descritti nei nodi figli siano soddisfatti. Si possono creare e verificare condizioni di OR o di AND lungo l'albero.

Abbiamo quindi:

- Radice: rappresenta l'asset che è oggetto dell'attacco,
- Nodi: sono possibili percorsi dell'attaccante per effettuare l'attacco. I nodi non foglie possono essere nodi AND, si devono eseguire entrambi gli attacchi per raggiungere il nodo, o nodi OR, dove ogni singolo percorso è un attacco.

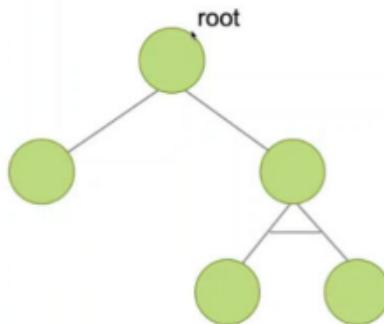


Figura 5.1: Questa immagine rappresenta un Attack Tree.

Gli alberi di difesa (**Defence Tree**) sono un'estensione degli Attack Trees (nodi quadrati). Si costruiscono per capire quali sono le possibili contromisure da adottare. Considerando tutte queste possibilità di un eventuale attacco, devo definire le contromisure adatte per la protezione dell'asset, tenendo sempre presente l'attack tree. Un metodo piuttosto semplice prevede di mettere gli attacchi in ordine di importanza e scegliere le contromisure da adottare in base ad una certa priorità: per esempio in base alla pericolosità, al costo, ecc.

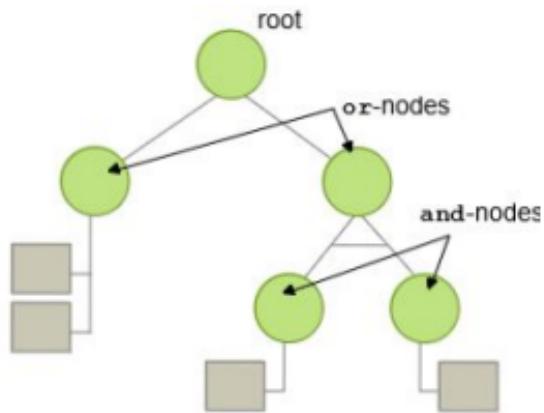


Figura 5.2: Questa immagine rappresenta un Defence Tree.

5.2.2 Cp-nets

Le conditional preference networks, dette anche cp-nets, sono delle rappresentazioni grafiche usate per esprimere un ordine di preferenza.

Per capire meglio il concetto riportiamo subito un esempio (Figura 5.3). Poniamo il caso che siamo in un ristorante e che non sappiamo quali piatti sono disponibili e ci vogliamo affidare al cameriere su cosa mangiare. Utilizzando solo una frase siamo in grado di esprimere una preferenza condizionale su due variabili, in questo caso il piatto D e il vino W. Per ogni variabile dell'insieme di variabili, l'utente può specificare la variabile genitore che può influenzare la sua preferenza: nel nostro esempio D è genitore di W perché il tipo di piatto influenza la scelta del vino.

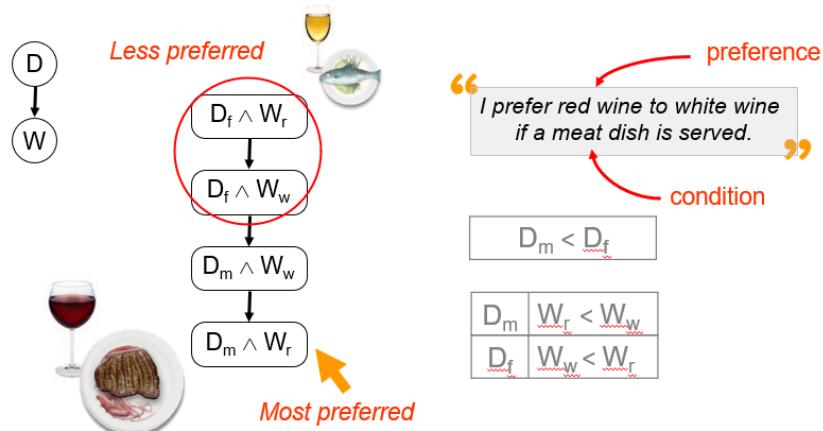


Figura 5.3: Esempio utilizzo cp-nets.

Come visto in Figura 5.3, la preferenza va dalla condizione “meno preferita” a quella “preferita” seguendo un ordine o da sinistra verso destra o dall’alto verso il basso.

Sfruttando lo stesso concetto le cp-nets vengono usate in combinazione ai Defence Tree in maniera tale che:

- I *Defence Tree* ci permettono di determinare, in modo qualitativo, le strategie di attacco che un attaccante può seguire per danneggiare un sistema, le diverse azioni che compongono ogni attacco e le misure di sicurezza che possono essere adottate nel sistema.
- Le *cp-nets* vengono poi usate per determinare l’ordine di preferenza dell’amministratore di sistema nella selezione delle contromisure da adottare per ogni singolo nodo di attacco nel Defence Tree.

Di seguito possiamo vedere un reale esempio di utilizzo delle cp-nets con i Defence Tree. In questo caso l’attacco a_4 è meno pericoloso dell’attacco a_3 che è meno pericoloso dell’attacco a_5 e così via. Poi, per ogni attacco, andiamo a dare un ordine di preferenza alle relative contromisure adottabili. Ad esempio, per l’attacco a_1 avremo che la contromisura c_1 sarà meno preferita rispetto alla contromisura c_2 ma che a sua volta sarà meno preferita della contromisura c_3 .

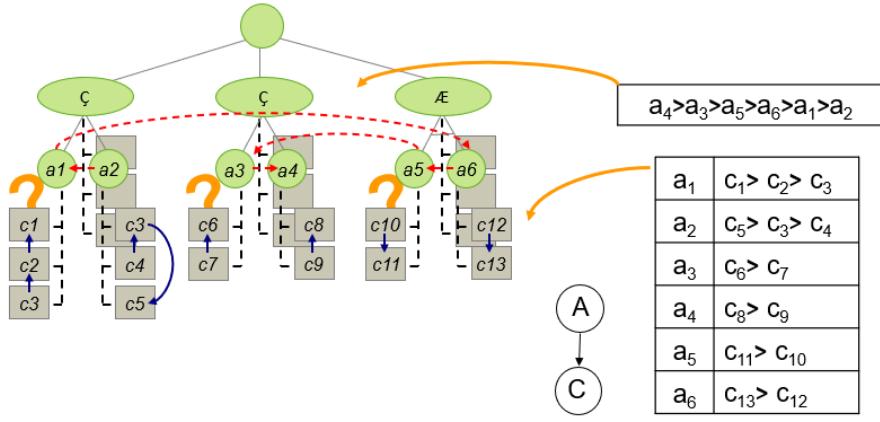


Figura 5.4: Esempio utilizzo cp-nets con Defence Tree.

