# Week 3
# Access Control

THE UNIVERSITY OF SYDNEY

Dr. Thilini Dahanayaka

School of Computer Science, The University of Sydney

# Agenda

- What is access control?
- Operating system access control
- Middleware access control
- Other access control
- Hardware access control
- Security policy model or access control for the organization

# Recommended Reading

- Security Engineering - 3rd Edition by Ross Anderson
  - **Chapter 6** – Access Control
  - **Chapter 9** - Multilevel Security

# 1. Access Control Functions in Brief

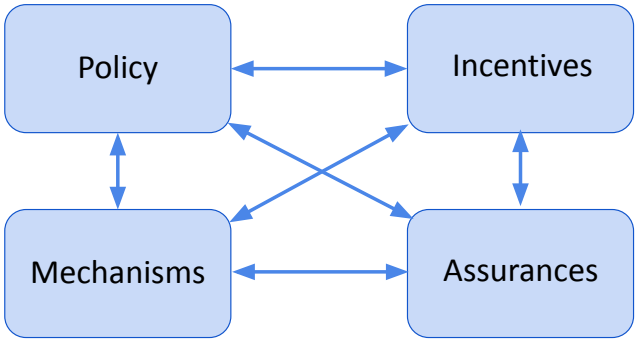
- **Access control**
  - Access control is an important element of a security policy
  - Aim is to control access to resources and assets
  - Can a customer of a bank access the bank vault?
  - Can a program access the passwords of other users?

- **Authentication**
  - Process of acquiring evidence of the identity of another party

- **Authorization**
  - Process of verifying if and entity is allowed an action

# Within our Reference Framework

- Access control is specified in the policy
- Counters incentives of the attacker who wants to access resource/asset
- Implemented by one/more mechanisms
- Which gives us a certain level of assurance.
- Example: Password for access control
  - Passwords are a mechanism.
  - The strength of passwords relates to assurance.



# Example Access Control

- A Naïve access control matrix

| | Operating System | Accounts Program | Accounting Data | Audit Trail |
|---|---|---|---|---|
| Sam | rwx | rwx | rw | r |
| Alice | x | x | rw | – |
| Bob | rx | r | r | r |

**Source:** Security Engineering (Ross Anderson)

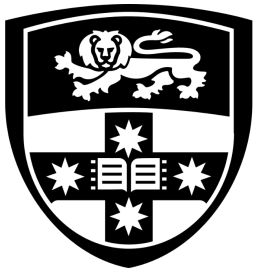
# DAC and MAC

**Objects** (files, processes, ...) are owned by **subjects** (people, groups, processes, ...)

**Discretionary Access Control (DAC):** Access is controlled by the resource owner. Users can grant or revoke permissions to other users at their discretion.

**Mandatory Access Control (MAC):** Access is enforced by a central authority based on security labels (e.g. classification levels). Users can't change access permissions themselves.

- **Discretionary Access Control (DAC)**
  - Access to objects is based on the identity of the owning subject
  - Subjects can pass on a privilege over their object to other subjects—unless constraints exists, e.g., defined by MAC below
  - High level of flexibility; control by subjects
- **Mandatory Access Control (MAC)**
  - An external entity (security policy administrator) defines access
  - 'Security kernel' checks via security attributes whether a subject is allowed to interact with an object
  - High level of rigour



THE UNIVERSITY OF
SYDNEY

# 2. Operating System Access Control

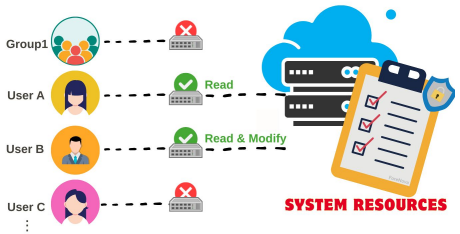
# 2a. Access Control Lists (ACLs)

ACLs define the access of subjects to objects

- Objects are associated with an ACL
  - Can be defined globally or be more fine-grained
- Entry in ACL user ID with a mapping to operations
  - Operations can be coarse (read/write/execute) or very fine-grained
- ACLs are common in all UNIX systems (POSIX standards)
- Also exist in Windows
- Wide-spread mechanism
  - Also found on social networking sites
  - Content Management, Enterprise Resource Management, etc.

