Chapter 7: Principles of Security Design, Models, and Capabilities

- **Learning Objectives:**
  - Understand core security design principles
  - Explain major security models
  - Apply principles to real world systems
  - Recognize modern threats and architectures

Principles of Security Design

Security Architecture principles

- "**Security architecture**": the overall system required to protect an organization's IT infrastructure.
  - includes the H/W & S/W specifications, processes, and procedures involved in preventing, mitigating, and investigating different threats.
  - The main goal of having this architecture is to have (Confidentiality, Integrity, and Availability)

- **"Security Architecture principles"**
  - fundamental security rules that should be met during the development of a security architecture and applied when the corresponding security controls are implemented.

Why Security Design Matters

- Security failures are design failures
- Retrofitting security is costly
- Modern systems are:
  - Distributed
  - Cloud-based
  - Continuously evolving

- 💡 ***Security must be built in—not added later***

Security Architecture principles

- Core Security Architecture principles, which are considered fundamental within the information security community are:
  - #1 Security by design.
  - #2 Simplicity.
  - #3 Defense in depth.
  - #4 Least privilege.
  - #5 Default deny.
  - #6 Fail secure.
  - #7 Separation of duties
  - #8 Do not trust external systems.
  - #9 Zero Trust
  - #10 Micro segmentation
  - #11 Assume breach
  - #12 Continuous monitoring
  - #13 Security Automation

Security Architecture principles

- **#1 Security by design**
  - The security requirements of a system or application should be considered as part of its overall requirements (and not as an afterthought),
    - To avoid wasting unnecessary time, money and effort.
    - Reduce cost and vulnerability

- **#2 Simplicity**
  - Make it simple: by reducing the complexity and diversity of security controls, less mistakes and errors should occur.
  - Simplicity of security controls should result in **better understanding and management of security controls, and the prompt resolution** of security-related issues.

Security Architecture principles

- **#3 Defense in depth**
  - Using layers of **security increases the level of effort required by an attacker** to gain unauthorized access to a system or application.
  - In the event **one security control fails or is compromised, another security control should prevent** the exposure of information or an information system.

- **Example**: An enterprise network might have **a perimeter firewall**, an **intrusion detection system** (IDS), **network segmentation**, secure endpoints, and **encryption to protect data**. Even if the firewall is bypassed, the IDS and other security layers provide additional barriers.

Security Architecture principles

- **#4 Least privilege**
  - Only the **minimum possible privileges should be granted** to a user or a process for accessing a resource.
  - Grant the users the privileges they need to accomplish a certain task: No more, No less
  - Limit damage if compromised
- **Example**: A database administrator should only have access to databases they manage, and not to applications or files unrelated to their job. Similarly, a user account that typically views records should not have write or delete permissions unless explicitly needed.

Security Architecture principles

- **#5 Default deny**
  - The default setting for a security control should be **to deny access** to a resource and require a configuration to specifically grant access.

- **#6 Fail secure**
  - One of the most common programming or design errors is "forgetting to handle a case"

    - **Fail Secure: The control will block accessing the process to avoid system damage.** (results in denial of service)

    - **Fail Safe: The control will be disabled to ensure the process is accessible** (results in the possibility of illegitimate access).

Security Architecture principles

- ***#7 Separation of duties***
  - no one has sole control over the lifespan of a transaction.
    - Task Shadowing (one works and one monitors)
    - Task splitting (split the task between two persons)
  - **Example**: In a financial organization, one person might initiate a transaction while another authorizes it. This ensures that fraud is less likely since it requires collusion between two individuals.
- **#8 Do not trust external systems**
  - External systems and environments are typically not under the control of an organization. Therefore, it is recommended that external systems are assumed to be insecure until a level of trust is established.