5.2.3 Indici Economici

SLE. La *Single Loss Exposure (SLE)* rappresenta una misura della perdita di un’impresa da un singolo evento di minaccia e può essere calcolata utilizzando la seguente formula:

$$SLE = AV \times EF$$

Dove l’*Asset Value (AV)* è il costo di creazione, sviluppo, supporto, sostituzione e proprietà di un bene; l’*Exposure Factor (EF)* rappresenta una misura dell’entità della perdita o dell’impatto sul valore di un’attività derivante da un evento di minaccia.

ALE. The *Annualized Loss Expectancy (ALE)* è la perdita finanziaria annuale prevista di un’impresa che può essere attribuita a una minaccia. Può essere calcolata utilizzando la seguente formula:

$$ALE = SLE \times ARO$$

Dove *Annualized Rate of Occurrence (ARO)* è un numero che rappresenta il numero stimato di eventi annuali di una minaccia. Questo è il costo da proteggere.

ROI. L’indicatore *Return on Investment (ROI)* è dato da:

$$ROI = \frac{(ALE \times RM) - CSI}{CSI}$$

Dove RM è il rischio mitigato (Risk Mitigated) da una contromisura, rappresenta l'efficacia di una contromisura nel mitigare il rischio di perdita derivante dallo sfruttamento di una vulnerabilità; CSI è il costo dell'investimento in sicurezza (Cost of Security Investment) che un'impresa deve affrontare per attuare una determinata contromisura.

ROA. Il *Return On Attack* (**ROA**) misura il guadagno che un aggressore si aspetta da un attacco riuscito, rispetto alle perdite che subisce a causa dell'adozione di misure di sicurezza da parte del suo obiettivo.

$$ROA = \frac{GI \times (1 - RM) - (cost_a + cost_{ac})}{cost_a + cost_{ac}}$$

Dove:

- GI è il guadagno atteso dall'attacco riuscito al bersaglio specificato;
- $cost_a$ è il costo sostenuto dall'aggressore per avere successo;
- $cost_{ac}$ è il costo aggiuntivo portato dalla contromisura c adottata dal difensore per mitigare l'attacco a .

Altri Indicatori.

- *Exposure Factor* durante il Critical Time: esprime l'influenza che la criticità di una specifica istanza temporale gioca sull' EF .
- *Critical Time*:
 - *Annualized Rate of Occurrence*, **AROCT**, è il tasso di occorrenza di un attacco ad una specifica CTF all'anno;
 - *Single Loss Exposure* (**SLECT**), è il costo di un singolo attacco ad un specifico CTF:

$$SLECT = AV \times EFCT$$

- *Annualized Loss Expectancy* (**ALECT**) è il costo annuo di un attacco presso una specifica CTF:

$$ALECT = SLECT \times AROCT$$

- *Return On Investment*, **ROI CT** , è il ritorno economico dell'investimento di un'impresa contro un attacco sferrato in un determinato CTF:

$$ROI CT = \frac{(ALECT \times MR) - CSI}{CSI}$$



5.3 Risk Monitoring

All'interno di grandi imprese i sistemi IT subiscono frequenti modifiche dovuti ad aggiornamenti, cambiamento dei componenti, modifica dei software, cambio del personale, ecc. Mutano le condizioni del sistema, modificando anche gli effetti delle contromisure adottate.

Capitolo 6

Security Design

6.1 Principi Fondamentali

I principi fondamentali di progettazione della sicurezza sono una serie di linee guida che consentono agli sviluppatori di creare sistemi protetti. In totale sono 13 di cui i primi 8 sono stati definiti da un docente di nome Saelzer in un articolo del 1975. Gli altri sono stati aggiunti dai ricercatori nel tempo. Nello specifico:

Economy of mechanism La progettazione delle misure di sicurezza incorporate nell'hardware che nel software dovrebbe essere la più semplice e piccola possibile. Tenere il sistema più semplice possibile nel progetto, nell'implementazione, per quanto riguarda il numero di operazioni, le interazioni con altre componenti e addirittura nella specifica. “Semplice” significa che ci sono meno cose da controllare e quindi in caso di errori è più facile identificarli e correggerli. Particolare attenzione deve essere posta nelle interfacce ad altri moduli (possono fare assunzioni non verificate); bisogna sempre definire completamente tutte le interfacce, ad esempio le variabili ambientali e le variabili globali, nonché le liste dei parametri. Include altri principi come:

- **KISS** è un acronimo usato in progettazione, che sta per “*Keep It Simple, Stupid*”, ossia ”*rimani sul semplice, stupido*”. In riferimento al codice sorgente di un programma significa non occuparsi delle ottimizzazioni fin dall'inizio, ma cercare invece di mantenere uno stile di programmazione semplice e lineare, affidando le ottimizzazioni al compilatore o a successive fasi dello sviluppo;

- **Minimalists Design:** gli sviluppatori possono creare interfacce utente fatte per essere il più semplice possibile eliminando pulsanti e finestre di dialogo che possono potenzialmente confondere l'utente;
- **Worse is better:** la qualità non è necessariamente funzionale;
- **YAGNI:** l'espressione “*you aren't gonna need it*”, dall'inglese “*non ne avrai bisogno*”, in ingegneria del software si riferisce a un principio secondo cui un programmatore non dovrebbe sviluppare software che implementi funzionalità non esplicitamente richieste.¹

Fail-safe defaults Le decisioni di accesso dovrebbero essere basate sull'autorizzazione piuttosto che sull'esclusione: la situazione predefinita è la mancanza di accesso e il sistema di protezione identifica le condizioni in cui l'accesso è consentito (default deny). Per default l'accesso di un soggetto ad un oggetto deve essere negato. Ogni diritto deve essere concesso esplicitamente. Inoltre se un'azione fallisce il sistema deve restare sicuro, come se l'azione non fosse stata svolta. In realtà sarebbe più corretto parlare di **fail-secure**: un sistema è fail-secure quando in caso di un determinato guasto, risponde in modo tale che l'accesso o i dati vengano negati. Un sistema **fail-safe**, invece, in caso di guasto non provoca alcun danno o solo in minima parte. Mentre i prodotti fail-safe sono sbloccati quando viene tolta la corrente (cioè viene applicata la corrente per bloccare la porta), i prodotti fail-secure sono bloccati quando viene tolta la corrente (cioè viene applicata la corrente per sbloccare la porta). In sintesi, i sistemi:

- **Fail-secure:** vige la default-deny nei firewall. I programmatori dovrebbero controllare i valori di ritorno in caso di eccezioni o fallimenti; si devono basare le decisioni sul permesso, non tanto sull'esclusione.
- **Fail-insecure:** rappresentano esempi di design sbagliati.

