Capitolo 1

Basi della sicurezza

1.1 Introduzione

La sicurezza ha il compito di:

- **Individuare quali sono le risorse da progettere** -> hardware, software, dati, collegamenti e apparati di rete (se il mio HW è protetto ma non poso comunicare all'esterno, sono isolato). La protezione delle risorse si rifà a cinque aspetti fondamentali:

  1. Confidenzialità: nessun utente deve poter ottenere o dedurre dal sistema info che non è autorizzato a conoscere (riservatezza + privacy);
  2. Integrità: deve essere impedita l'alterazione diretta e indiretta da parte di utenti/processi non autorizzari oppure in seguiro a eventi accidentali (bug software) (integrità dati + integrità sistema);
  3. Disponibilità: specifici dati devono essere forniti solo agli utenti che hanno diritto di accedere, nei tempi e nei modi previsti;
  4. Autenticità: le info non devono essere state manipolate;
  5. Tracciabilità: ogni azione deve essere tracciabile, in modo da supportare la non ripudiabilità e l'isolamento della responsabilità.

- **Individuare i modi in cui le risorse sono minacciate** -> Le minacce possono compormettere le propietà di confidenzialità (es: lettura non autorizzata, copia SW non autorizzata), integrità (es: file modificati, eseguibile modificato) e disponibilità (es: file cancellati, eseguibili cancellati). Una minaccia è una possibile violazione della sicurezza. La violazione effettiva è un attacco. Gli attacchi possono essere:

  1. Attivi: modifica del sistema/risorse;
  2. Passivi: acquisizione delle info senza modificare le risorse;
  3. Interni: iniziati da un'entità interna al sistema
  4. Esterni: effettuati da entità esterne, solitamente tramite la rete.

  Gli attacchi possono essere classificati in:

  1. Disclosure: accesso non autorizzato ai dati;
  2. Deception: accettazione di dati falsi;
  3. Disruption: interruzione o prevenzione di operazioni corrette;
  4. Usurpation: controllo non autorizzato di parti del sistema.

- **Individuare i modi per proteggere le risorse** -> non esiste un modo unico per proteggere le risorse, in quanto i modi cambiano nel tempo (sistemi sempre più complessi, condizioni impreviste).

  Nella progettazione dei sistemi è necessario tenere in considerazione i possibili attacchi. Le soluzioni per la sicurezza possono essere contro-intuitive. I meccanismi di sicurezza possono essere usati sia a livello fisico sia a livello logico.

  Nonostante non esiste un modo unico per proteggere i dati, esistono delle "best practices" per farlo. Le principali sono:

  - I meccanismi di sicurezza devono essere semplici ed efficaci (aspetto economico dei meccanismi);

- Si specificano dei comportamenti sicuri, mentre tutti gli altri comportamenti non specificati devono rientrare in quello di default (fail-safe default)
- Se il codice è open-source, la comunità può lavorare sulla sicurezza (progettazione aperta);
- Ogni operazione deve essere tracciabile (tracciabilità delle operazioni);
- Differenziare i privilegi in base al tipo di utente (separazione dei privilegi);
- Isolare i sottosistemi per evitaare che l'attacco si propaghi agli altri componenti (isolamento dei sottosistemi);
- Modularità del sistema.

Le "best practices" sono seguite per realizzare una politica di sicurezza, ovvero un'insieme di specifiche che determinano cosa si può e cosa non si può fare (alto livello). Le politiche di sicurezza sono realizzate tramite meccanismi. Un meccanismo può occuparsi di:

- Prevenire un attacco, come la crittografia;
- Scoprire un attacco, come un monitoraggio sui paccehtti;
- Recuperare da un attacco, come isolare il sistema in caso di attacco.

I meccanismi possono essere implementati a basso livello, dove sono più semplici ma dimostrabilmente corretti, o ad alto livello, dove sono più complesso e più difficili da dimostrare come corretti.
I principali meccanismi sono: crittografia, firma digitale, autenticazione e controllo degli accessi, rilevamento degli eventi, gestione degli audit, recovery.

Come ottenere un sistema sicure In generale per ottenere un sistema sicuro devo: descrivere il funzionamento desiderato dal sistema, tradurre le specifiche in componenti che le implementano, creare un sistema che rispetta le specifiche. Ovviamento devo verificare costantemente la sicurezza dell'implementazione.

Considerazione implementative Prima di progettare un sistema, è necessario effettuare delle considerazioni sull'implementazione della sicurezza. Queste considerazioni possono essere: analisi costi-benefici della sicurezza, analisi della probabilità di subire un attacco e dei possibili danni, aspetti legali, problemi organizzativi.

