# principi di progetto di politiche e meccanismi di sicurezza

# minimalità dei diritti

- ad un soggetto devono essere concessi solo i diritti necessari a eseguire i suoi compiti
  - diritti non necessari non vengono concessi
  - i diritti sono dati solo per il tempo necessario
  - i diritti devono essere dati in base al “ruolo” e non all’identità (vedi Role Based Access Control - RBAC)

# default sicuri

- le configurazioni di default devono essere sicure
  - il default deve essere di negare l'accesso
    - politica chiusa
  - un accesso negato lascia sicuramente il sistema in uno stato sicuro
    - forse un utente sarà scontento? l'amministratore sarà avvertito dallo stesso utente, se questi è stato privato di un diritto
  - un accesso concesso può rendere il sistema insicuro quindi l'amministratore deve configurarlo esplicitamente
    - un utente contento per avere un diritto in più non avvertirà l'amministratore per farselo togliere

# semplicità

- meccanismi e politiche di sicurezza devono essere più semplici possibili
  - interfacce ed interazioni semplici tra pochi elementi
  - se il sistema ha pochi elementi vi sono pochi punti in cui si possono commettere errori
    - meno bugs
    - meno errori di configurazione
  - un sistema semplice è più facile da capire e aggiustare in caso di problemi
  - “Ciò che non c’è non si può rompere” – H. Ford

# progetto aperto

- la sicurezza non deve dipendere dalla segretezza del progetto o dell'implementazione
  - no "Security through obscurity"
  - la sicurezza non risiede nella segretezza dei meccanismi ma nella segretezza di passwords e chiavi crittografiche (detti per l'appunto "segreti")
  - non significa che il codice deve essere pubblico
    - il codice PUO' essere pubblico, se non contiene bugs, senza inficiare la sicurezza
- se progetto, codice o politica sono fatti male è meglio tenerli nascosti, la sicurezza, però, non può essere basata su questo
  - approccio ammissibile come rimedio temporaneo non essendo sicuri della correttezza del progetto, del codice o della politica in questione

# progetto aperto

- principio applicato spesso ai prodotti con larga diffusione
  - un prodotto con larga diffusione giova del controllo dei suoi clienti
  - i clienti ritengono che un progetto aperto sia più affidabile quindi più appetibile commercialmente
- le politiche di sicurezza e i particolari realizzativi per le singole organizzazioni spesso vengono mantenuti segreti
  - la politica o i meccanismi di una organizzazione sono difficilmente esenti da bugs o falle di sicurezza
  - la pubblicazione faciliterebbe gli attacchi ma anche la verifica

# isolamento (o confinamento)

- minimizzare la condivisione di risorse tra soggetti (servizi, utenti, processi)
  - in particolare se i soggetti hanno “criticità” differenti, cioè appartengono ad una diversa **classe di sicurezza**
- la condivisione di risorse …
  - …permette attacchi da un servizio/utente ad un altro
  - …rende entrambi i servizi/utenti vulnerabili ad un fallimento della risorsa

# mediazione completa

- effettuare il controllo ad ogni accesso
- spesso nei sistemi il controllo è effettuato solo una volta all'inizio della "transazione"
  - cambiamenti di permessi non hanno effetto su transazioni già iniziate (es. file già aperti)
  - es. nei sistemi UNIX il controllo di accesso ai file è effettuato solo dalla system call open e non dalle operazioni successive

# defence in depth

- richiedere il consenso di più entità di controllo per ottenere l'accesso
  - più meccanismi di difesa assieme sono più difficili da forzare
  - distribuzione di parti di uno stesso segreto su più host
    - una parte di segreto è inutile quindi l'accesso ad un singolo host non dà vantaggi sostanziali per l'accesso al segreto

# usabilità (accettabilità psicologica)

- meccanismi e politiche di sicurezza non devono aggiungere difficoltà all'accesso alle risorse da parte degli utenti
  - altrimenti gli utenti si rifiuteranno di usare il sistema
  - o cercheranno di aggirare il le limitazioni per poter svolgere agevolmente il loro lavoro
  - o passano alla concorrenza
- è un obiettivo difficile

# eterogeneità

- usare sistemi eterogenei
  - servizi su sistemi diversi (es. windows e linux) hanno meno probabilità di essere attaccati dallo stesso hacker perché gli exploit sono diversi
  - vedi “defence in depth” e isolamento
- raggiungere alta eterogeneità è problematico
  - trovare sistemi diversi è difficile
    - quanti sistemi operativi conosci? quanti web server?
  - difficoltà di gestione
    - gli amministratori rifiutano di gestire sistemi molto eterogenei (vedi accettabilità psicologica)

# compromesso

- questi principi sono spesso in conflitto tra loro
  - tipicamente usabilità e della semplicità sono in conflitto con tutti gli altri
- un buon progetto prevede il giusto compromesso tra
  - questi principi
  - vincoli di budget
  - altri vincoli (es. tecnologici)