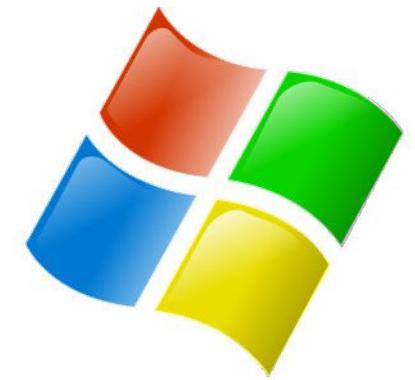




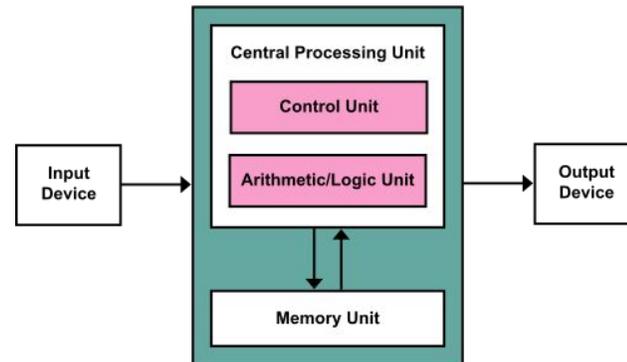
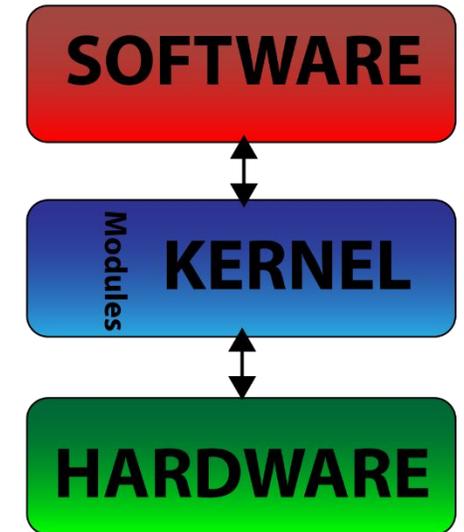
**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Corso di Sistemi Operativi



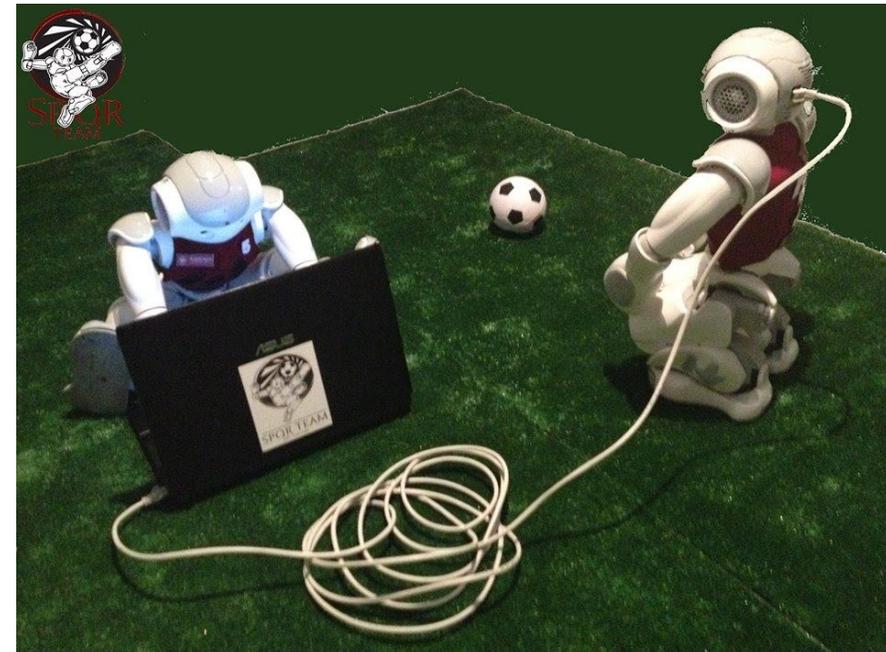
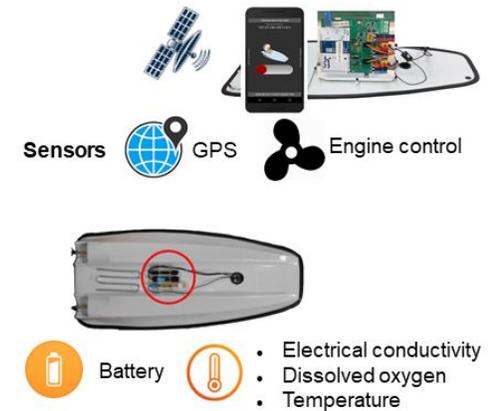
Protezione