Complete mediation Bisogna controllare ogni accesso: di solito il controllo viene fatto al primo. Ad esempio in UNIX il controllo di accesso in lettura ad un file è fatto solo all'apertura del file e non ad ogni operazioni di lettura. Se i permessi cambiano durante l'operazione, si possono generare accessi non validi.

¹Un esempio è la vulnerabilità trovata all'interno della libreria Log4J. Per saperne di più <https://www.cybersecurity360.it/nuove-minacce/log4shell-ecco-come-funziona-lexploit-della-vulnerabilita-di-log4j/>, <https://logging.apache.org/log4j/2.x/security.html>.



Open design La progettazione di un meccanismo di sicurezza dovrebbe essere aperta e non segreta. La sicurezza non dovrebbe dipendere dalla segretezza nella progettazione o dell’implementazione del sistema, ma dalla bontà di tale progettazione ed implementazione. Si potrebbe però intendere che il codice sorgente dovrebbe essere pubblico. È molto importante ricordare che questo metodo non va bene con informazioni tipo passwords o chiavi crittografiche.

Separation of privilege Sono necessari molteplici attributi di privilegio per ottenere l’accesso a una risorsa limitata. Non si può concedere un diritto in base ad una condizione sola. Ad esempio in molti Unix per diventare root bisogna conoscere la password di root e far parte del gruppo wheel. Perciò si devono richiedere più condizioni per garantire un privilegio. Anche il design può dipendere dai diversi privilegi: non sempre l’accesso al file viene considerato un privilegio, come lo stesso accesso alla rete. Vanno inoltre distinti lettura e scrittura. Vige anche il principio:

- della “separazione funzionale”: persone diverse, pur avendo diversi compiti, devono cooperare;
- del “controllo duale”.

Least privilege Ogni processo e ogni utente del sistema dovrebbe operare ricorrendo al minor numero di privilegi necessari per eseguire il compito. I meccanismi di accesso alle risorse dovrebbero essere condivisi il meno possibile. L’informazione può fluire attraverso i canali condivisi.

Least common mechanism Il progetto dovrebbe ridurre al minimo le funzioni condivise dai diversi utenti, garantendo la sicurezza reciproca (separazione dei compiti). Un soggetto deve possedere solo i diritti di cui ha bisogno per svolgere i propri compiti. I diritti sono concessi in base alla funzione, non all’identità (RBAC). I diritti vengono aggiunti quando servono e quando non servono più sono revocati (sessions/Dynamic). Non tutti i sistemi, a causa della granularità dei permessi, sono in grado di soddisfare tale principio.

Psychological acceptability Implica che i meccanismi di sicurezza non devono interferire sul lavoro degli utenti, soddisfacendo al tempo stesso le esigenze di chi autorizza l’accesso. I meccanismi di sicurezza non dovrebbero rendere più difficile l’accesso alle risorse. È pertanto importante la chiarezza nei messaggi di errore e la facilità di installazione, configurazione e uso.



Isolation Questo principio che si applica in tre contesti:

1. i sistemi di accesso pubblico dovrebbero essere isolati dalle risorse critiche per evitare la divulgazione ed eventuali manomissioni;
2. i processi e i file dei singoli utenti dovrebbero essere isolati l'uno dall'altro, tranne quando ciò sia esplicitamente voluto;
3. i meccanismi di sicurezza dovrebbero essere isolati nel senso di impedire l'accesso ad essi.

Encapsulation Considerato come una forma specifica di isolamento basata sulla funzionalità orientata agli oggetti.

Modularity Si riferisce sia allo sviluppo di funzioni di sicurezza come moduli separati e protetti, sia all'uso di un'architettura modulare per la progettazione e l'implementazione di meccanismi.

Layering Si riferisce all'uso di approcci di protezione multipli e sovrapposti che riguardano le persone, la tecnologia e gli aspetti operativi dei sistemi informativi.

Least astonishment Un programma o un'interfaccia utente dovrebbe sempre comportarsi nel modo in cui la maggior parte degli utenti si aspetta che si comporti, senza quindi stupire o sorprendere gli utenti.

Defence in Depth La “*difesa in profondità*” o *Defense in Depth* è l'approccio alla sicurezza delle informazioni che prevede il raggiungimento di una corretta postura di sicurezza attraverso l'utilizzo coordinato e combinato di molteplici contromisure di sicurezza. Questa complessa strategia difensiva si fonda sull'integrazione di tre differenti categorie di elementi: le *persone*, la *tecnologia* e le *modalità operative*. La ridondanza e la distribuzione delle contromisure possono essere sintetizzate in due frasi: “*difesa in differenti posizioni*” e “*difesa a differenti livelli*” (“*Defense in Multiple places*”, “*Layered Defenses*”). Nel caso che un attacco abbia successo, che tradotto in termini di difesa consiste nel fallimento di un meccanismo di sicurezza, altri meccanismi di sicurezza possono intervenire per consentire un'adeguata protezione dell'intero sistema.



Confinement problem Uno dei principali problemi che si può presentare in un sistema in cui ci sono più utenti ed un server che elabora le richieste, è il “*confinement problem*”: evitare che il server lasci trapelare informazioni che l’utente ritiene confidenziali. Una possibile soluzione è rappresentata dall’*isolamento totale*, in cui i processi non possono comunicare tra loro, né osservare gli altri. Ciò evita la fuoriuscita di informazioni dal sistema, ma è irrealizzabile se i processi usano delle risorse osservabili. In tal caso, si potrebbe aggirare il problema utilizzando i **covert channel**, i quali sfruttano dei percorsi nati per compiere altre operazioni diverse dalla comunicazione, per fare invece delle comunicazioni in realtà proibite. Un’altra soluzione, potrebbe essere quella che prevede l’utilizzo di **sandbox**, ossia scatole dove viene confinata l’esecuzione di processi e su cui sono vigenti delle policy di sicurezza restrittive a piacere.

