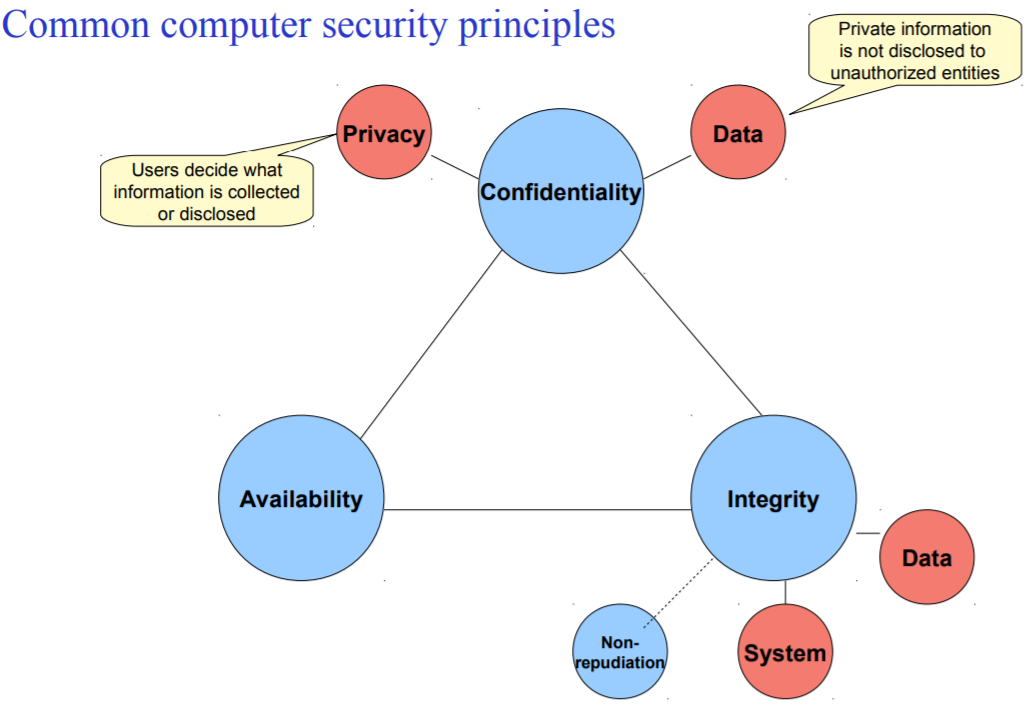
**Security policy** è una dichiarazione di ciò che è, e cosa non lo è, consentito.

**Security mechanism** è un metodo, uno strumento o una procedura per l'applicazione di una politica di sicurezza (come applicare una politica)

**Security model** è un modello che rappresenta una particolare politica o una serie di politiche: i modelli in genere seguono uno o più principi (**principles**)



***Ideal security model***

**Inevitabile (Inescapable)** incapacità di rompere le politiche di sicurezza aggirando i meccanismi di controllo dell'accesso

**Invisibile (Invisible)** interazione utente e amministrativa senza interruzioni

**Fattibile (feasible)** conveniente e pratico da implementare

***Separate execution contexts***

Un sistema operativo deve utilizzare **un robusto meccanismo di protezione** per impedire di bypassare meccanismi di astrazioni dei SO, utilizzare istruzioni pericolose, chiamare le procedure senza controllo)

***Kernel mode, user mode***

Alcune istruzioni (disabilitare gli interrupt o modific tabelle di conversione degli indirizzi) non essere disponibili.

**- Privilegiato (modalità kernel)** (controllo del HW, un errore può causare arresto anomalo)

**- Non privilegiato (modalità utente)** (HW e astrazione tramite chiamte di sistema, errori influenzano solo il processo)

***Ring based security***

1 Cerchio in mezzo = 0 kernel mode (tutti i privilegi) 🡪 ultimo cerchio = 3 – User mode (meno privilegi)

**How does the CPU know it it's running privileged code? (pensare uno stack on dentro I livelli permessi)**

Le aree di memoria (segmenti, pagine) possono essere assegnate con un livello privilegiato:

- Se istruzione privilegiata eseguita da livello non privilegiato, generato un errore di protezione

• Su x86 il livello di privilegio corrente può essere ottenuto leggendo un registro CPU

**Security issues**

- Kernel viene eseguito in modalità privilegiata e imposta la memoria del processo utente come non privilegiata.