Docente:
Domenico Daniele
Bloisi



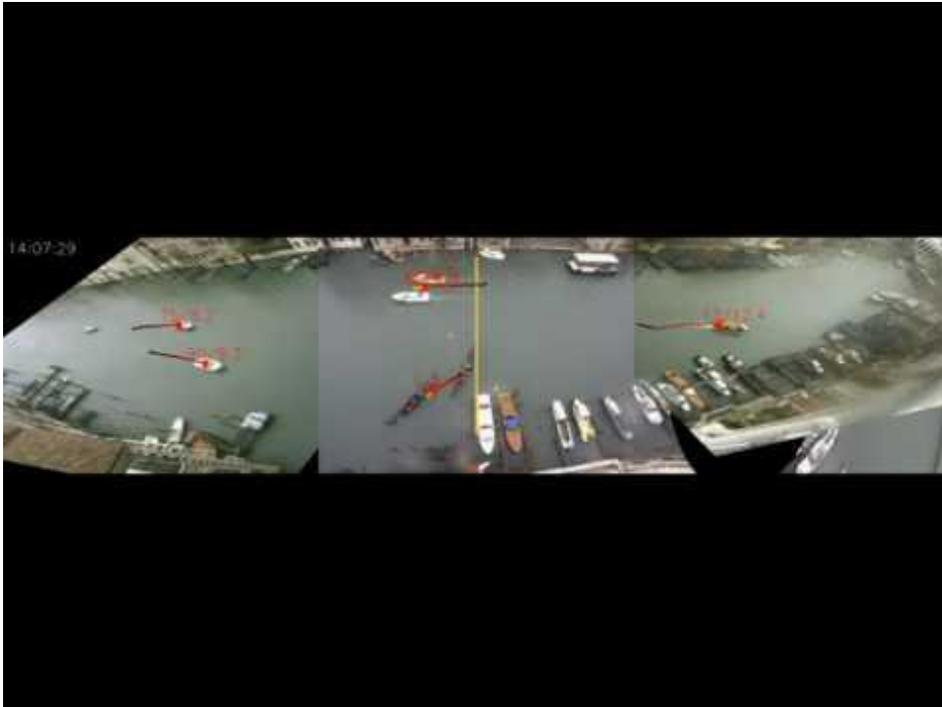
Domenico Daniele Bloisi

- Professore Associato
Dipartimento di Matematica, Informatica
ed Economia
Università degli studi della Basilicata
<http://web.unibas.it/bloisi>
- SPQR Robot Soccer Team
Dipartimento di Informatica, Automatica
e Gestionale Università degli studi di
Roma “La Sapienza”
<http://spqr.diag.uniroma1.it>



Interessi di ricerca

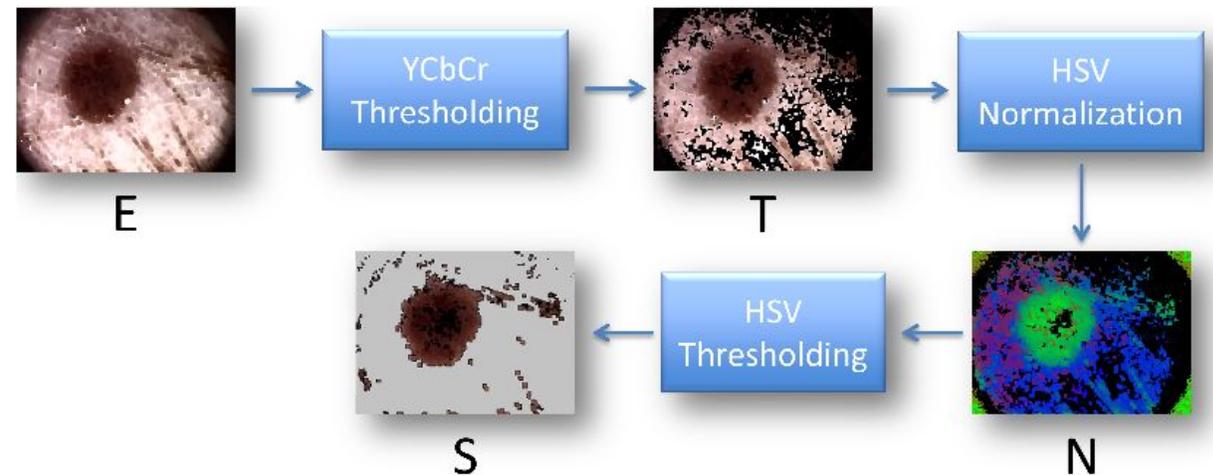
- Intelligent surveillance
- Robot vision
- Medical image analysis



https://youtu.be/9a70Ucgbi_U



<https://youtu.be/2KHNZX7UIWQ>



UNIBAS Wolves <https://sites.google.com/unibas.it/wolves>



- UNIBAS WOLVES is the robot soccer team of the University of Basilicata. Established in 2019, it is focussed on developing software for NAO soccer robots participating in RoboCup competitions.