| User | Accounting Data |
|---|---|
| Sam | rw |
| Alice | rw |
| Bob | r |

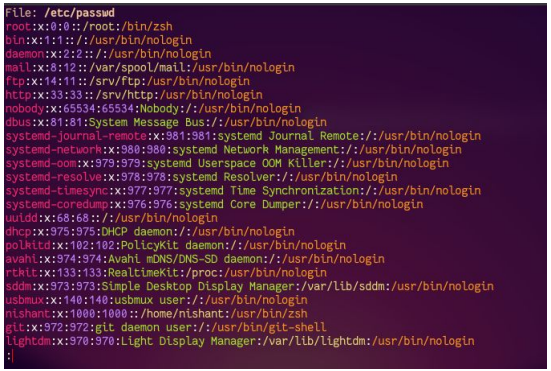
ACL Example

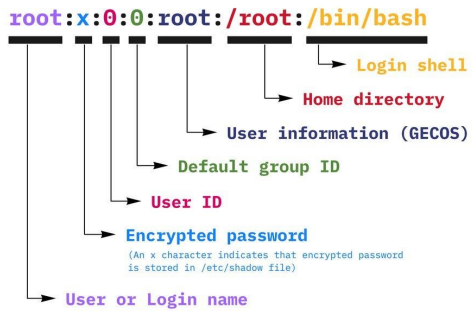


# 2b. Unix/Linux Operating System Security

## User

- Each user has a User ID (UID) and a Group ID (GID)
- E.g. the user could also belong to the sudo group
- Example `/etc/passwd` entry:
  - `sam:x:1000:1000:Sam Jones,,,:/home/sam:/usr/bin/zsh`
  - Also states `login shell`
- Example /etc/group entry
  - `sudo:x:27:sam,alice`
  - group name, GID, users in group, etc.

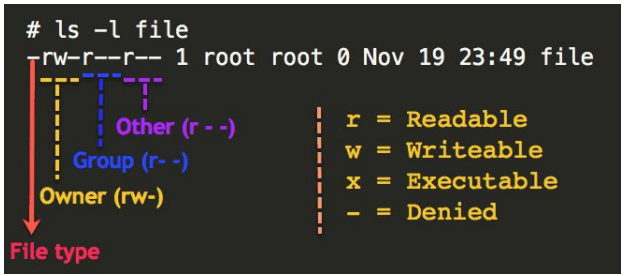


# Unix/Linux Operating System Security

## File Permissions

- **Owner ID:** the user that owns the file
- **Group ID:** the group that ‘relates to’ the file
- **Owner R/W/X:** three bits that determine what the ‘owner’ can do
  - Bit R set means read, W means write, and X means execute
  - Directories that have X set can be traversed
- **Group R/W/X:** bits determine what the group can do
- **Other R/W/X:** bits determine what everyone else can do



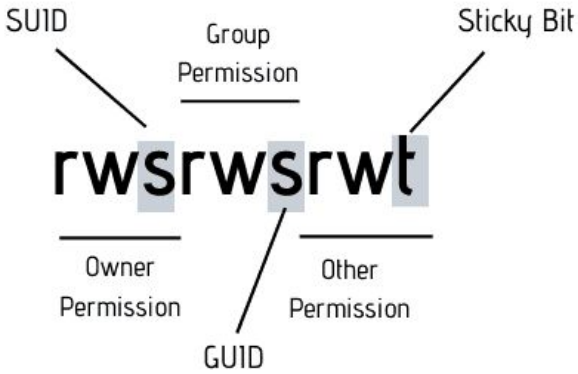

## Examples

```
-rw-r--r-- 1 sam sam 24K Jun 12 10:20 email.png
```

Owner, group and others can all read. Only the owner (sam) can write.