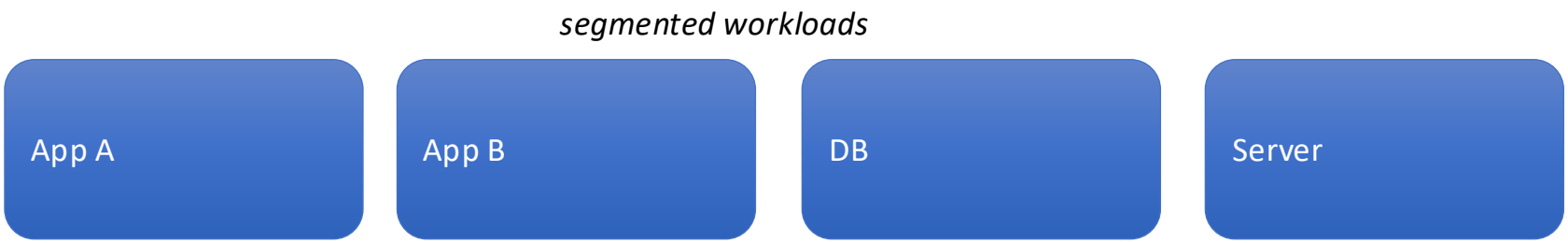
Security Architecture principles

- **#9 Zero Trust**
  - Zero Trust is a security model that assumes no actor, system, or service, inside or outside the perimeter, is trusted. Every interaction must be verified before access is granted.
  - **Example**: An internal employee trying to access a company file system might still need to undergo multifactor authentication (MFA) and device verification even though they are inside the corporate network.

Security Architecture principles

#10 Micro Segmentation

- Divides a network into small, isolated segments
- Prevents lateral movement of attackers
- Enforces least-privilege access
- Controls east–west traffic
- Key component of Zero Trust security



Security Architecture principles

#10 Micro Segmentation

- Example: **University Network (like CCIS labs)**
- Without micro-segmentation:
- Student PC → can access:
  - Database server ❌
  - Admin systems ❌
  - Other students' machines ❌
- With micro-segmentation:
- Student PC → can access:
  - Only course server ✅
- Database → only accessible by authorized apps ✅
- Admin systems → completely isolated ✅
- 💡 Even if one machine is hacked → attacker is **trapped**

Security Architecture principles

#11 Assume Breach

- Assume attacker is inside
- Focus on detection and containment
- Traditional mindset:
  👉 “We must keep attackers out”
- Assume breach mindset:
  👉 “Attackers may already be in—limit the damage”

💡 *Drives monitoring and segmentation*

#12 Continous Monitoring

- Real-time visibility
- Logs + alerts + analytics
- **Tools:**
  - SIEM, EDR

Security Architecture principles

#13 Security Automation

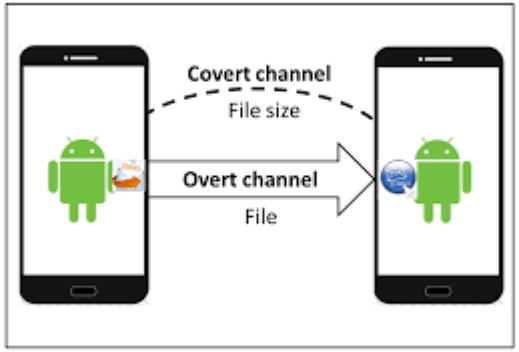
- Automate detection & response
- Use AI-driven tools
- 💡 *Reduces human error*

Security Architecture principles

- **Security is a Process**
  - Every system has vulnerabilities
    - Impossible to eliminate all of them
    - Goal: assurance
  - Systems change over time
    - Security requirements change over time
    - Context of mechanisms changes over time
  - Secure systems require maintenance
    - Check for obsolete users
    - Update virus software
    - Patch security holes
    - Test firewalls

Common Architecture Flaws and Security Issues 1/2

- **Covert Channels**
  - an evasion or attack technique that is used **to transfer information in a secretive, unauthorized or illicit manner**.
  - A **covert channel** can be used to extract information from or implant information into an organization
- **There are two types:**
  - **Covert timing channels**
    - The process can access the data of another process in a shared resource (RAM, CPU)
  - **Covert storage channels**
    - The process access the storage media to read or write the data
      - Android permission to access photos



Common Architecture Flaws and Security Issues 1/2

- **Attacks based on Design or Coding Flaws and Security Issues**
  - **Maintenance hook**: A trapdoor in software that allows easy **maintenance** and development of additional features and that may allow entry into the program at unusual points or without the usual checks.
  - **privileged programs**: one that can **give users extra privileges** beyond that are already assigned to them.
  - **Data diddling** is the changing of data before or during entry into the computer system.
  - **Salami (aggregation) attack:** small **attacks** add up to one major **attack** that **can** go undetected also called **Incremental**
    - *Collecting user data from multiple parties*
    - *Start with small action and then increase the attack's coverage*
- **How to protect against Attacks Based on Design or Coding Flaws and Security Issues**
  - **Trusted recovery**: refers to mechanisms and procedures necessary to ensure that failures and discontinuities of operation don't compromise a system's secure operation.
  - **Input and parameter checking (input validation)**