- UNIBAS WOLVES team is twinned with SPQR Team at Sapienza University of Rome



<https://youtu.be/ji0OmkaWh20>

Informazioni sul corso

- Home page del corso:
<http://web.unibas.it/bloisi/corsi/sistemi-operativi.html>
- Docente: Domenico Daniele Bloisi
- Periodo: I semestre ottobre 2022 – gennaio 2023
 - Lunedì dalle 15:00 alle 17:00 (Aula Leonardo)
 - Martedì dalle 08:30 alle 10:30 (Aula 1)

Ricevimento

- In presenza, durante il periodo delle lezioni:
Lunedì dalle 17:00 alle 18:00
presso Edificio 3D, Il piano, stanza 15
Si invitano gli studenti a controllare regolarmente la bacheca degli avvisi per eventuali variazioni
- Tramite google Meet e al di fuori del periodo delle lezioni:
da concordare con il docente tramite email

Per prenotare un appuntamento inviare
una email a
domenico.bloisi@unibas.it



Programma – Sistemi Operativi

- Introduzione ai sistemi operativi
- Gestione dei processi
- Sincronizzazione dei processi
- Gestione della memoria centrale
- Gestione della memoria di massa
- File system
- Sicurezza e protezione

Sicurezza e protezione

La sicurezza misura la fiducia nel fatto che l'integrità di un sistema e dei suoi dati siano preservati

La protezione è l'insieme di meccanismi che controllano l'accesso di processi e utenti alle risorse di un sistema informatico

Protezione

Il ruolo della **protezione** è quello di offrire un meccanismo d'imposizione di **criteri** che controllino l'uso delle risorse.

I **criteri** vanno distinti dai **meccanismi**.

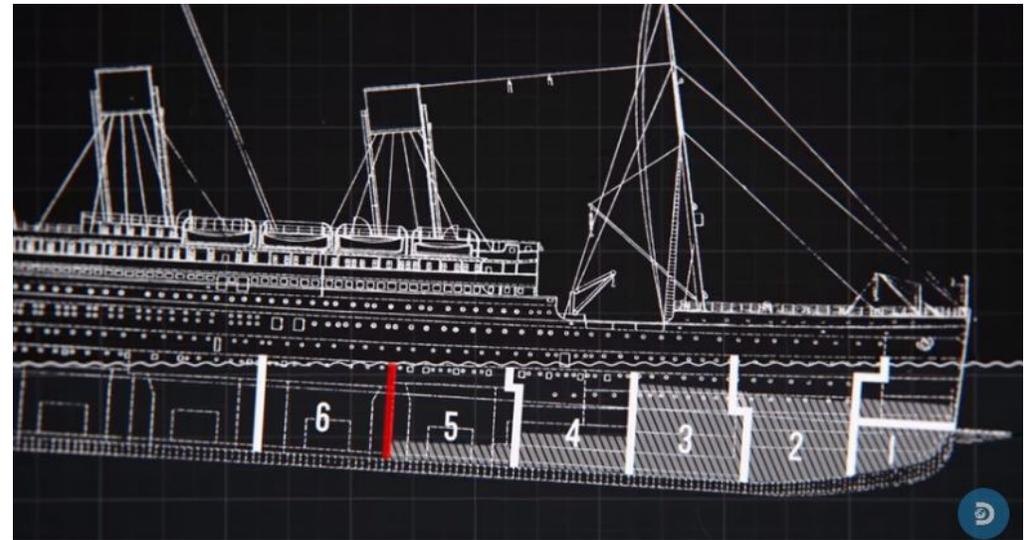
I meccanismi determinano come qualcosa si debba eseguire; i criteri decidono che cosa si debba fare.

Principi della protezione

minimo privilegio



compartimentazione



<https://youtu.be/45B6soUWBoI>

Principio del minimo privilegio

principio del minimo privilegio → i programmi, gli utenti (e anche i sistemi) devono ricevere solo i privilegi strettamente necessari per l'esecuzione dei rispettivi compiti

È un principio guida, che nel tempo ha confermato la sua importanza per la protezione

Compartimentazione

compartimentazione → processo di protezione di ogni singolo componente del sistema attraverso l'uso di autorizzazioni specifiche e restrizioni di accesso

È implementata in molte forme, dalle *zone demilitarizzate* (DMZ) a livello di rete, alla *virtualizzazione*

Anelli di protezione

Un modello di separazione dei privilegi utilizzato di frequente è quello degli **anelli di protezione**