# Unix/Linux Operating System Security

## Files: also have three more bits:




- **Set-user-ID bit (SUID)**
  - File can be executed as the owner
  - Useful for running specific programs with additional permissions without being a privileged user
  - mount is an example
- **Set-group-ID bit**
  - On file: like SUID, but as the group
  - On directories: new files/directories are created with group of parent directory
- **Sticky bit**
  - Only the owner can modify the file
  - Used in `/tmp/`

# Unix/Linux Operating System Security

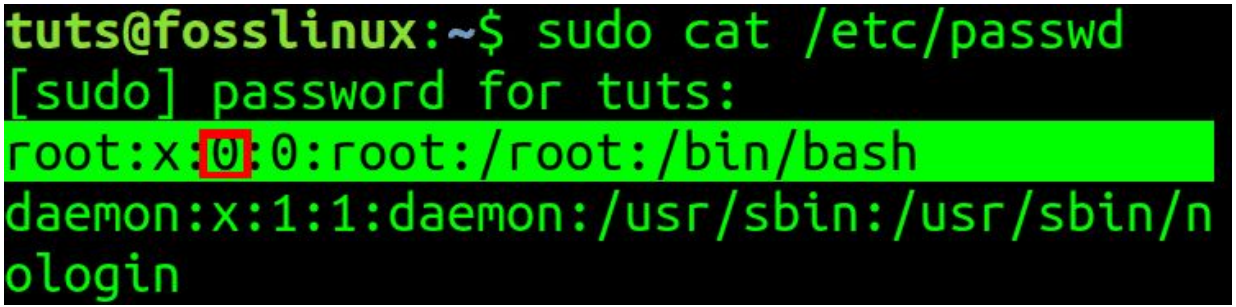
## Processes: inherit permissions from the running user (UID and GID)

- **Real:** UID and GID
  - The user/group that called the processes
- **Effective:** UID (eUID), and GID (eGID)
  - What the process is running as (SUID/SGID affects this)
- **Saved:** UID and GID
  - Used so that the kernel knows what to switch back to

# Unix/Linux Operating System Security

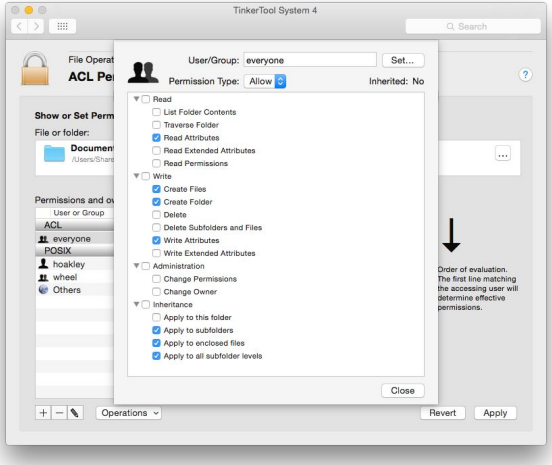
## Root: is an all-powerful user

- UID and GID of 0
- The sudo command lets you run a command as root
  - Only can be done if the invoking user is part of the sudo group
  - Usually requires password
  - Non-sudo users can still run some commands using sudo, if specified via the sudoers file



# 2c. Other operating Systems - MacOS

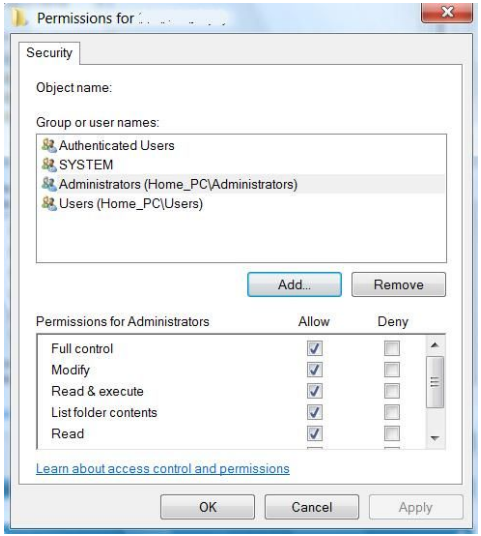
- MacOS is a fork of BSD (Unix)
- Thus, has similar permissions.
- Files have attributes and ACLs.
- Also has a system called the Keychain
- This stores extra passwords and some ACLs
- Also has some network authentication built-in (Kerberos)



# Other operating Systems - Windows

- Very powerful (and Complex)
  - Similar to Unix, but instead of RWX, has 15-16 attributes (!)
  - Traverse Folder/Execute File, List Folder/Read Data, Read Attributes…
  - But complexity, may result in errors despite secure default settings