Techniques for Ensuring Confidentiality, Integrity, and Availability

Techniques for Ensuring Confidentiality, Integrity, and Availability

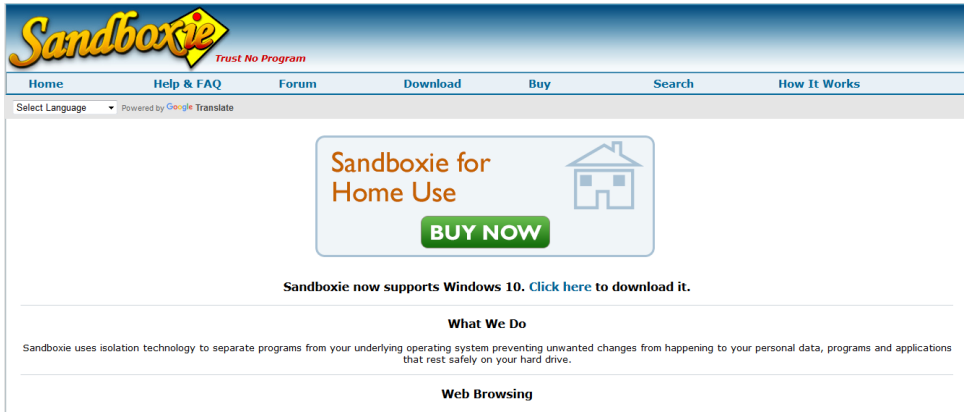
- Software designers use different techniques to **ensure that programs do only what is required** and nothing more.
  - If an affected program is processing sensitive or secret data, that data's confidentiality is no longer guaranteed.
  - If that data is overwritten or altered in an unpredictable way, there is no guarantee of integrity.
  - if data modification results in corruption or loss, it could become unavailable for future use.

Techniques for Ensuring Confidentiality, Integrity, and Availability

Concepts affecting the design of secure programs and operating systems

Confinement

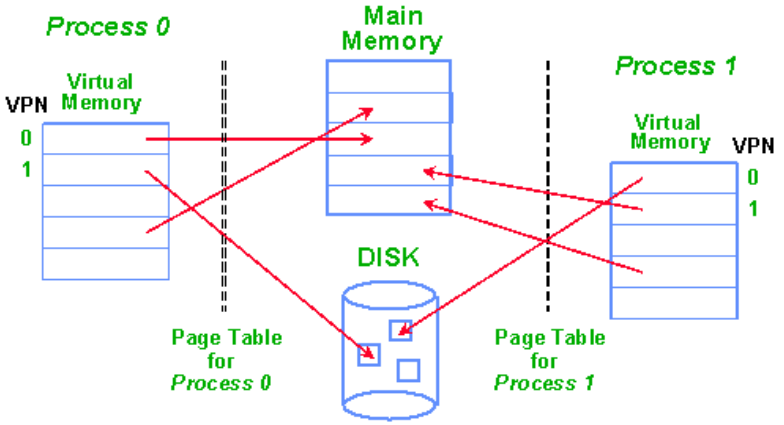
- process confinement allows a process to read from and write to only certain memory locations and resources (e.g. Sandboxing)
- Confinement used in the operating system (for process isolation and memory protection)



Techniques for Ensuring Confidentiality, Integrity, and Availability

Process Isolation

- Process isolation ensures that any behavior will affect only the memory and resources associated with the isolated process.

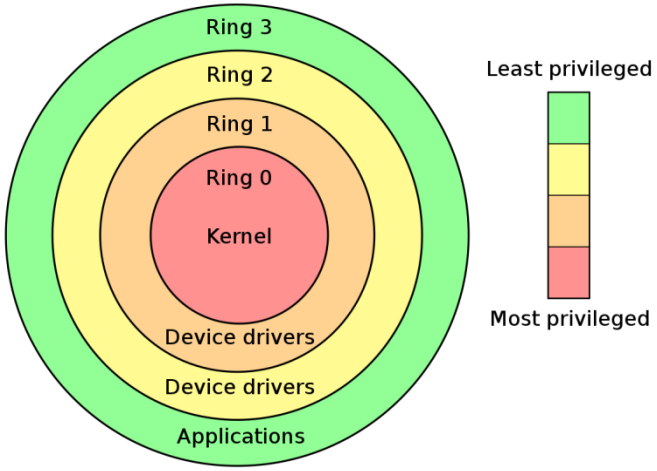


Virtual addresses of each process begin from 0
Protection implies at each instant page tables are disjoint

Techniques for Ensuring Confidentiality, Integrity, and Availability

Bounds

- **Each process that runs on a system is assigned an authority level**.
- The authority level tells the operating system what the process can do
- The two common authority levels are user and kernel.