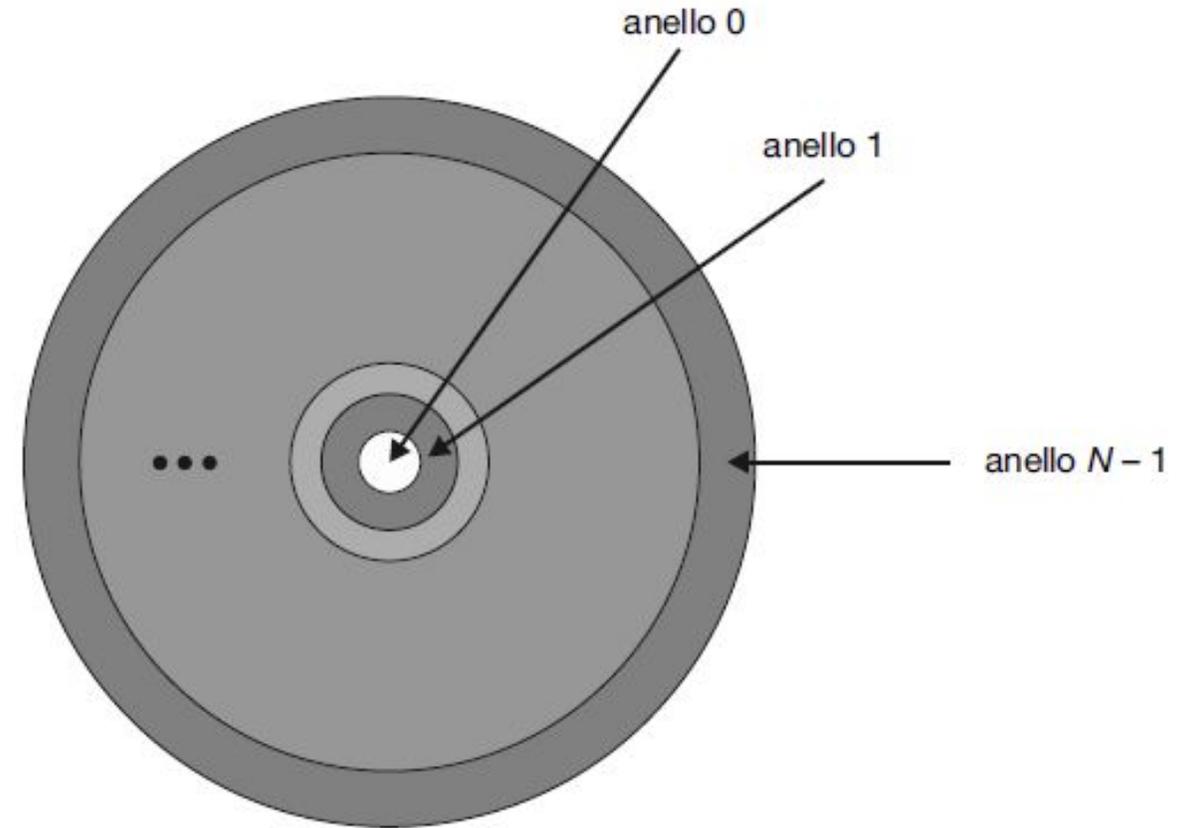


Figura 17.1 Struttura di protezione ad anelli.

TrustZone

La **TrustZone** (TZ) ha fornito un anello aggiuntivo

Android utilizza la **TrustZone** in maniera estesa a partire dalla sua versione 5.0