- Enforced via a Security Reference Monitor (SRM)
  - Trusted component
  - Fine-grained security

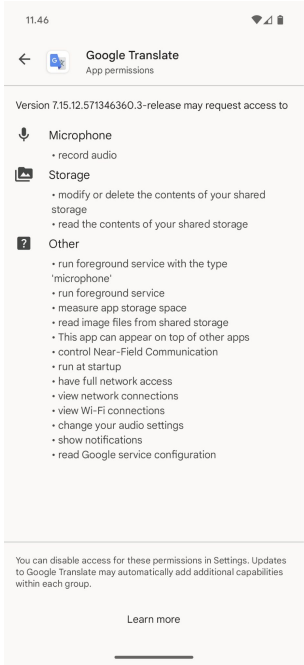


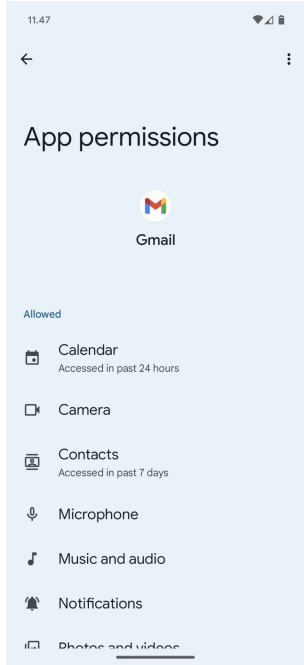
# Other operating Systems - Apple iOS

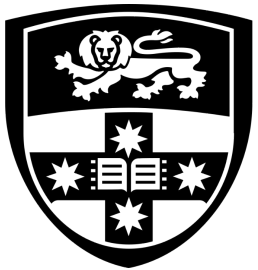
- Many user permissions, often for privacy.
- Read-only for sensor values (Biba model - later!)
- More access control:
  - Hardware support
  - Signed software
  - Encryption
- Sandboxing:
  - Apps cannot access files of other apps
  - Capabilities for apps via APIs
  - Non-privileged execution
  - OS is read-only to app

# Other operating Systems - Android

- Similar permission system
  - Supports many different hardware platforms!
  - Concessions: cannot assume features across hardware
- Problematic issues were there
  - Storage is a particularly well-known one
  - Two forms: app-internal and ‘public storage’
  - Evidence that developers misuse the latter!
  - Easier to work with as everyone can write to it!
  - Unfortunately, everyone can also read!
- More permissive API
  - E.g., apps can send SMS on your behalf
- Google Play Store is less tightly controlled than Apple’s
  - But apps are screened for malware

