Architetture sicure

H1 Architettura per la sicurezza

> H2 Insieme dei principi di alto livello e delle decisioni per la progettazione di sistemi da ritenersi sicuri

È un framework per progettare SW sicuro, che si basa su 4 principi:

- **protect**
- **deter**
- **detect**
- **react**

Una **architettura per la sicurezza** è diversa dalla **policy di sicurezza**. Quest'ultima infatti viene stabilita a priori ed è in base ad essa che viene definita l'architettura.

Vediamo ora i principi su cui costruire una architettura sicura.

Principi

H3 Domande iniziali

H2

> Inizia facendo domande e decidi *quale livello di sicurezza è sufficiente*

Domande fra le seguenti tipologie:

- sulle **vulnerabilità**
  - punti deboli
  - cosa difendere
  - da chi difendersi
  - . . .
- sulle **risorse**

    - librerie
    - standard da mantenere
    - ...
- sul **SW**
    - catena di trust
    - accessibilità
    - ...
- sui **goal**
    - impatto
    - qualità
    - ...

Progettare con in mente il nemico

> Detto anche **adversary principle**: progetta il SW come se il nemico più astuto lo attaccherà

Conoscere e rispettare la *chain of trust*

- Non invocare programmi non fidati da programmi fidati
- Un programma NON deve delegare l'*autorità* di fare un'azione senza delegare anche la *responsabilità* di controllare se l'azione è appropriata

Privilegi adeguati

Spesso conviene seguire il **principio del minimo privilegio**.

Test delle azioni contro la *policy*

> **Principio di mediazione completa:** testare in ogni passo e stato del SW ogni azione che è possibile tentare per violare la *policy*

Mantenere lo stato minimale

> **Principio del *minimal retained state:*** un programma deve tenere in memoria lo stato minimale, cioè il minor numero di informazioni possibili

Garantire un livello adeguato di *fault tolerance*

Si basa sull'identificazione delle *funzionalità critiche*, secondo l'uso delle "3 R":

- **Resistance:** la capacità di impedire un attacco
- **Recognition:** la capacità di riconoscere attacchi e la misura del danno
- **Recovery:** la capacità di fornire servizi durante un attacco e ripristinare tutti i servizi successivamente ad esso

Applicare la *graceful degradation*

> **Principio di *graceful degradation:*** quando si verifica un guasto, il sistema non si blocca ma continua ad operare con funzionalità ridotte

Applicare il principio di *fault safely*

> **Principio di *fault safely:*** in caso di fallimento, un sistema deve terminare in una configurazione sicura

Misure di sicurezza adeguate all'utente

> **Principio di *accettabilità psicologica:*** è importante usare modelli mentali che rendano accettabile all'utente seguire le misure di sicurezza

Se l'utente è irritato da come le misure di sicurezza intervengono nella sua esperienza utente, eviterà di usarle, compromettendo la sicurezza.

Possibilità di ricostruire gli eventi

> **Principio di *auditability:*** deve essere possibile ricostruire la sequenza di eventi

Modularizzazione e semplicità

> **Principio di *modularizzazione:***
>
> - suddividere la progettazione in moduli
> - definire con precisioni i punti di interfaccia tra moduli
> - limitare i privilegi e le risorse a moduli che veramente ne necessitano

Inoltre è utile realizzare sistemi che siano *il più **semplici** possibili.*