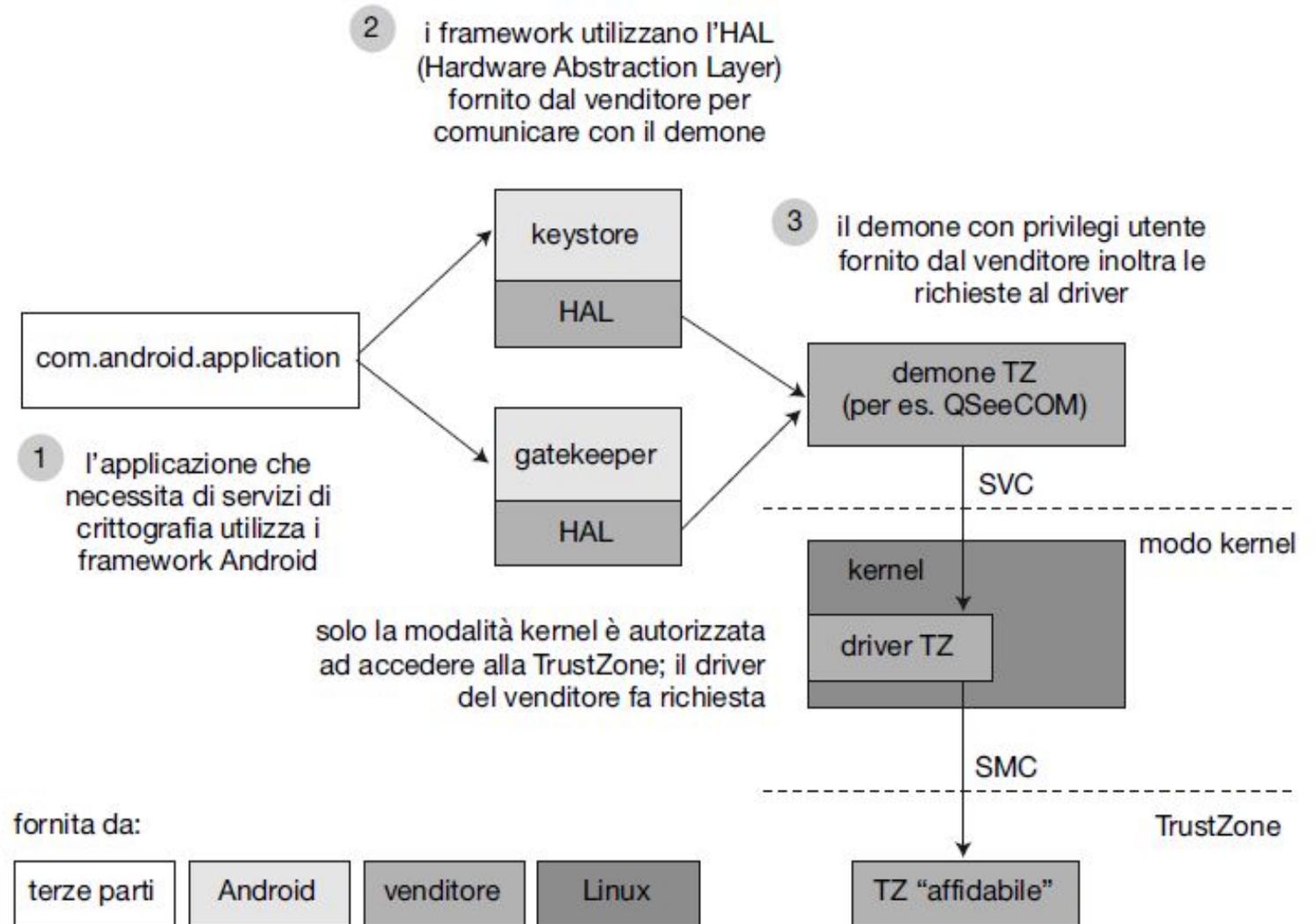


Figura 17.2 Utilizzo della TrustZone in Android.

Livelli di eccezione

Nell'architettura ARMv8 a 64 bit, ARM ha esteso il suo modello per supportare quattro livelli, denominati “**livelli di eccezione**” e numerati da EL0 a EL3

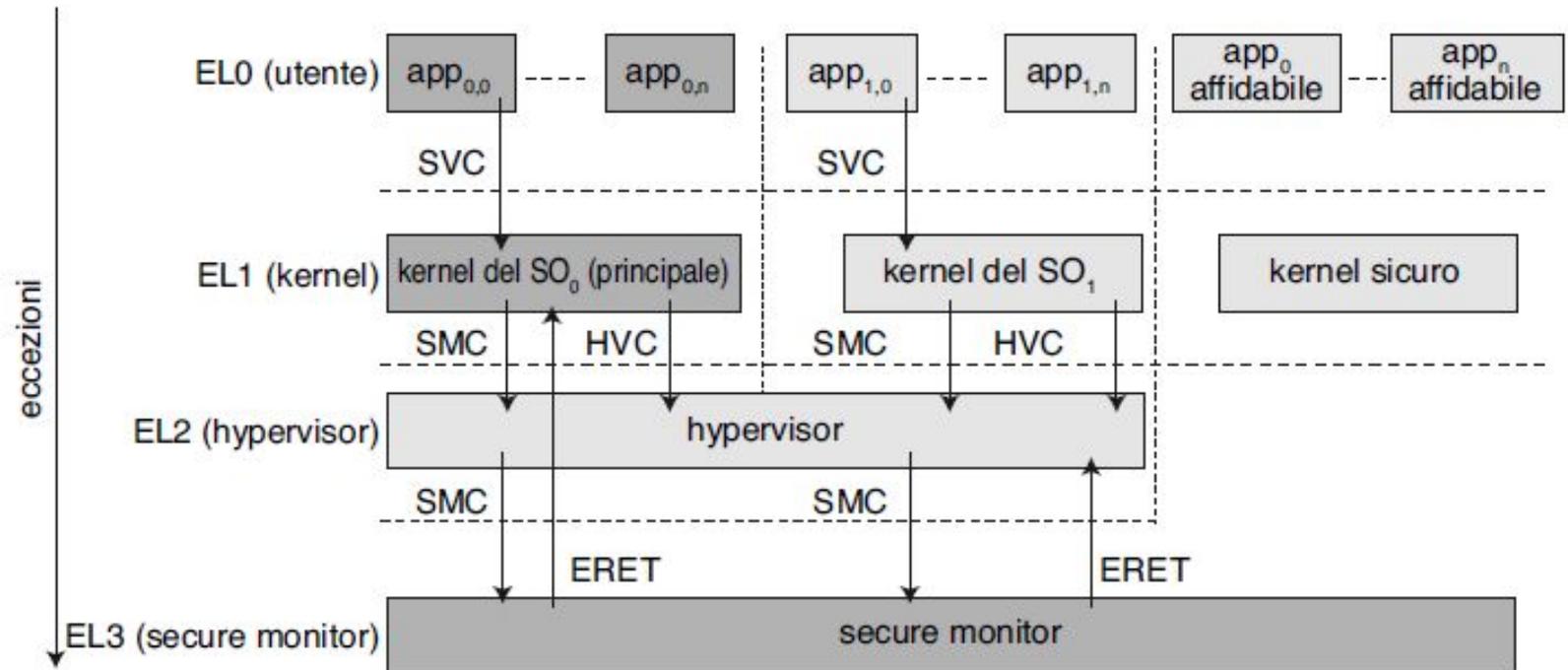


Figura 17.3 Architettura ARM.