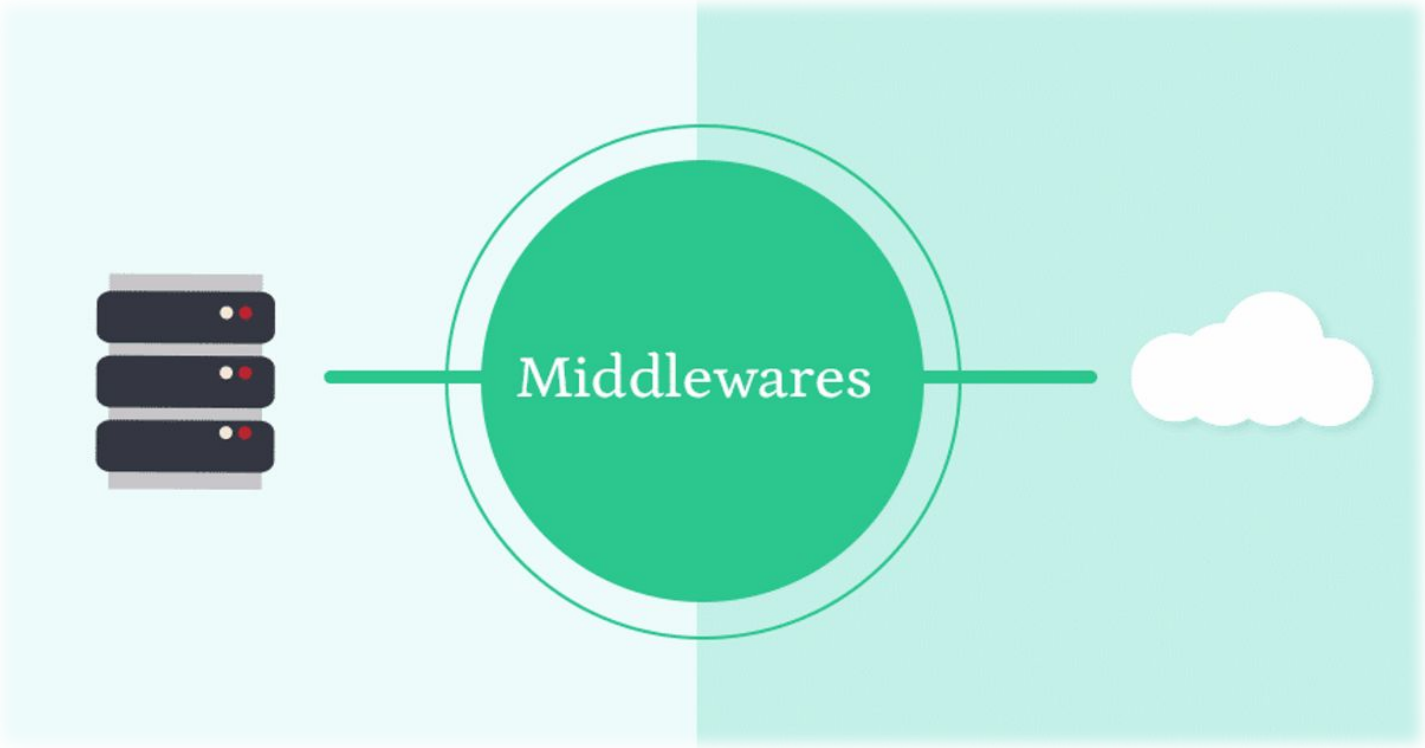




THE UNIVERSITY OF
SYDNEY

# 3. Middleware Access Control

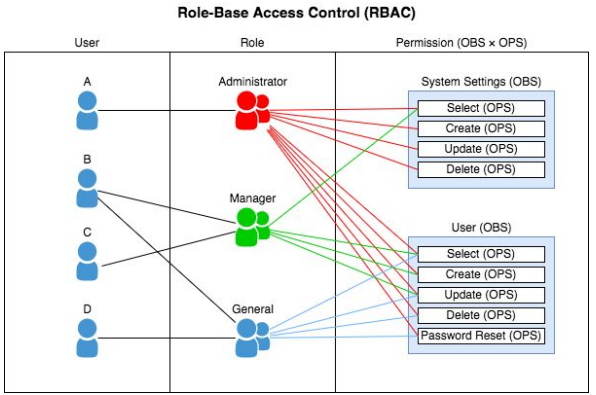


# 3a. Access Control in Middleware

- Middleware can be on the same host or distributed
  - E.g., message brokering (AMQP)
  - RPCs: Java RMI, CORBA, XML-RPC, …
  - Often left open to probing by outsiders!
- Huge complexity:
  - Define granularity well
  - Keep track of permissions and enforce
  - Easily misconfigured
- Policy languages exist to define access control but are hard to use themselves!
- If possible: protect middleware from external access
  - At least protects from random, remote exploitation
  - But note that malware started in one participating system can try to exploit the middleware

# 3b. Database Access Control

- DBMSes with multi-user, multi-process support have their own access control
  - Actual data is still often on disk or in RAM
  - Hence, OS protections are still necessary!
  - Connecting to DB requires a username and password
- Internal privilege management
  - Map users on OS to roles in DB
  - SQL standard defines fine-grained methods for access control
  - In theory, any kind of object can be access-controlled
  - Can lead to overkill in complexity (what is the consequence?)
- DBMSes also allow specific integrity controls
  - Computationally heavy to check!

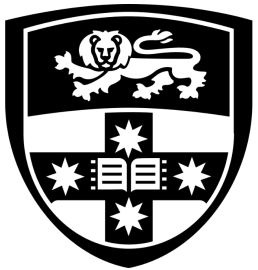


# 3c. Browsers

- Another middleware access control platform
- Main access control rule is the "same-origin policy" – JavaScript or other active content on a web page is allowed to communicate with the IP address that it originally came from
- Code run in a "sandbox" to prevent it from altering the host system

**Confused Deputy Problem:** The Confused Deputy Problem occurs when a privileged program (the "deputy") is tricked by a less-privileged entity into misusing its authority on the attacker's behalf.