6.2 Trust Network Brainstorming

Ogni sistema di sicurezza dipende dalla fiducia, in una forma o nell'altra, degli utenti del sistema. In generale esistono diverse forme di fiducia per affrontare diversi tipi di problemi e mitigare il rischio in determinate condizioni. Quale forma di fiducia applicare in una determinata circostanza è generalmente dettata dalla politica aziendale. Le **trust network** (reti di fiducia) consistono in ramificazioni di connessioni interpersonali costituite principalmente da forti legami in cui una parte è disposta a dipendere da qualcuno, in una determinata situazione, con una sensazione di relativa sicurezza, anche se sono possibili conseguenze negative a causa del fallimento altrui. Le relazioni di fiducia consentono agli utenti di un dominio di accedere alle risorse di un altro dominio. I trust funzionano facendo sì che un dominio si affidi all'autorità dell'altro dominio per l'autenticazione dei suoi account utente. La scelta che deve affrontare l'amministratore della sicurezza consiste nel dare maggiore importanza all'autonomia o alla sicurezza del sistema: in entrambi i casi, l'altro aspetto verrà necessariamente sacrificato.

Le proprietà del trust sono:

- grafo in cui i nodi sono gli utenti mentre gli archi indicano la relazione di fiducia;
- peso associato ad ogni arco ($0 = \text{mistrust}$, $1 = \text{max trust}$);
- soggettività: l'utente può avere opinioni differenti rispetto agli altri;
- asimmetria: la fiducia che detiene un utente verso un altro non deve essere necessariamente speculare;
- i giudizi sono dipendenti dal contesto.

Semirings viene utilizzata come struttura generica per il calcolo della fiducia nelle reti fiduciarie (cioè pesate). I *semirings* “bipolari” permettono di considerare insieme fiducia e sfiducia e di avere così una composizione sicura delle reti trust (avviene una fusione critica ogni volta che due comunità basate sulla fiducia devono essere amalgamate).



Capitolo 7

Database & Data Center Security

7.1 Introduzione

Negli ultimi anni il modello di Database Relazionale è diventato molto popolare ed ampiamente utilizzato da privati e aziende. Per molte organizzazioni è importante essere in grado di fornire a clienti, partner e dipendenti l'accesso alle informazioni da loro memorizzate. Ma tali informazioni possono essere prese di mira da minacce interne ed esterne, possono essere utilizzate impropriamente o modifiche da entità non autorizzate. Di conseguenza, la sicurezza specificamente adattata ai database è una componente sempre più importante di una strategia generale per la protezione dei dati.

Di seguito andremo ad elencare le principali ragioni per cui la sicurezza dei database non ha tenuto il passo con l'aumento della dipendenza dai database:

1. C'è un drammatico squilibrio tra la complessità dei moderni database (DBMS) e le tecniche di sicurezza usate per proteggere questi sistemi critici. Un DBMS è un software molto complesso e di grandi dimensioni, che fornisce molte opzioni che devono essere tutte ben comprese e poi protette per evitare violazioni dei dati.
2. I database hanno un sofisticato protocollo di interazione chiamato *Structured Query Language (SQL)*, che è molto complesso e potente, quindi deve essere appreso pienamente prima di poter essere utilizzato.
3. La mancanza di personale specializzato. L'azienda media non ha personale addetto alla gestione della sicurezza di database. Generalmente hanno uno staff per la sola amministrazione del DB che non è sufficientemente formato per gestirne anche la sicurezza.



4. La maggior parte degli ambienti aziendali consiste in una miscela eterogenea di piattaforme di database, piattaforme aziendali e OS, quindi non avere un solo DB o un solo tipo di sistema operativo rappresenta una problematica che crea un'ulteriore complessità per il personale.
5. La crescente dipendenza dalla tecnologia cloud per ospitare parte o tutto il database aziendale risulta essere un'ulteriore sfida in quanto aggiunge ulteriori oneri per il personale di sicurezza.

Possiamo definire quindi un **Database** come una raccolta strutturata di dati immagazzinati e messi in relazione tra di loro per essere utilizzati da una o più applicazioni; organizzati in tabelle che possono essere manipolate tramite un apposito linguaggio chiamato Data Definition Language (**DDL**); recuperati tramite un apposito linguaggio di query standardizzato (**SQL**) e gestiti da un apposito software chiamato **Database Management System**.

Il principale vantaggio dei Database, oltre alla possibilità di organizzare e recuperare informazioni in pochissimo tempo, è quella di poter limitare l'accesso e la visibilità solo a singole parti di un determinato file. Nei classici sistemi operativi questo non risulta possibile dato che si può solo specificare se un utente può o no accedere ad un file ma non limitarne la visibilità solo ad una sua parte.

7.2 SQL Injection Attack

Gli attacchi di tipo SQL Injection sono una delle minacce alla sicurezza diffuse e pericolose. Questo attacco si basa sull'invio di dati contenenti una parte di query malevola per l'estrazione, modifica o cancellazione di tutti i dati presenti all'interno di un database fruibile tramite un'applicazione web.