1.2 Cenni di crittografia

La crittografia è la scienza che si occupa di proteggere l'info rendendola sicura, in modo che l'utente non autorizzato che ne entra in possesso non sia in grado di comprenderla.
La crittoanalisi è invece la scienza che cerca di aggirare o superare le protezioni crittografiche, accedendo alle info protette.
Un algoritmo crittografico è una funzione che prende in ingresso un messaggio e un parametro detto **chiave** e produce un messaggio trasformato. L'algoritmo può occuparsi della cifratura o della decifratura.
Gli algoritmi crittografici sono di due categorie:

- A chiave *simmetrica*: le chiavi di cifratura e decifratura sono uguali;
- A chiave *asimmetrica*: vengono usate due chiavi differenti, una chiave è pubblica, l'altra è privata.

Ogni algoritmo crittografico deve essere robusto, vale a dire:

- Deve essere difficile ottenere il testo in chiaro senza chiave da quello cifrato;
- Dato un testo cifrato e uno in chiaro, deve essere difficile ottenere la chiave di cifratura.

Bisogna tenere sempre a mente che **nessun algoritmo crittografico è assolutamente sicuro**, quindi un algoritmo di dice *computazionalmente sicuro* se il costo necessario a violarlo è superiore a quello dell'informazione contenuta, oppure il tempo necessario a violarlo è superiore al tempo di vita dell'informazione.
In ogni caso, per analizzare un algoritmo crittografico, bisogna mantenere presente che la segretezza deve risiedere nella chiave, non nella struttura dell'algoritmo.

Attacco a forza bruta La crittoanalisi tenta di ricostruire il testo in chiaro senza conoscere la chiave di decifratura. L'attacco più banale è quello a forza bruta, ovvero tentare di decifrare il messaggio provando tutte le chiavi possibili (ovviamente devo avere delle info sul formato del testo in chiaro, per sapere se ho trovato al chiave corretta).

Principio di Kerckhoff Secondo questo principio, la robustezza deve stare nella chiave (si suppone l'algoritmo noto).

1.2.1 Crittografia a chiave simmetrica

La crittografia a chiave simmetrica utilizza una chiave condivisa e gli stessi algoritmi per cifrare e decifrare le informazioni (ovviamente la chiave deve essere scambiata su canali sicuri).
Esempi ci cifratura a chiave simmestrica sono:

- Cifratura di Cesare: si basa sullo shift dell'alfabeto, la chiave è il numero di posizioni shiftate;
- Cifratura monoalfabeta: si basa sulla permutazione dell'alfabeto (riordino casuale), la chiave è il nuovo alfabeto ottenuto;
- Cifratura a blocchi: dati k bit, i possibili 2^k input vengono permutati. Nello specifico, divido l'input in blocchi, associando ad ognuno una permutazione differente, ricodtruisco l'input e lo permuto nuovamente. Il processo può essere reiterato n volte prima di ritornaree il dato.

Nei primi due casi è facile ottenere la chiave perchè si può procedere con l'analisi delle frequenze, ovvero l'analisi di quanto spesso in un testo si presenta una stessa sillaba.Nel terzo caso l'analisi delle frequenza non è possibile perchè ogni lettera ha più cifrature.
Esempi di agloritmi a chiave simmetrica sono:

- DES: chiavi a 56 bit, ormai obsoleto;
- Triplo-DES: un DES applicato 3 volte con chiavi diverse (di lunghezza 112 o 168 bit);
- AES: usa chiavi a 128, 192 o 256 bit.

Tutti questi algoritmi sono soggetti al problema della distribuzione delle chiavi.

1.2.2 Crittografia a chiave asimmetrica

In questo tipo di crittografia ogni utente ha una coppia di chiavi, una **pubblica**, che viene resa nota, e una **privata**, che viene mantenuta segreta. L'idea è che il messaggio venga cifrato con la chiave pubblica del destinatario, che potrà poi decifrarlo con la sua chiave privata.
L'uso di algoritmi a chiave asimmetrica comporta i seguenti vantaggi:

- Non è più necessario scambiarsi chiavi;
- La stessa chiave pubblica può essere usata da più utenti.

I requisiti principali di un sistema crittografico di questo tipo sono:

- Deve essere semplice la generazione delle chiavi;
- Devono essere semplici le operazioni di cifratura/decifratura quando si hanno le relative chiave;
- Deve essere computazionalmente impraticabile ricavare la chiave privata da quella pubblica;
- Deve essere computazionalmente impraticabile ricavare il testo in chiaro avendo il testo cifrato e la chiave pubblica.

Un esempio di algoritmo simmetrico è RSA, che si basa sulla difficoltà di scomporre un numero in fattori primi. Le chiavi usate da RSA hanno le dimensioni di 2^{10} bit.

Gli algoritmi asimmetrici richiedono molte più risorse degli algoritmi a chiave simmetrica (sono infatti anche molto più lenti), e vengono usati per scambiarsi una chiave di sessione che verrà poi usata da un algoritmo simmetrico sicuro, computazionalmente più efficiente.

1.3 Integrità

Lo scopo storico della crittografia è quello di garantire la privacy, ossia come garantire che un'informazione ricevuta provenga effettivamente dall'utente che ci si aspetta l'abbia mandata.

1.3.1 Funzioni Hash