More on this later in web security



THE UNIVERSITY OF
SYDNEY

# 4. Other Access Control

# 4a. Sandboxing

- Shielded environment: malicious applications cannot interfere with the surrounding environment
  - Typically provides a limited API to interact with

- Examples:
  - Java applets: running in the browser (almost historical today)
  - JavaScript in the browser
  - Google Chrome: tabs run in their own process

- Problem: the sandbox is hard to secure: can be ‘broken out of’

# 4b. Virtualization

- Use case: Virtual Machines (VMs) emulate a computer system
  - Old technology: around the 1960s!
  - Allows to run software even if the real environment it was written for is not available
  - OS running in VM sees only its ‘own’ hardware

- Varying degrees of virtualisation:
  - Full emulation of hardware (full virtualisation even across hardware architecture)
  - Para-virtualisation (guest OS must support being virtualized)
  - Containers and jails are actually a form of light-weight virtualisation

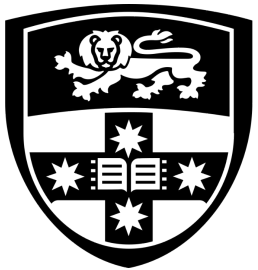
# Virtualization

- Hypervisor manages and runs VMs: hardware support in Ring -1 (later)
  - In full and para-virtualisation
  - Examples: Virtualbox, VMWare, Xen, QEMU, …

- Not designed for access control, but the typically high effort required to break out
  - Raises the bar for attackers
  - Note that execution in a VM is detectable: the VM does not perfectly imitate a real environment
  - Malware can detect it is inside one and act differently
  - Vulnerabilities exist in the hypervisors and can be exploited: always keep up-to-date, and do not use as a single defence