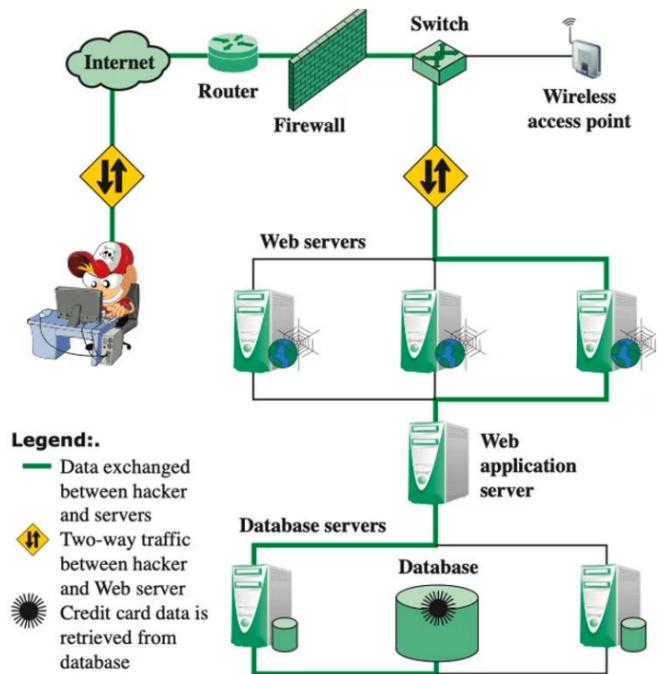
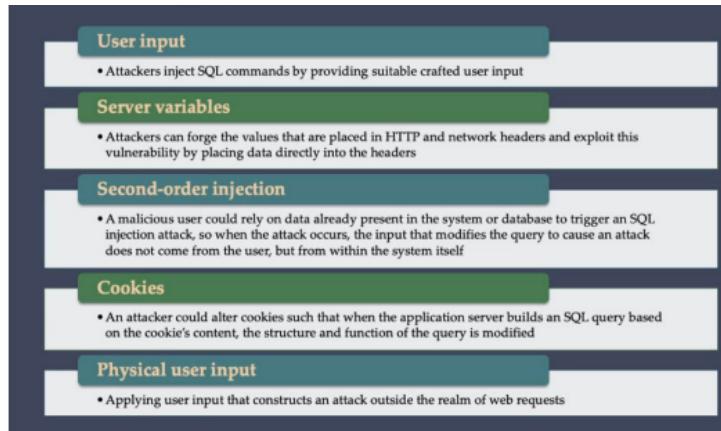


Figura 7.1: Esempio di un tipico attacco SQL Injection.

Questo attacco funziona tipicamente terminando prematuramente una stringa di testo, generalmente con un commento `--`, e aggiungendo un nuovo comando che verrà eseguito dal database ed impedirà l'esecuzione del resto della query.

Possiamo caratterizzare gli attacchi SQL Injection in termini di via d'attacco e tipo di attacco.

7.2.1 Attack Avenues



Le principali vie d'attacco sono le seguenti:

Input dell'utente: l'aggressore fornisce al database un input opportunamente modificato per forzare l'esecuzione di una query malevola.

Variabili server: Le variabili del server sono un insieme di variabili che contengono intestazioni HTTP, intestazioni del protocollo di rete e variabili ambientali. Le applicazioni web usano queste variabili del server in una varietà di modi, come la registrazione di statistiche e informazioni di utilizzo per identificare le tendenze di un utente. Se queste variabili sono registrate in un database senza sanificazione, questo potrebbe creare una vulnerabilità di SQL injection in quanto è possibile modificarle ed inserirci codice arbitrario.

Second Order Injection: Un attaccante può sfruttare dati già presenti nel sistema o nel database per attivare un attacco SQL Injection cosicché quando avviene effettivamente l'attacco la query malevola non proviene dall'utente ma direttamente dal sistema stesso.

Cookie: Di solito vengono utilizzati anche i cookie per creare in modo automatico query per salvare dati o manipolarli. Un attaccante può modificare il contenuto di un cookie per andare a forzare l'esecuzione di una query maligna all'interno di un database.

Input fisico dell’utente: Gli attacchi SQL Injection possono anche avvenire all’esterno di una richiesta web: prendiamo in considerazione un sistema che in automatico scannerizza e archivia documenti tramite l’ausilio di modelli di machine learning, questo può essere attaccato scrivendo fisicamente una query malvagia all’interno del documento da acquisire.

7.2.2 Attack Types

I tipi di attacco possono essere raggruppati in tre categorie principali: **In-band**, **Inferential**, **Out of Band**.

In-Band. Un attacco in banda utilizza lo stesso canale di comunicazione per iniettare codice SQL e recuperare i risultati. I dati recuperati sono presentati direttamente nella pagina web dell’applicazione. I tipi di attacco In-Band includono i seguenti:

- **Tautologia:** Questa forma di attacco inietta codice in una o più dichiarazioni condizionali in modo che valutino sempre vero.
- **Commento di fine riga:** Dopo aver iniettato codice in un particolare campo, il codice legittimo che segue viene annullato attraverso l’uso di commenti di fine riga. Un esempio potrebbe essere quello di aggiungere “--” dopo gli input in modo che le query rimanenti non siano trattate come codice eseguibile, ma come commenti.
- **Query piggybacked:** L’attaccante riesce ad aggiungere ulteriori query oltre a quella originale. Questa tecnica si basa su configurazioni del server che permettono diverse query all’interno di una singola stringa di codice.

Inferential. Sono attacchi che non fanno trasferimento dei dati, quindi sono attacchi su confidenzialità e non integrità. L’attaccante è in grado di ricostruire le informazioni e risalire a specifiche segrete del database inviando particolari richieste e osservando il comportamento risultante del server. Le principali metodologie di attacco sono:

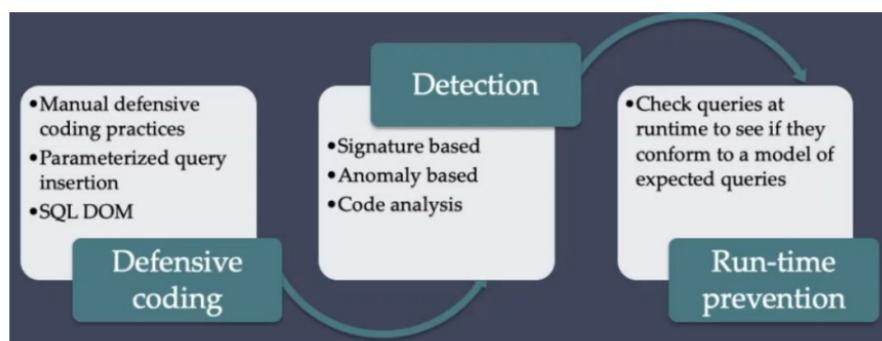
- **Query illegali/logicamente scorrette:** La vulnerabilità sfruttata da questo metodo è che la pagina di errore predefinita restituita dai server delle applicazioni è spesso eccessivamente descrittiva. Infatti, il semplice fatto che viene generato un messaggio di errore può spesso rivelare parametri vulnerabili per un attaccante.

- **Blind SQL Injection:** Permette agli attaccanti di dedurre i dati presenti in un sistema di database anche quando il sistema è sufficientemente sicuro da non mostrare alcuna informazione errata all'attaccante.