- Le chiamate di sistema vengono implementate dal kernel e quindi archiviate in un'area privilegiata della memoria per poter essere eseguite in modalità privilegiata.

Per motivi di sicurezza non possiamo saltare liberamente tra regioni di memoria con diversi livelli di privilegio:

Se possiamo passare da - privilegiata a + privilegiata (qualsiasi indirizzo) postremo bypassare protezione e magari cambiare indirizzo di ritorno

**... quindi come possiamo implementare in modo sicuro le chiamate di sistema?**

• Abbiamo bisogno di un modo per consentire il salto solo a specifici indirizzi (porte), non da nessuna parte

gli interrupt fermano l'esecuzione di un processo e danno il controllo al sistema operativo (livello + alto).

- Il software non privilegiato può generare anche interrupt, ma la routine di gestione è determinata dal kernel (cioè il codice non privilegiato non può saltare da nessuna parte, ma solo a indirizzi predefiniti)

Le applicazioni utente devono eseguire chiamate di sistema (costosa, e non possiamo eseguire solo in modalità non privilegiata)

**Quindi, perché non implementiamo la maggior parte delle funzionalità a livello di kernel?**

– Security, robustness

– “Principle of least privilege”

– Give software only the privilege it needs

– Give access only to the information and resources that are necessary for its legitimate purpose

• We should not execute everything in privileged mode

**Micro-kernel** (solo mimine funzionalità in privilegata, In genere il resto del sistema operativo si basa su server indipendenti che comunicano utilizzando il passaggio dei messaggi)

**Monolitico** (Linux, Tutto in modalità privilegiata)

**Ibrido** (Windows NT, simile a micro-kernel ma eseguita principalmente in privilegiata)

**L'autenticazione** è il processo per verificare che un utente, un dispositivo (computer, server) o un servizio sia ciò che dichiara di essere (controllo dell'identità)

**Autorizzazione** è il processo per determinare cosa può fare un'entità autenticata su un sistema o su una rete (controllo di accesso)

L’autenticazione utente permette l’accesso al sistema o risorsa, tipicamente basato su password tramite hash, token, biometrica

**Salt value** permette di combinare un valore fisso (random) con la password quindi hanno più combinazioni

**Pam** è responsabile dell’autenticazione auth, accout, password, session

**Login** 🡪 eseguita in privilegi di root, credenziali verificate traimite moduli pam, utilizzerà la chiamata di sistema setuid per modificare l'UID associato al processo della shell all'ID dell'utente

**Windows**

**Local Security Authority (LSA)** gestisce gli accessi interattivi nel sistema. Si affida a SAM per verificare le credenziali

**Windows Security Account Manager (SAM)** gestisce gli hash delle password dell'utente (per autenticare locale o remoto, hash memorizzati in hive un gruppo di chiavi)

Chiavi trasmessi con hash o chiave AES o digest

**Windows Challenge / Response (NTLM)** è un protocollo di autenticazione per reti di computer Windows e sistemi stand-alone

**LDAP (Lightweight Directory Access Protocol)** è un protocollo di rete standard progettato per interrogare i servizi di directory e accedere / modificare i dati

**Active Directory** è un servizio di directory per fornire autenticazione, criteri e altri servizi a cui è possibile accedere tramite il protocollo LDAP (login ad ogni servizio

**Kerberos** 🡪 Gli utenti effettuano il login una volta e ottengono un token che può essere utilizzato per più servizi sul nework

Tipi di controlli

**Discretionary Access Control (DAC)** 🡪 consente ai soggetti di decidere i diritti di accesso sugli oggetti di loro proprietà (in linux abbiamo il controllo di rwx|rwx|rwx (other|group|user) salvati in i-node)

**ACL (Access Control Lists)** consentono un controllo più preciso sui diritti di accesso

* Esplicito 🡪 definito dall’utente
* Ereitato 🡪ereditato dalla directory principale
* Efficace🡪combinazione di autorizzazioni esplicite ereditate