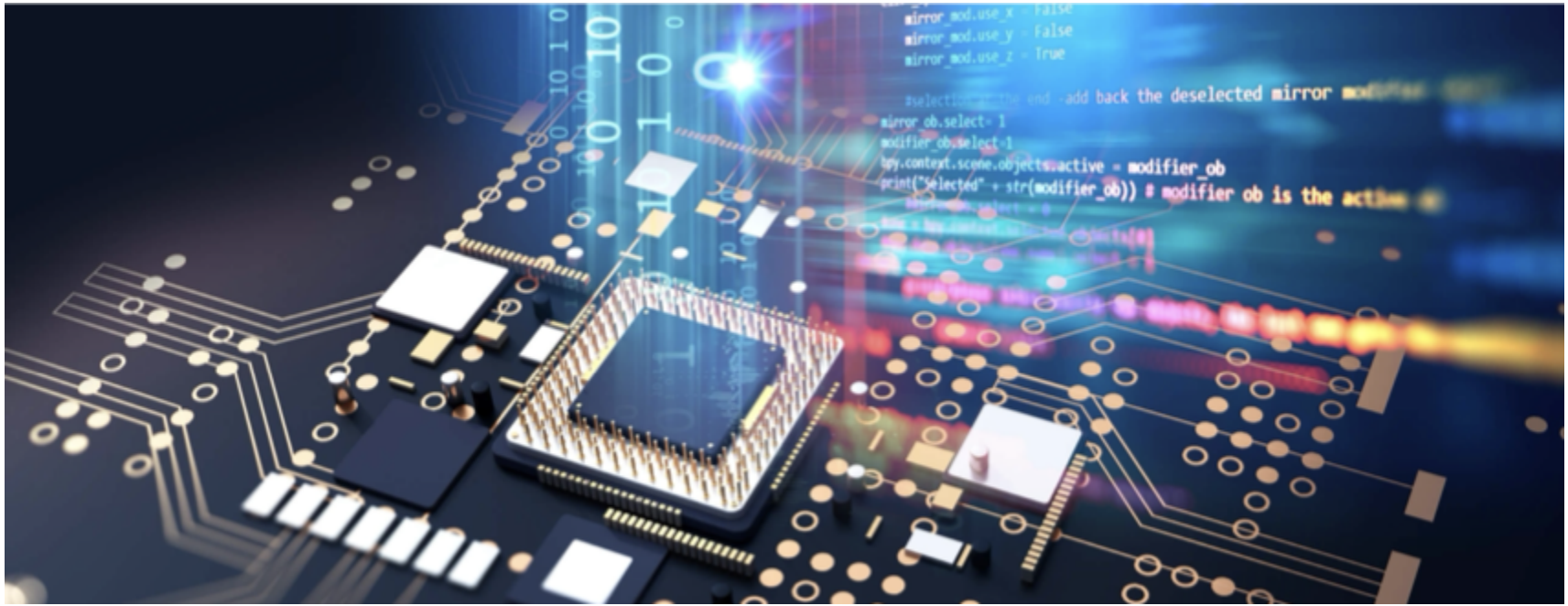
# 4c. Example: Docker

- Initial use case: continuous integration and delivery (CI/CD)
- Abstraction over containers
  - Supports several OSes via their native jail implementations
  - Builds images out of apps and supporting libraries (!)
  - Fast to spin up and delete, easy on resources
  - Management systems (e.g. Kubernetes) available
- Sandboxes the app in the container
  - App cannot communicate with app in another container
  - Docker daemon is broker but runs as root
  - Means any user who can start/stop containers can interact with the daemon: attack surface
- While not designed for access control, containers provide an extra degree of separation and raise the bar for attacks



THE UNIVERSITY OF
SYDNEY

# 5. Hardware Access Control



# Hardware Protections

## Memory Protection

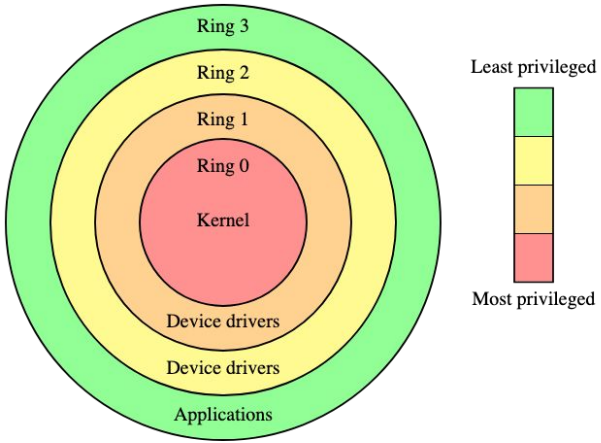
- CPUs support the OS in keeping processes' memory allocations apart
  - Segment addressing: memory is addressed by two registers – a segment register that points to a segment of memory **and** an address register that points to a location within that segment
  - Segmentation fault: SIGSEGV is raised if a process tries to access memory outside its address space
  - Both read and write

## Privileged execution

- CPUs have layered modes (rings) to support OS
- Rings separated with different privilege classes
- ARM has even separate registers with different privilege classes

# Kernel and Hardware

- Code runs in one of the available CPU modes, called rings.
- CPUs enforce access control from higher privileged rings to lower
- Process running in a higher ring cannot directly interact with process in lower
- **Principle of least privilege:** all code should run only with minimal required privilege, never more.
- Number of ways to communicate with the kernel should be limited to the required minimum, well-controlled



# Hardware Protections

**Isolated Cryptographic Components**

- Storing keys in hardware
  - Functionality that uses it via an interface

- Example: Intel TPM (Trusted Platform Module)
  - Originally conceived for Digital Rights Management

# Intel Processors

- Overview of Intel processors:
  - The Intel 80286 has segment addressing and rings
  - The Intel 80386 has built-in virtual memory and a memory segment of 4GB

- Rings of protection are supported by a number of mechanisms

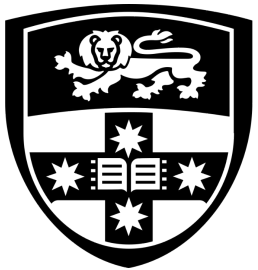
- Modern Intel CPUs have nine rings: 0-3 for normal code, under which is a further set of rings 0-3 Virtual Machine Manager (VMM) root made for hypervisor, and System Management Mode (SMM) for BIOS

# ARM Processors

- Most used in phones, tablets and IoT devices

- ARM-based designs have their hardware protection extensively customised.
  - Memory management units (MMUs)
  - Memory protection units (MPUs)

