Secure Architecture and Design Principles

What This Course Is About

Your Goal: Learn how to build security into systems from the very beginning—not as an afterthought. You'll master core security principles, understand proven security models, and know how to configure systems safely.

Section A: Core Security Design Principles

Defense in Depth

Principle 1: Layering

What It Means: Never rely on a single security control. Instead, use multiple layers of overlapping security measures that work together to protect your systems.

The Three Types of Controls

Preventive Controls

Stop attacks before they happen

- Firewalls that block malicious traffic
- Access controls that prevent unauthorized logins
- Encryption that protects data from being read

Detective Controls

Identify when attacks are happening

- Intrusion detection systems that alert you to suspicious activity
- Log monitoring that tracks what's happening in your systems
- Security audits that find vulnerabilities

Corrective Controls

Fix problems after they occur

- Backup systems that restore lost data
- Patch management that fixes vulnerabilities
- Incident response procedures that contain breaches

How to Apply This:

- Design your architecture with multiple security layers at different points
- Make sure each layer uses different security approaches
- Ensure that if one control fails, others are still protecting the system
- Include preventive, detective, and corrective controls at each layer

Principle 2

Least Privilege

What It Means: Give every user, application, and system process only the bare minimum permissions they need to do their job—absolutely nothing more.

How to Implement Least Privilege

01

Identify actual needs

Determine what each user or system truly needs to access

02

Grant minimal permissions

Provide only those specific rights

03

Use access control systems

Implement formal systems to manage and enforce permissions

04

Review regularly

Audit permissions periodically and adjust when roles change

05

Remove unused access

When someone changes roles or leaves, immediately revoke their old permissions

Principle 3

Fail Securely (Secure Failure)

What It Means: Design systems so that when something breaks, crashes, or malfunctions, the default behavior is to deny access and protect resources rather than leaving things wide open.

Principle 4

Zero Trust / Trust but Verify

What It Means: Never automatically trust anyone or anything based solely on their network location. Always verify identity and permissions before granting access, regardless of where the request comes from.