Out-of-band. Sono degli attacchi in cui l'input dell'attacco avviene per un canale e l'output per un altro canale. In un attacco fuori banda, i dati vengono recuperati utilizzando un canale diverso, ad esempio una e-mail con i risultati della query viene generata e inviata all'attaccante. Questo può essere utilizzato quando ci sono limitazioni sul recupero delle informazioni, ma la connettività in uscita dal server del database non presenta importanti restrizioni.¹

7.3 SQL Injection Mitigation

Poiché gli attacchi SQL Injection sono così diffusi, dannosi e variegati sia per via dell'attacco che per tipo, una singola contromisura non è sufficiente. Piuttosto un insieme integrato di tecniche è necessario. Le contromisure possono essere classificate in tre tipi: *Defensive Coding*, *Detection*, e *Run-time Prevention*.



Defensive Coding:

- **Pratiche manuali di codifica difensiva:** Una vulnerabilità comune sfruttata da SQL Injection è l'insufficiente convalida dell'input. La soluzione semplice per eliminare queste vulnerabilità è applicare adeguate pratiche di codifica difensiva. Un esempio è il controllo del tipo di input, per verificare che gli input che si suppone siano numerici

¹Questo concetto è finemente espresso con la seguente parola: *lassista* .

non contengano caratteri diversi dalle cifre. Questo tipo di tecnica può evitare attacchi basati sulla forzatura di errori nel sistema di gestione del database. Un altro tipo di pratica di codifica è quella che esegue il pattern matching per cercare di distinguere l'input normale da quello anormale.

- **Inserimento di query parametrizzate:** Questo approccio cerca di prevenire l'SQL Injection permettendo allo sviluppatore dell'applicazione di specificare più accuratamente la struttura di una query SQL, e passare i parametri di valore ad essa separatamente in modo che qualsiasi input dell'utente non possa modificare la struttura della query.
- **SQL DOM:** è un insieme di classi che permette la validazione automatica del tipo di dati, la validazione e l'escape. Questo approccio usa l'incapsulamento delle query di database per fornire un modo sicuro e affidabile per accedere ai database. Questo cambia il processo di costruzione delle query da un processo non regolamentato che usa la concatenazione di stringhe ad uno sistematico che utilizza un'API controllata.

Detection:

- **Signature based:** Questa tecnica tenta di rilevare specifici pattern di attacco in quanto ogni tipologia di attacco presenta un'esecuzione simile. Similmente agli antivirus questa contromisura deve essere costantemente aggiornata in quanto cambiare anche di pochissimo il pattern di attacco potrebbe renderlo non rilevabile.
- **Anomaly Based:** Questo approccio tenta di definire il comportamento normale e poi rilevare i modelli di comportamento al di fuori della gamma normale. C'è una fase di addestramento, in cui il sistema impara la gamma di comportamenti normali, seguita dalla fase di rilevamento vera e propria.
- **Code Analysis:** Le tecniche di analisi del codice coinvolgono l'uso di una suite di test per rilevare le vulnerabilità. La suite di test è progettata per generare una vasta gamma di attacchi e valutare la risposta del sistema.

Run-time Prevention: Queste tecniche controllano le query in fase di esecuzione per vedere se risultano conformi ad un modello di query attese. Sono disponibili vari strumenti automatizzati per questo scopo.



Capitolo 8

Blockchain & Bitcoin

8.1 Introduzione

8.1.1 Blockchain

La *blockchain* (letteralmente “catena di blocchi”) è una struttura dati **condivisa** e “**immutable**”. È definita come un *registro digitale* le cui voci sono raggruppate in “**blocchi**”, concatenati in ordine cronologico, e la cui integrità è garantita dall’uso della **crittografia**. Sebbene la sua dimensione sia destinata a crescere nel tempo, è immutabile in quanto, di norma, il suo contenuto una volta scritto non è più né modificabile né eliminabile, a meno di non invalidare l’intera struttura. Tali tecnologie sono incluse nella più ampia famiglia delle “*Distributed Ledger*”, ossia sistemi che si basano su un registro distribuito, che può essere letto e modificato da più nodi di una rete. Non è richiesto che i nodi coinvolti conoscano l’identità reciproca o si fidino l’uno dell’altro. Difatti, per garantire la coerenza tra le varie copie, l’aggiunta di un nuovo blocco è globalmente regolata da un protocollo condiviso. Una volta autorizzata l’aggiunta del nuovo blocco, ogni nodo aggiorna la propria copia privata: la natura stessa della struttura dati garantisce l’assenza di una sua manipolazione futura. Le caratteristiche che accomunano i sistemi sviluppati con le tecnologie Blockchain e Distributed Ledger sono digitalizzazione dei dati, **decentralizzazione**, **disintermediazione**, **tracciabilità dei trasferimenti**, **trasparenza**/verificabilità, **immutabilità** del registro e programmabilità dei trasferimenti. La natura distribuita e il modello cooperativo rendono robusto e sicuro il processo di validazione, ma presentano tempi non trascurabili, dovuti in gran parte al processo di validazione dei blocchi e alla sincronizzazione delle reti. L’autenticazione avviene tramite collaborazione di massa ed è alimentata da interessi collettivi. L’utilizzo di questa tecnologia consente anche di superare il problema dell’infinita ripro-



ducibilità di un bene digitale e della doppia spesa, senza l'utilizzo di un server centrale o di un'autorità. Talvolta risulta possibile che alcuni nodi della rete producano simultaneamente più blocchi “concorrenti” (ossia collegati a uno stesso blocco già esistente, ma diversi tra loro nel contenuto): ciò dà origine a una biforcazione (fork) nella catena. Il protocollo di aggiornamento specifica quale regola i nodi debbano adottare per selezionare il blocco da accettare, tra quelli concorrenti. I blocchi non selezionati per l'inclusione nella catena sono chiamati “*blocchi orfani*”.

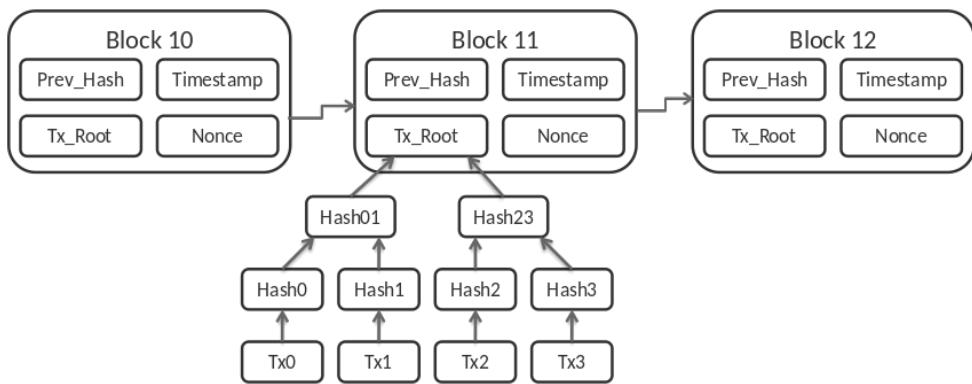


Figura 8.1: Esempio di blocchi e del loro contenuto all'interno della Blockchain.