- ARM latest offered **CHERI (Capability Hardware Enhanced RISC Instructions)** – helps multiple sandboxes run in the same process, and provides fine-grained spatial memory safety at a hardware level → potential to stop the zero-day exploits



THE UNIVERSITY OF
SYDNEY

# 6. Security Policy Models

# Multilevel Security (MLS) Policy

- MLS is a foundational form of access control
- Idea: information must be handled at different levels of classification
  - Top Secret, Secret, Confidential, etc.
  - Enormous importance in military and government
- Surprisingly difficult to implement while maintaining high utility
- **Example:** how can a system provide automated means to reduce the classification of a document from top secret to secret, so that other, dependent principals at lower classification can read it?

# Security Policy Models

- When analyzing a system's security, one often follows this order:
  - Analyze incentives and create a *t*hreat model—a model that describes the attacker's motivations and capabilities
  - Analyze security policy—the policy describes the goals to be achieved (relatively verbose)
  - Analyze security mechanisms to implement the security policy

- A Security Policy Model is a highly precise and succinct statement about the protection properties of the system
  - Can be highly formal
  - Drives engineering

# Bell-LaPadula Model

- Classic security policy model for MLS
- 1970s: growing realization that OSes would always have vulnerabilities that even users could work around
- Realized need to implement MAC that could not be bypassed
- Two forms of Mandatory Access Control and one form of Discretionary Access control

1) **No read up** aka **Simple Security**: no principal may read data at a higher security level
2) **No write down** aka ***-property**: no principal may write data to a lower security level

The *-property is crucial:

- Example: user at low security level writes a program to exfiltrate data
- Waits for admin at high security level to accidentally execute it
- Needs MAC to prevent this!

# Bell-LaPadula Model

- Bell-LaPadula has one **discretionary** form of access control

3) External entity can define the access of subjects to objects in a matrix

| | $o_1$ | $o_2$ | $o_3$ | $o_4$ |
|---|---|---|---|---|
| **Emma** | read | append | ∅ | read |
| **Daniel** | ∅ | read-write | append | execute |

# Discussion of Bell-LaPadula

**Innovations**

- Blends itself well to formal analysis
- Combines DAC and MAC

**Criticisms**

- Administration via one omnipotent Trusted Principal—how to secure that?
- Definition with confidentiality in mind, not integrity
- Does not consider covert channels
  - E.g. write program to encode 1 as high CPU load; 0 as low
  - Program on lower level can estimate CPU load
- System composed of two Bell-LaPadula-secure systems is **not** necessarily Bell-LaPadula-secure

# Biba Model

- Addresses **integrity**, not confidentiality: prevent malicious change
- A reverse of Bell-LaPadula:

**1) Read up**: principal may only read data at own level or higher

**2) Write down**: principal may only write data at own level or lower

**3) Invocation property**: principal cannot obtain any access to objects at higher level (only own and lower)

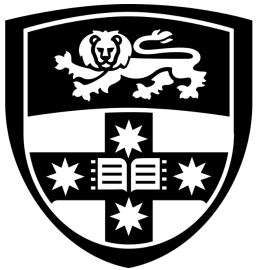
- Examples to provide clarity:
  - Aircraft in-flight entertainment system can read data from avionics (e.g., airspeed), but cannot affect it
  - Odometer can display mileage, but not change it

# Use of Model

- Pure Bell-LaPadula implementations (or any other model) are rare
- Variants of Biba, however, are more common:
  - Windows Vista
  - UNIX: SELinux, some Red Hat versions, FreeBSD module
- Concepts, however, are useful to discuss real-world access control
- In the remainder of this lecture, we focus on real-world concepts found in mobile, desktop, and server OSes and hardware

# Recap

- **We discussed:**
  - Terminology: Authentication, Access Control, and Authorization
    - Principals
    - Layered approach for access control
  - DAC and MAC as well as other forms of access control
  - Bell-LaPadula and Biba Models - Their security policies
  - Access control in a computer system (Applications, Middleware, Operating System, Hardware)
  - Hardware support for access control
  - Linux/Unix permissions and access control specifics



THE UNIVERSITY OF
SYDNEY