Core Concepts of Zero Trust

No implicit trust

Don't assume inside the network = trustworthy

Always authenticate

Verify who or what is making every access request

Always authorize

Confirm they have permission for what they're trying to do

Continuous verification

Keep checking throughout a session, not just at login

Confinement

Restrict users and processes to only the specific resources they need

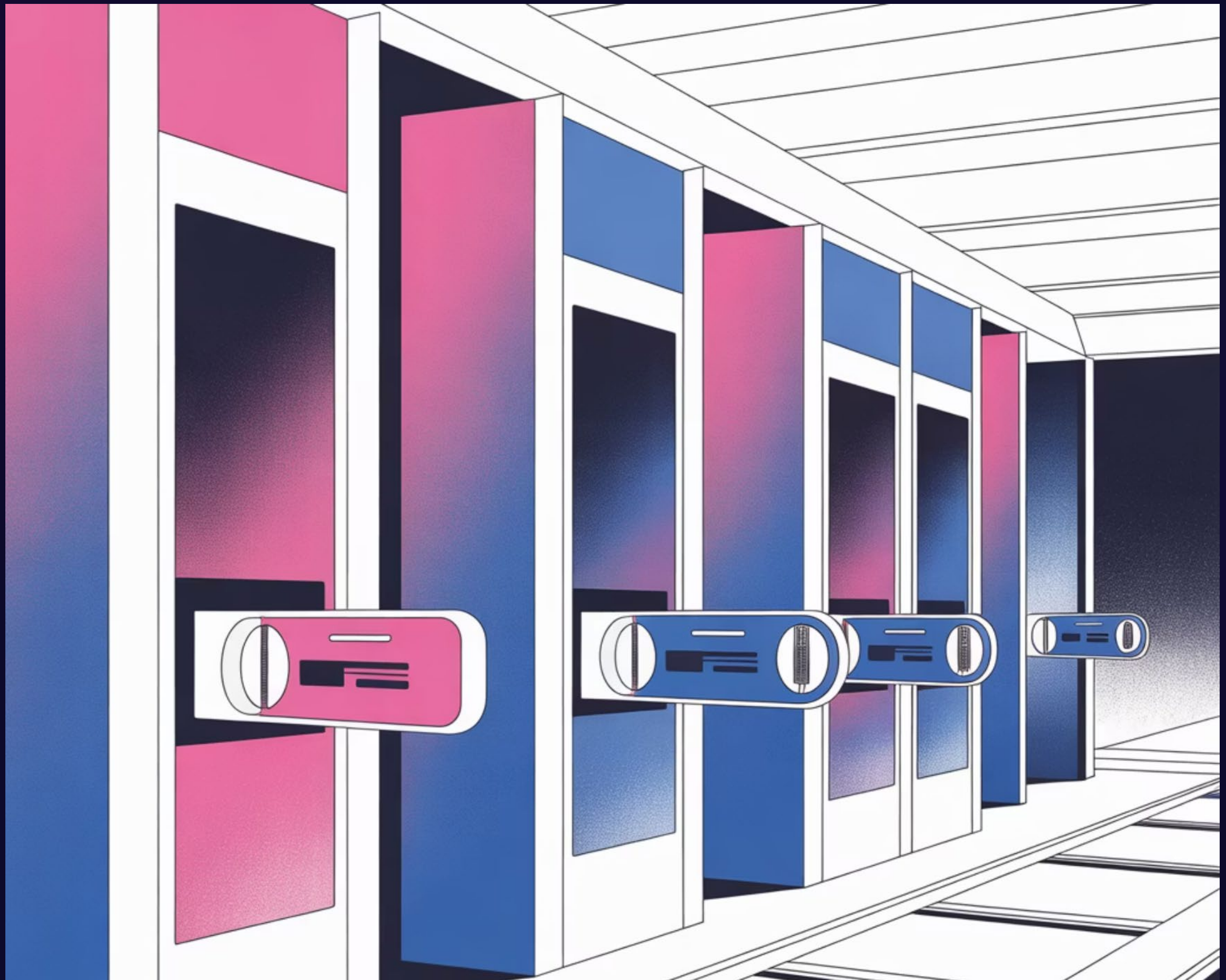
Traditional Security vs. Zero Trust

Traditional (Perimeter) Security:

- Strong defences at the network edge
- Once inside, users can move freely
- Trust based on being "inside" the network
- Like a castle with high walls—once you're in, you have the run of the place

Zero Trust:

- Security controls everywhere, not just at the perimeter
- Every access request is verified
- Trust is never assumed, always earned
- Like a hotel where you need a key card for every room, elevator, and amenity



How to Implement Zero Trust

Require authentication everywhere

Even for internal resources

Implement micro-segmentation

Divide your network into small, isolated zones

Use identity-based access

Grant access based on who you are, not where you are

Apply confinement

Limit each user/process to the minimum necessary operations

Monitor continuously

Watch for anomalous behaviour even after authentication

Principle 5

Keep It Simple

What It Means: Simpler systems are more secure systems. Avoid unnecessary complexity in design, implementation, and deployment. Use only the resources and tools you actually need.

Key Concepts:

Computing Minimalism

Use the least necessary hardware and software

Rule of Least Power

Choose the simplest tool or language that can effectively solve your problem

Avoid feature creep

Don't add functionality you don't need

Optimise resources

Techniques like PERT (Program Evaluation and Review Technique) help achieve efficiency

How to Implement Simplicity

1

Start with requirements

Identify what you actually need, not what might be nice to have

2

Choose appropriate tools

Don't use enterprise solutions for simple problems

3

Minimise dependencies

Fewer libraries and frameworks mean fewer vulnerabilities

4

Remove unnecessary components

Uninstall or disable anything not essential

5

Use straightforward designs

Clever, complex solutions often hide bugs

6

Document clearly

Simple systems are easier to understand and maintain

Section B

Classic Security Models

These are mathematically rigorous frameworks developed over decades of research. They provide formal rules for enforcing specific security goals. While they may seem abstract, they're used in real systems protecting government secrets, financial data, and critical infrastructure.