Security Models

Security Models

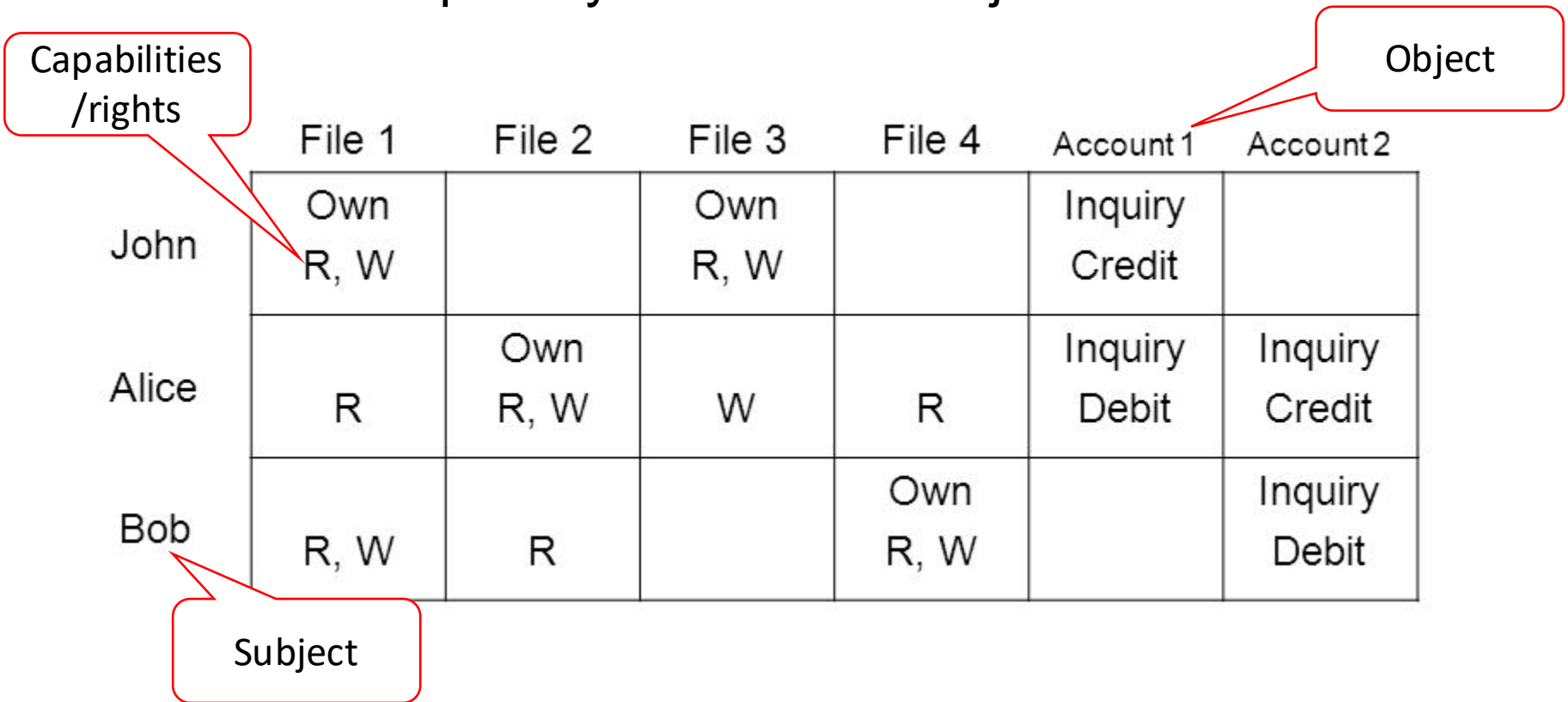
- is a scheme for **specifying and enforcing security policies**.
- A security model may be founded upon a formal model of access rights, a model of computation, a model of distributed computing, or no particular theoretical grounding at all.