8.1.2 Bitcoin

Il bitcoin si comporta più come “*oro*” che come “moneta” perché il valore cambia nel tempo. Più che proprietà o possesso di bitcoin/monete si parla di *diritto di spesa*, che si ottiene conoscendo una determinata chiave di cifratura. Si possono inviare e ricevere bitcoin sfruttando una exchange platform che consente l'invio di denaro utilizzando un sistema a doppia chiave. In sostanza, il passaggio di possesso di bitcoin avviene cambiando l'utente che è a conoscenza della chiave che è in grado di decriptarli. Questo passaggio di chiavi viene fatto dall'exchange platform. Quindi si genera una coppia chiave pubblica-privata, si divulgla la pubblica e si ottengono i bitcoin sopra che poi con la chiave privata potranno essere spesi o ceduti ad un altro utente.

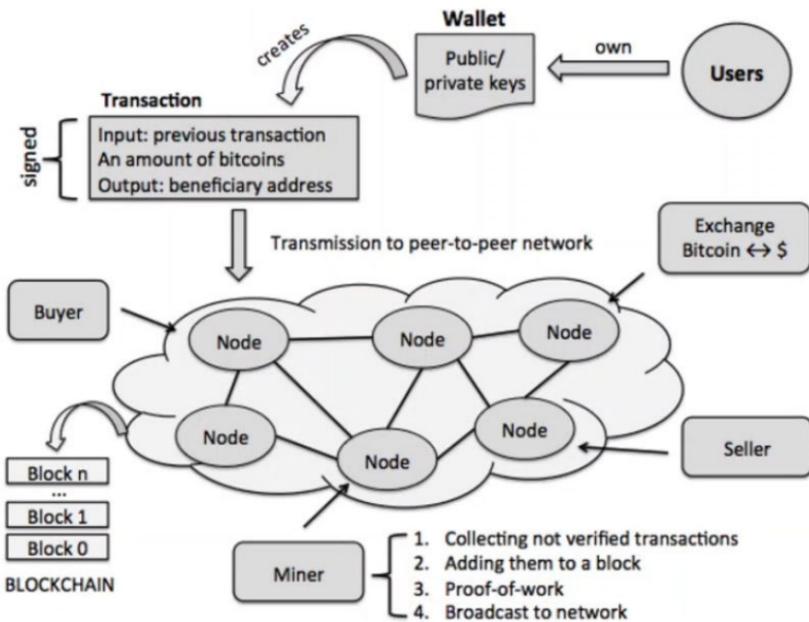


Figura 8.2: Immagine raffigurante l'ecosistema Bitcoin.

Nella precedente immagine possiamo distinguere:

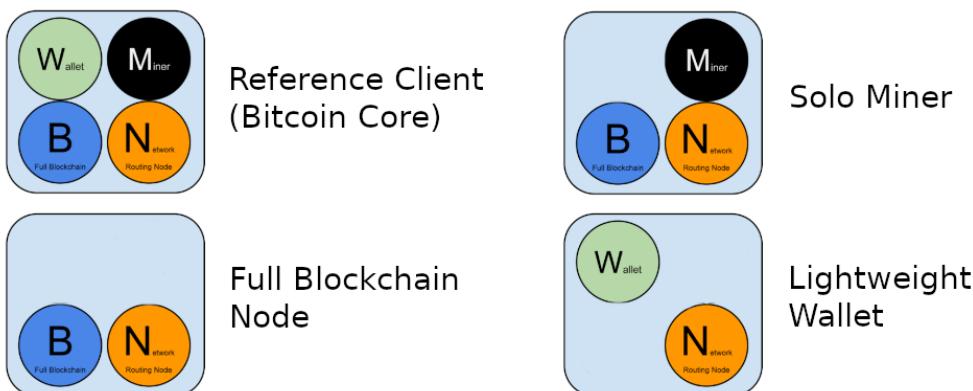
- **Exchange:** trasformazione bitcoin e moneta,
- **Nodi** per registrare transazioni,
- Gli **utenti** che voglio partecipare all'ecosistema devo essere dotati di un **Wallet**,
- Un **wallet** è un software che permette di gestire in maniera facile le coppie di chiavi private-pubbliche, che servono per ricevere o dare diritti di spesa/transazioni,
- Le **transazioni** vengono salvate sui database distribuiti e vengono replicate su tutti i nodi della rete.

8.1.2.1 Bitcoin Core

Il software Bitcoin Core¹ (evoluzione Bitcoin-Qt) può essere scaricato come qualsiasi altro programma sul nostro computer. Bitcoin Core implementa tutti gli aspetti della rete Bitcoin, quindi scaricarlo ti renderà un nodo completo della rete, ciò include una copia esatta e completa di tutte le operazioni che sono state effettuate con Bitcoin dal suo lancio nel 2009. Naturalmente sarà costantemente aggiornato, quindi la richiesta di spazio di archiviazione disponibile sul disco rigido sarà di almeno 400 GB.

8.1.2.2 Bitcoin Actors

Possiamo distinguere diverse tipologie di nodi in base alle funzioni che implementano:



- *Wallet* (●): mantiene le chiavi pubbliche e private.
- *Routing* (○): quando riceve una transazione può fare il broadcast in modo da comunicarla a tutti.
- *Full blockchain* (●): tiene il database di tutte le transazioni fatte e/o può creare transazioni
- *Miner* (●): in genere le transazioni da inserire su un blocco sono tantissime, il miner ha il compito di selezionarle e metterle in un blocco (nelle blockchain si può inserire solo un blocco di transazioni per volta, e non una transazione alla volta), per poi comunicare le operazioni portate a termine agli altri nodi in modo che tutti aggiornino le informazioni.