Necessità di sapere

Principio della necessità di sapere (*need-to-know-principle*)

utile per limitare i danni che possono essere causati al sistema da un processo difettoso.

Confrontando la politica della **necessità di sapere** con quella del **privilegio minimo** si riscontra che la prima è volta alla politica adottata, mentre la seconda al meccanismo per ottenere questa politica.

Diritti di accesso

- Un **diritto d'accesso** è un permesso per eseguire un'operazione su un *oggetto*.
- Un **dominio** è un insieme di diritti d'accesso.

Domini di protezione

I processi vengono eseguiti in **domini** e possono usare tutti i diritti d'accesso del dominio per accedere agli *oggetti* e manipolarli.

Durante il suo ciclo di vita un processo può essere vincolato a un **dominio di protezione** o può essergli consentito **di passare da un dominio a un altro**.

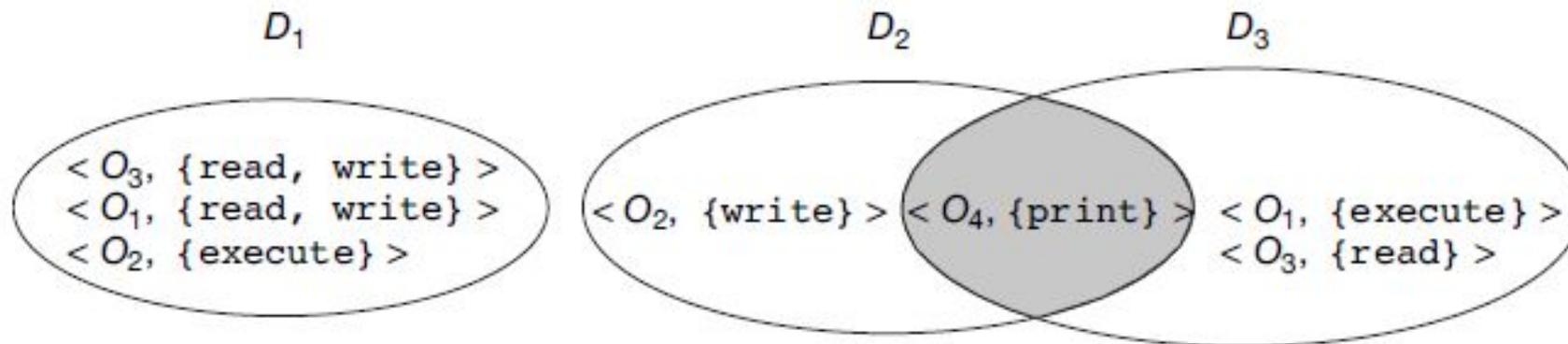


Figura 17.4 Sistema con tre domini di protezione.

Domini di protezione

Un **dominio** si può realizzare in diversi modi:

Ogni *utente* può
essere un
dominio

Ogni *processo*
può essere un
dominio

Ogni *procedura*
può essere un
dominio

Matrice d'accesso

- Le righe della matrice rappresentano i *domini*, e le colonne gli *oggetti*.
- Ciascun elemento della matrice consiste di un *insieme di diritti d'accesso*.

dominio \ oggetto	F_1	F_2	F_3	stampante
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Figura 17.5 Matrice d'accesso.

Matrice d'accesso

Un processo in esecuzione nel dominio D_2 può passare al dominio D_3 oppure al dominio D_4 . Un processo del dominio D_4 può passare al dominio D_1 , e uno del dominio D_1 può passare al dominio D_2 .

oggetto \ dominio	F_1	F_2	F_3	stampante	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

Figura 17.6 Matrice d'accesso della Figura 17.5 con domini come oggetti.

Matrice d'accesso

Un processo in esecuzione nel dominio D_2 può copiare l'operazione *read* in un elemento qualsiasi associato al file F_2 .

Quindi, la matrice d'accesso della Figura 17.7(a) si può modificare nella matrice d'accesso illustrata nella Figura 17.7(b).

oggetto \ dominio	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

oggetto \ dominio	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

Figura 17.7 Matrice d'accesso con diritti copy.

Matrice d'accesso

Il dominio D_1 è il proprietario di F_1 e quindi può aggiungere e cancellare qualsiasi diritto valido nella colonna di F_1 .

Così, la matrice d'accesso della Figura 17.8(a) si può modificare nella matrice d'accesso illustrata nella Figura 17.8(b).

oggetto \ dominio	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write
D_3	execute		

(a)

oggetto \ dominio	F_1	F_2	F_3
D_1	owner execute		
D_2		owner read* write*	read* owner write
D_3		write	write

(b)

Figura 17.8 Matrice d'accesso con diritti owner.

Matrice d'accesso

Confronto tra le due matrici d'accesso delle Figure 17.6 e 17.9.

oggetto dominio	F_1	F_2	F_3	stampante	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

Figura 17.6 Matrice d'accesso della Figura 17.5 con domini come oggetti.

oggetto dominio	F_1	F_2	F_3	stampante	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

Figura 17.9 Matrice d'accesso della Figura 17.6 modificata.