Security Models

- ❑ Trusted Computing Base
- ❑ State Machine Model
- ❑ Information Flow Model
- ❑ Noninterference Model
- ❑ **Take-Grant Model**
- ❑ **Access Control Matrix**
- ❑ **Bell-LaPadula Model**
- ❑ Biba Model
- ❑ Clark-Wilson Model
- ❑ Brewer and Nash Model (aka Chinese Wall)
- ❑ Goguen-Meseguer Model
- ❑ Sutherland Model
- ❑ Graham-Denning Model

Access Control Matrix

- A table of subjects, objects, and access
- Columns are access control list(ACLs) tied to objects
- Rows are capability lists tied to subjects

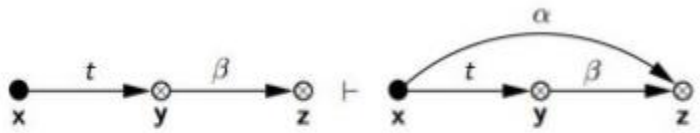


Take-Grant Model

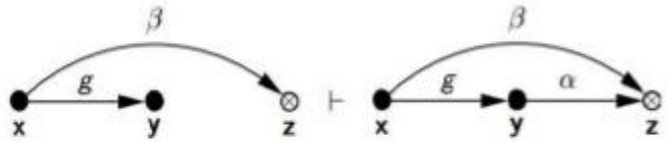
Dictates how rights can be passed between subjects

Take rule	Allows a subject to take rights over an object
Grant rule	Allows a subject to grant rights to an object
Create rule	Allows a subject to create new rights
Remove rule	Allows a subject to remove rights it has

The key to this model is that using these rules allows you to figure out <u>when rights in the system can change and where leakage occurs</u>



The rule can be read: "x takes (α to z) from y."



The rule can be read: "x grants (α to z) to y."

The rule can be read: "x creates (α to) new {subject/object}n."

The rule can be read: "x removes (α to) y."

Bell-LaPadula Confidential Model

- Security levels arranged in linear ordering
  - Top Secret: highest
  - Secret
  - Confidential
  - Unclassified: lowest
- Levels consist of *security clearance L(s)*
  - Objects have *security classification L(o)*

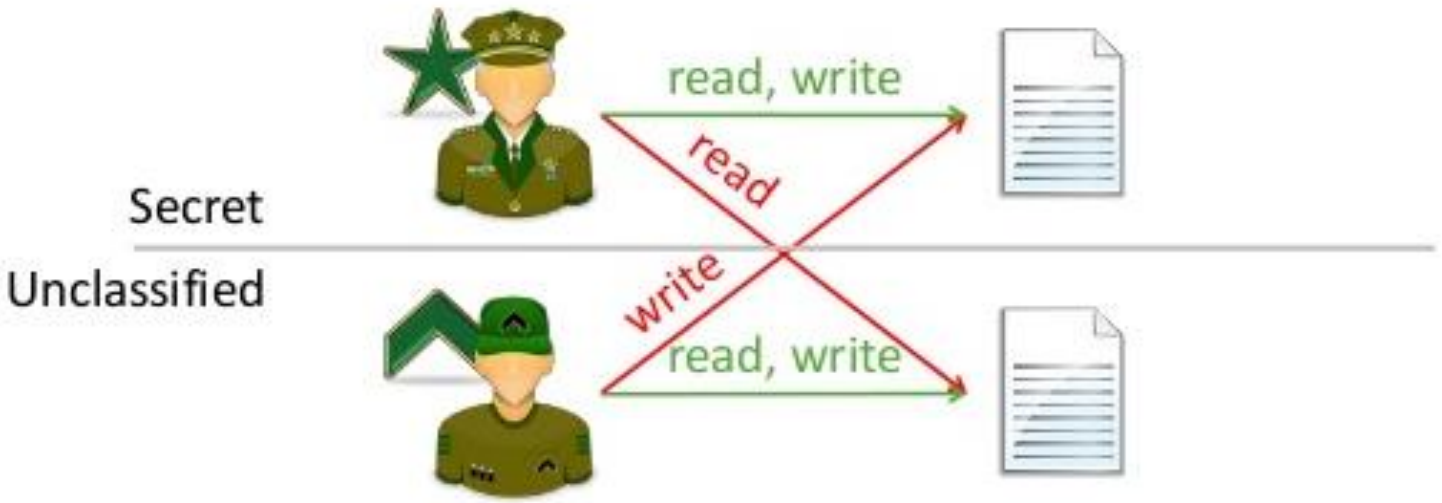
security level	*subject*	*object*
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	Alice	Telephone Lists