Bell-LaPadula (BLP) — Protecting Confidentiality

What It's For: The Bell-LaPadula model prevents unauthorised disclosure of sensitive information. It's all about keeping secrets secret. This model focuses exclusively on confidentiality.

The Classification System:

Information is assigned to security levels (e.g., Unclassified, Confidential, Secret, Top Secret). People are given clearance levels that determine what they can access.

The Two Fundamental Rules:

Rule 1: Simple Security Property ("No Read Up")

The Rule: A subject (person or process) cannot read information classified higher than their clearance level.

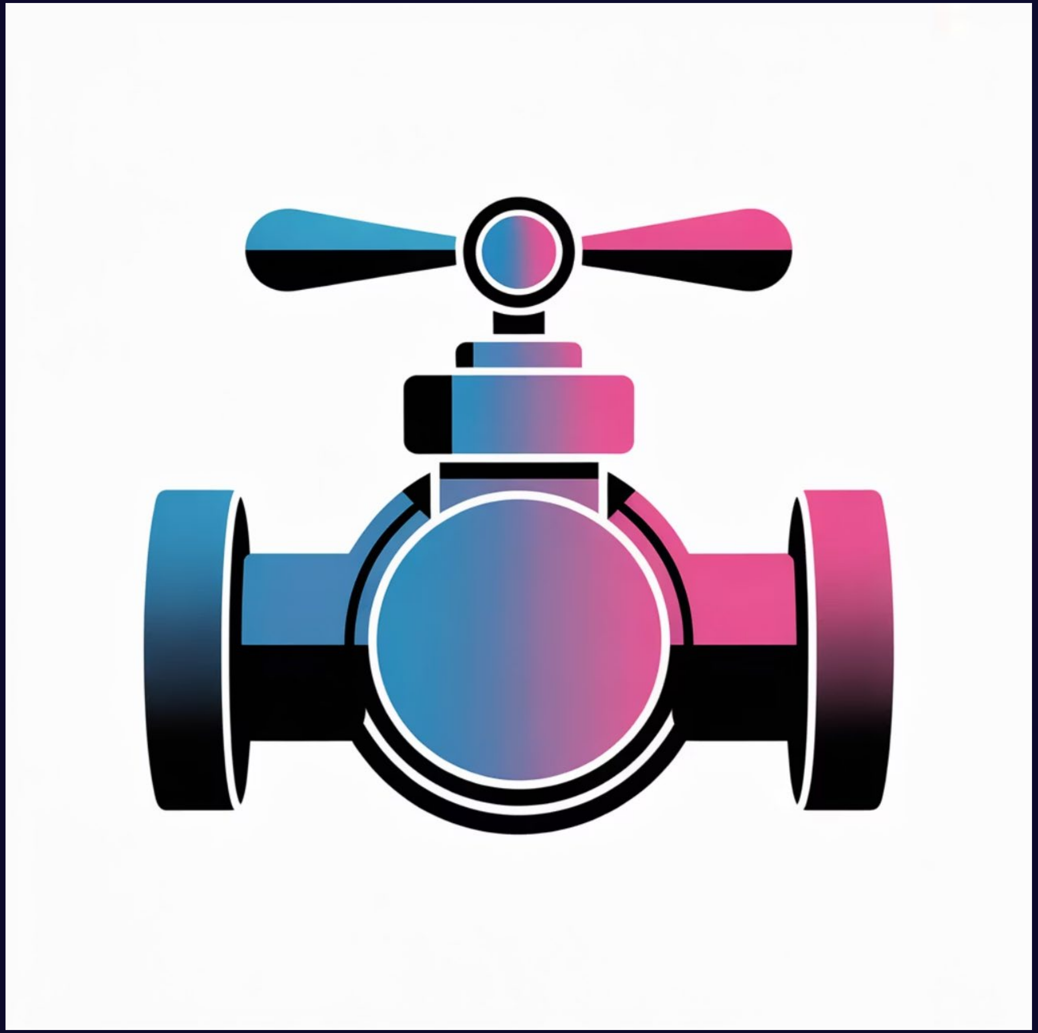
Why: This prevents people from accessing information they're not authorised to see.