Realizzazione della matrice d'accesso

La **matrice d'accesso** è **sparsa**, ossia la maggior parte dei suoi elementi è vuota.

Normalmente si realizza per mezzo di **liste d'accesso** associate a ciascun *oggetto*, oppure per mezzo di **liste di abilitazioni** associate a ciascun **dominio**.

Si può inserire la **protezione dinamica** nel modello della **matrice d'accesso** considerando i domini e la stessa matrice d'accesso come *oggetti*.

Lo **schema chiave-serratura** (*lock-key scheme*) rappresenta un compromesso tra le **liste d'accesso** e le **liste di abilitazioni**.

Revoca dei diritti di accesso

La **revoca dei diritti d'accesso** in un modello di protezione dinamico è di solito più facile da realizzare con lo **schema delle liste d'accesso** che con **le liste di abilitazioni**.

Tra gli schemi che realizzano **la revoca delle abilitazioni** ci sono i seguenti:

Riacquisizione

Puntatori
all'indietro

Riferimento
indiretto

Chiavi

Controllo dell'accesso basato sui ruoli

controllo dell'accesso basato sui ruoli

(*role-based access control, RBAC*) è una funzionalità che si basa sui **privilegi**, cioè il diritto di eseguire una chiamata di sistema o di sfruttare un'opzione di tale chiamata.

Solaris, dalla versione **10**, realizza il **principio del privilegio minimo** attraverso il controllo dell'accesso basato sul ruolo, una forma di matrice d'accesso.

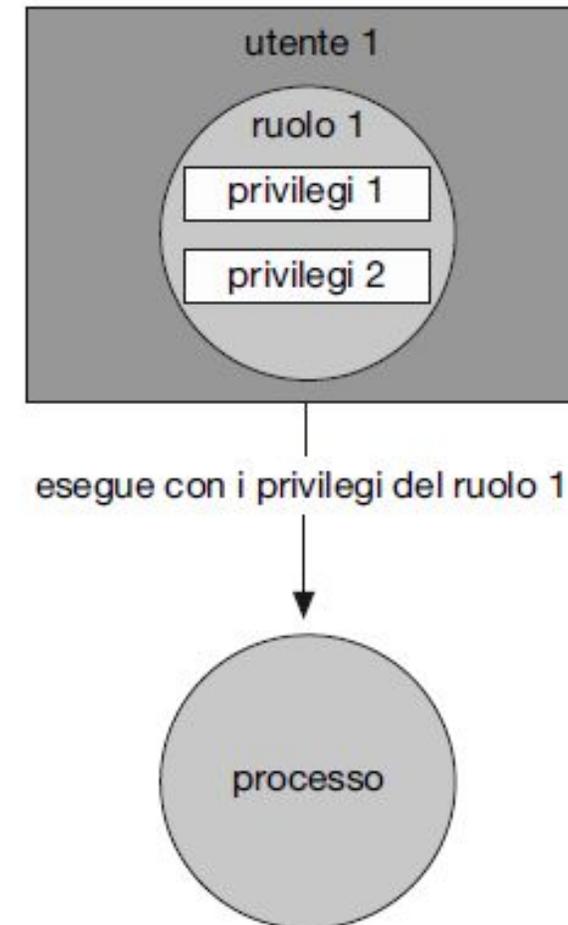


Figura 17.10 Controllo dell'accesso basato sui ruoli in Solaris 10.

Controllo obbligatorio dell'accesso

I sistemi operativi hanno tradizionalmente utilizzato il **controllo discrezionale di accesso (DAC)** come mezzo per limitare l'accesso ai file e agli altri oggetti del sistema.

Un'altra estensione di protezione è il **controllo obbligatorio dell'accesso (MAC)**, una forma di imposizione delle politiche di sistema. Il MAC viene applicato come una politica di sistema che nemmeno l'utente root può modificare.

Il cuore del MAC è il concetto di **etichette** → identificatori (di solito una stringa) assegnati a un oggetto (file, dispositivi e altro).

Sistemi basati su abilitazioni

Abilitazioni di Linux

Le abilitazioni di Linux “spezzettano” i poteri della root in aree distinte, ciascuna rappresentata da un bit in una maschera di bit

Nel vecchio modello, anche la semplice utility ping avrebbe richiesto i privilegi di root per poter aprire una socket di rete raw (ICMP)

Le abilitazioni possono essere pensate come uno “spezzettamento” dei poteri di root in modo che le singole applicazioni possano “tagliare e scegliere” solo quei privilegi di cui hanno effettivamente bisogno

Con le abilitazioni, il ping può essere eseguito come utente normale impostando CAP_NET_RAW, che consente di utilizzare ICMP ma non altri privilegi aggiuntivi