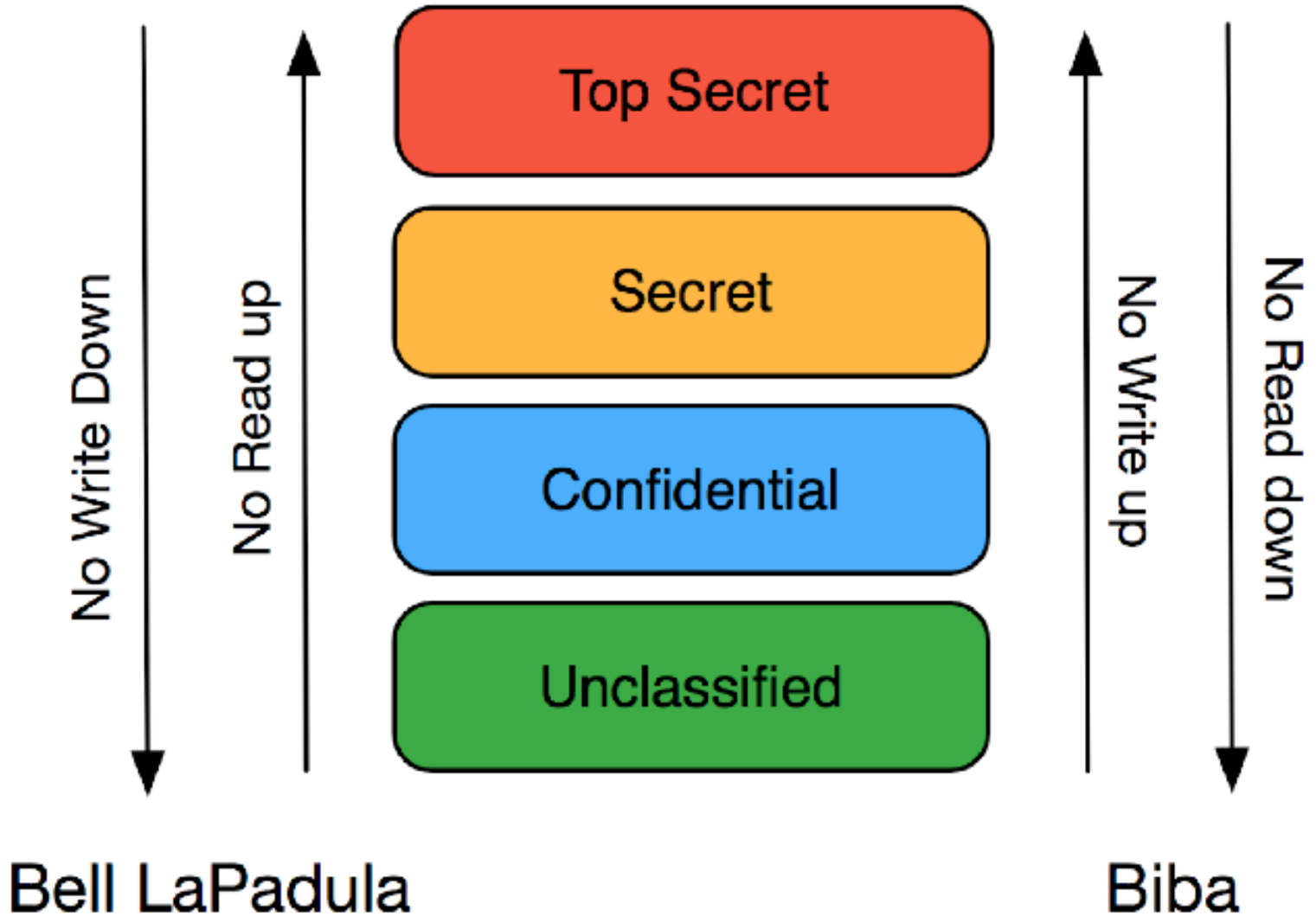
- Tamara can read all files
- Claire cannot read Personnel or E-Mail Files
- Alice can only read Telephone Lists

Biba Integrity Model

- Model of Biba:
  - No write up
  - No read down

  } Integrity




Top Secret
Secret
Confidential
Unclassified
No Write Down
No Read up
No Write up
No Read down
Bell LaPadula
Biba