Rule 2: Star Property (*-Property, "No Write Down")

The Rule: A subject cannot write information to a classification level lower than their current level.

Why: This prevents sensitive information from being leaked (intentionally or accidentally) to lower security levels.

Why Bell -LaPadula Is Important



Bell-LaPadula creates a oneway information valve. Once information reaches a higher classification, it cannot flow back down to less secure areas.

This prevents:

- Accidental disclosure of sensitive information
- Deliberate leaks by insiders
- Data mixing across security boundaries
- Classification errors that could expose secrets

Biba Integrity Model

Protecting Integrity

What It's For: The Biba model prevents corruption of trustworthy data. It's all about ensuring that reliable information stays reliable. This model focuses exclusively on integrity, not confidentiality.

The Integrity Hierarchy:

Information and subjects are assigned integrity levels (e.g., Untrusted, Low, Medium, High). Higher integrity means more trustworthy.

The Two Fundamental Rules of Biba

Rule 1: Simple Integrity Axiom ("No Read Down")

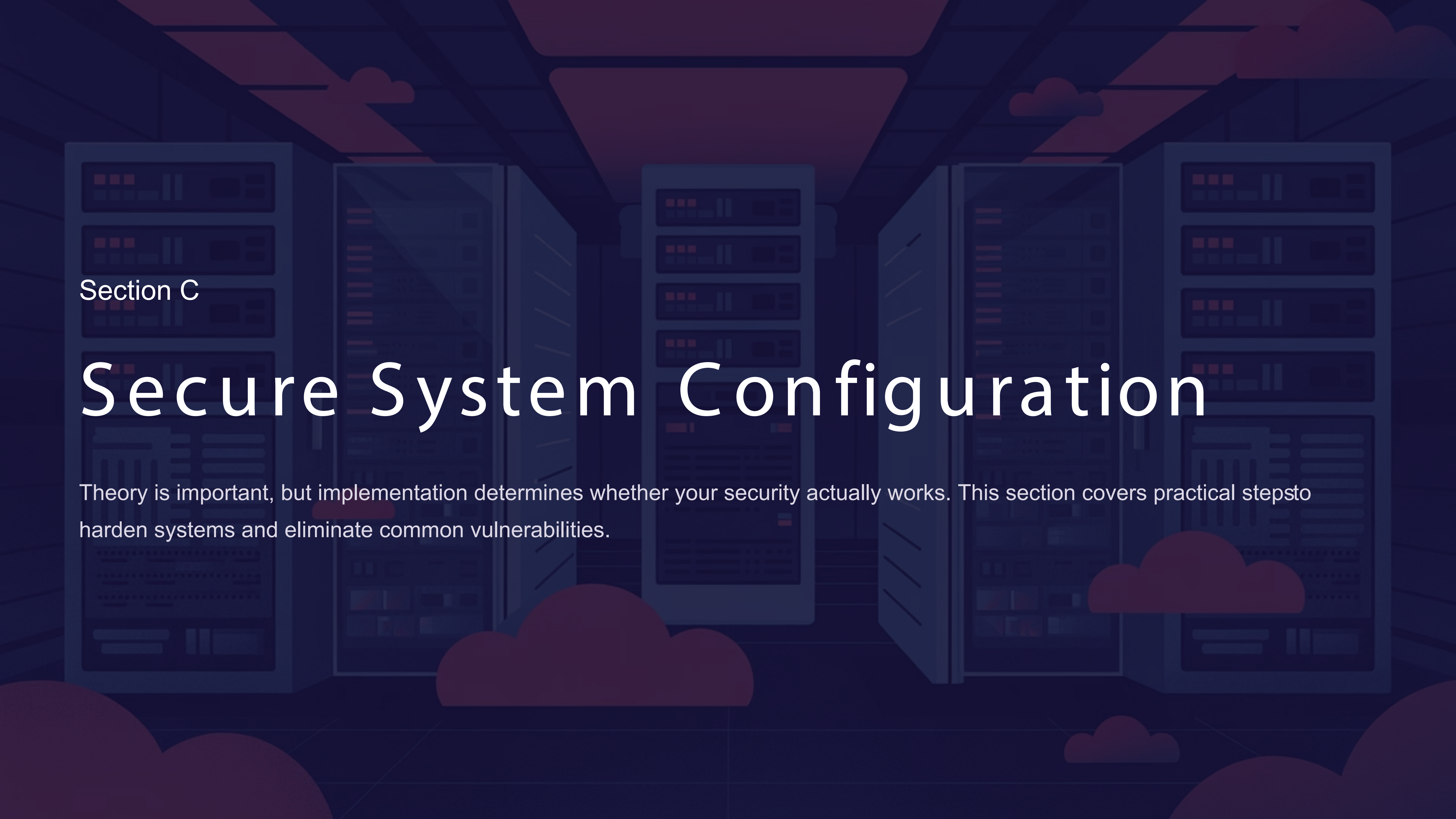
The Rule: A subject cannot read data from a lower integrity level than their own.