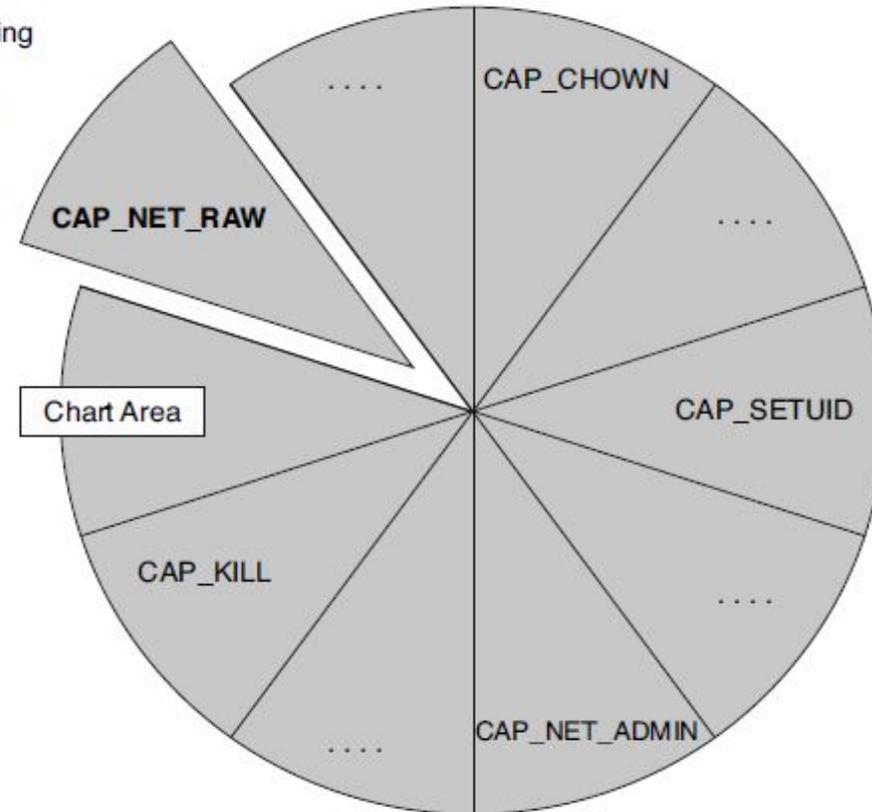


Figura 17.11 Abilitazioni in POSIX.1e.

Autorizzazioni di Darwin

La protezione del sistema di Apple si basa sulle **autorizzazioni** (*entitlement*). Le autorizzazioni sono **permessi dichiarativi** e consistono in un elenco XML di proprietà che indica quali autorizzazioni sono dichiarate come necessarie dal programma.

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>com.apple.private.kernel.get-kext-info
  <true/>
  <key>com.apple.rootless.kext-management
  <true/>
</dict>
</plist>
```

Figura 17.12 Autorizzazioni di Apple Darwin.

Altri metodi per il miglioramento della protezione

Protezione
dell'integrità
del sistema

Filtraggio delle
chiamate di
sistema

Sandboxing

Firma del
codice

```
(version 1)
(deny default)
(allow file-chroot)
(allow file-read-metadata (literal "/var"))
(allow sysctl-read)
(allow mach-per-user-lookup)
(allow mach-lookup)
(global-name "com.apple.system.logger")
```

Figura 17.13 Un profilo sandbox di un demone MacOS che nega la maggior parte delle operazioni.

Protezione basata sul linguaggio

La **protezione basata sul linguaggio** offre un controllo delle richieste e dei privilegi più selettivo di quello ottenibile con il sistema operativo.

Per esempio, una singola JVM può eseguire molti thread, ognuno in un diverso dominio di protezione. La JVM controlla le richieste di risorse attraverso un raffinato **meccanismo di ispezione dello stack** e attraverso la sicurezza dei tipi offerta dal linguaggio.

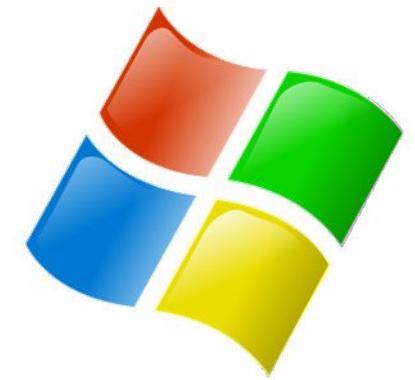
dominio di protezione:	<i>applet</i> non fidata	caricatore di URL	interconnessione
permesso della socket:	nessuno	*.lucent.com:80, connect	qualsiasi
classe:	<code>gui:</code> <code>...</code> <code>get(url);</code> <code>open(addr);</code> <code>...</code>	<code>get(URL u):</code> <code>...</code> <code>doPrivileged {</code> <code> open('proxy.lucent.com:80');</code> <code>}</code> <code><request u from proxy></code> <code>...</code>	<code>open(Addr a):</code> <code>...</code> <code>checkPermission</code> <code>(a, connect);</code> <code>connect (a);</code> <code>...</code>

Figura 17.14 Ispezione dello stack.



**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Corso di Sistemi Operativi



Protezione

Docente:
Domenico Daniele
Bloisi