¹In inglese Bitcoin Core ☺

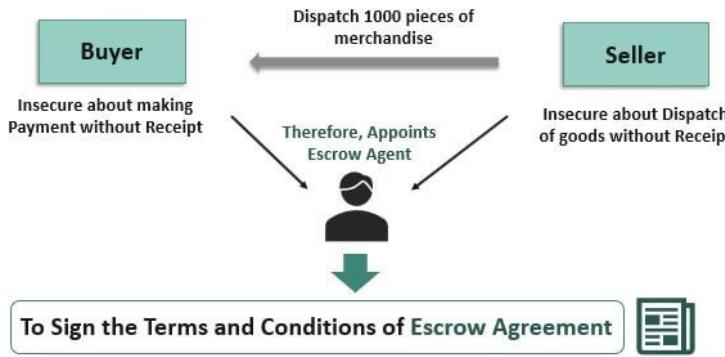
8.1.2.3 Miner

Tutte le transazioni che devono essere ancora inserite nella blockchain richiedono di essere validate dai miner e mentre sono in attesa risiedono in un pool generale. Sono i **Miner** a decidere quale transazione mettere in un blocco. In generale vengono inserite prima le transazioni con importo maggiore o quelle più vecchie, ma è possibile pagare una piccola commissione al miner per far salire la propria transazione di priorità. Inoltre come compenso aggiuntivo nel protocollo è stabilito che quando i miner inseriscono nuove transazioni ricevono un compenso in nuovi bitcoin (creati dal nulla). Questo compenso in bitcoin tuttavia è ideato per diminuire nel tempo fino a raggiungere eventualmente uno zero in base al rapporto compenso/numero di blocchi inseriti. Tutto ciò avviene affinché i miner si concentrino principalmente sulle transazioni (dove riceveranno le commissioni degli utenti) e non sull'inserimento di nuove monete/blocchi. Se così non fosse non sarebbero incentivati e l'ecosistema bitcoin crollerebbe. Per regolare queste operazioni di validazione delle transazioni esistono vari protocolli, i più comuni nell'ambito delle criptovalute sono il **Proof of Work** e il **Proof of Stake**.

PoW: nel proof of work la ricompensa non viene data a tutti i nodi ma al primo che riesce a minare un blocco (risolvendo un problema crittografico) e ad aggiungerlo alla blockchain. Per risolvere il problema il miner dovrà prendere un insieme di transazioni non verificate, mettere prima di queste l'hash dell'ultimo blocco presente nella blockchain e inserire in fondo un *nonce*, ovvero un numero randomico. Dovrà poi calcolare l'hash dell'intero blocco, cambiando di volta in volta il nonce, finché non otterrà un hash che inizi con tanti 0 quanti sono quelli richiesti dal sistema della blockchain in quel momento, `00000xxxxxxxxxxxx`. Tale quantità di 0 viene infatti incrementata e decrementata dal sistema affinché il tempo medio di mining di un blocco rimanga intorno ai 10 minuti. Una volta verificata la correttezza dell'hash dagli altri nodi l'operazione sarà approvata. Infine, prima di aggiungere definitivamente il blocco, viene inserita una transazione fittizia in più, chiamata **Coinbase**, dove viene registrato il fatto che il miner riceve il suo compenso di nuovi bitcoin.

8.1.2.4 Escrow Contracts

Questi sono un particolare tipo di smart contract basato su Bitcoin in cui per trasferire una somma di denaro da A a B ci si appoggia ad una terza entità in cui viene depositato il compenso del contratto che verrà rilasciato solo quando le condizioni definite in precedenza verranno soddisfatte.



8.2 Sicurezza e Blockchain

8.2.1 Double Spending

È un potenziale attacco in cui un bitcoin viene inviato a 2 persone contemporaneamente, benché non sia realizzabile nella blockchain di bitcoin poiché è intrinsecamente resistente fa da base per molti altri attacchi e dunque spiegheremmo brevemente come funziona. Un miner A invia denaro a due utenti (negozi) B e C contemporaneamente, di norma in bitcoin viene approvata la prima transazione (dato che ci si basa sull'hash e sul timestamp) ma supponiamo che entrambe vengano approvate in contemporanea su due blocchi diversi, allora entrambe verranno viste come transazioni valide e ci sarà una possibile fork della blockchain. In bitcoin però prima o poi una delle due blockchain risulterà essere più lunga dell'altra e verrà presa quella come blockchain valida e perciò una delle due transazioni verrà rifiutata ma nel mentre il miner ha già ricevuto gli oggetti comprati sui negozi B e C. La soluzione a tale problema è che i negozianti, prima di inviare le proprie merci, devono aspettare che la propria transazione risulti assere aggiunta in maniera definitiva alla blockchain e ciò avviene dopo il mining di altri 6 blocchi rispetto a quello dove è tale transazione (circa 1 ora di attesa).

8.2.1.1 51% Attack

Un possibile attacco che concettualmente è simile a quello del double spending è quello del 51% in cui un utente che dispone di un ampia porzione di potenza di calcolo della rete può tentare di manipolare la blockchain portando avanti in segreto una copia alterata della stessa. Se il miner che sta portando avanti questo attacco riesce a generare una blockchain più lunga di quella vera allora per la regola della catena più lunga, divulgandola la farà accettare dalla rete e diventerà dunque la vera blockchain.



Per imporre i suoi blocchi, un'organizzazione dovrebbe controllare il 51% dei nodi (è detto 51% attack), per avere potenza di calcolo sufficiente a produrre (con buona probabilità) catene di blocchi lunghe prima di tutti gli altri nodi, ciò è altamente improbabile. Attualmente non è possibile che questo attacco abbia successo perché nessuno dispone di una potenza di calcolo così elevata ma ci sono comunque delle *mining pool* che accrescono quotidianamente la propria hash power (la più grande ha circa il 30%).

8.2.2 Wallet attack

Ti vengono rubate le credenziali di accesso al wallet o le chiavi private .

8.2.3 Transaction Malleability

Gli attacchi di malleability consistono nello sfruttare la manipolabilità dell'id delle transazioni che viene calcolato con un hash. Andando a cambiare l'id della transazione si verificherà un effetto a cascata causato dall'hash e dunque il mittente della transazione non la riconoscerà più come propria anche se c'è la possibilità che i soldi vengano comunque inviati. Così facendo l'utente malevolo ha un pretesto per richiedere nuovamente i soldi alla vittima affermando che non ha mai inviato i soldi.

8.2.4 Sybil Attack

Un Sybil attack è un attacco in cui un utente o un gruppo di utenti prende il controllo della rete creando un elevato numero di nuovi account e nodi, avendo così la maggioranza dei nodi. Questo consentirebbe di poter far approvare dei blocchi con transazioni fraudolente. Tutto ciò risulta essere impossibile dato che viene mitigato dalla PoW.