Why: This prevents trusted processes from being contaminated by unreliable or untrusted data.

Rule 2: Integrity Star Axiom (* -Integrity Property, "No Write Up")

The Rule: A subject cannot write to a higher integrity level than their own.

Why: This prevents less trusted processes from corrupting more trusted data or systems.



Section C

Secure System Configuration

Theory is important, but implementation determines whether your security actually works. This section covers practical stepsto harden systems and eliminate common vulnerabilities.

Practice 8

Define and Protect the Trusted Computing Base (TCB)

What the TCB Is: The Trusted Computing Base is the foundation of your system's security. It's the complete set of hardware, software, and security controls that work together to enforce your security policy. If the TCB fails, your entire security system fails.

Components of the TCB:

Hardware

Processors, memory, storage devices that handle security functions

Software

Operating system kernel, security utilities, authentication systems

Firmware

BIOS/UEFI, hardware security modules

Controls

Access control mechanisms, encryption systems, audit tools

The Reference Monitor — The Heart of the TCB

The Reference Monitor is the critical component that checks every single access request. It must have three essential properties:

1. Tamperproof

- Cannot be modified or disabled by users or attackers
- Protected from unauthorised changes
- Maintains its integrity even under attack

2. Always Invokable

- Cannot be bypassed
- Every access request must go through it
- No backdoors or alternate paths exist

3. Correctly Implemented

- Works exactly as designed
- No bugs or flaws in the code
- Properly enforces all security policies

How to Protect Your TCB:

1. **Identify TCB components** —Document everything that's part of your security foundation
2. **Minimise the TCB** —Keep it as small as possible (fewer components = fewer vulnerabilities)
3. **Implement strong access controls** —Restrict who can modify TCB components
4. **Monitor for tampering** —Watch for unauthorised changes to TCB elements
5. **Verify integrity regularly** —Use checksums, digital signatures, or other methods to ensure components haven't been altered
6. **Audit thoroughly** —Regularly review TCB configuration and operation
7. **Isolate from untrusted components** —Keep the TCB separate from less secure parts of the system

Critical Understanding: If any part of this system fails—the vault is breached, the lock is picked, or the procedures aren't followed—the bank's entire security fails. Similarly, if your TCB is compromised, attackers can bypass all your other security measures.