Chapter 1: Introduction and Overview

Dr. Mayank Agarwal

Department of CSE
IIT Patna

CS457 Big Data Security
Jan-April 2026

Security in Computing, Fifth Edition

Chapter 1: Introduction and Overview

- Course and Text Introduction
- The Big Questions: What is Security? Why is it Hard?
- Chapter Learning Objectives

Welcome. This course will use the foundational textbook *Security in Computing* by Pfleeger, Pfleeger, and Margulies. Chapter 1 sets the stage by defining the landscape, the core problems, and the mindset needed to understand cybersecurity. We are not just learning about tools, but about a way of thinking.

Learning Objectives for Chapter 1

- Understand the fundamental components of security: confidentiality, integrity, and availability (CIA).
- Recognize additional security goals: authenticity, accountability, non-repudiation.
- Understand the relationship between threats, vulnerabilities, attacks, and controls.
- Appreciate the fundamental principles of secure design.
- Grasp why security is inherently hard.

By the end of this chapter, you should be able to articulate the core triad of security, differentiate between key terms like vulnerability and threat, and recite the basic principles that guide secure system design. Most importantly, you'll see why perfect security is an unattainable goal, but effective security is not.

The Context: Why Is Security a Universal Concern?

- Computing is ubiquitous and networked.
- Systems hold data that is critical, valuable, and personal.
- Attackers range from casual to highly resourced nation-states.
- Consequences of failure include financial loss, privacy violation, physical harm, and loss of trust.

We are not securing isolated mainframes. We are securing a global, interconnected web of devices—from phones to power grids. The data within is the lifeblood of modern society. This makes every developer, administrator, and user a stakeholder in security.

Defining Security: It's More Than Just "Keeping Hackers Out"

- Security is about **managing risk**.
- It is a **property** of a system, not a product you add on.
- It involves ensuring that assets—data, functionality, systems—are protected from harm.
- It is a balance between protection and utility.

A perfectly secure system is one that is turned off and buried in concrete. That's useless. So, security is the process of making informed trade-offs to reduce risk to an acceptable level while preserving the system's usefulness.

The Cornerstone: The CIA Triad

- The three core security goals:
  1. **Confidentiality**
  2. **Integrity**
  3. **Availability**
- This triad forms the primary benchmark for evaluating security needs.

Nearly all security discussions start here. The CIA triad is the fundamental model. Different systems prioritize these elements differently. A military secret prioritizes Confidentiality, a bank transaction Integrity, and a 911 dispatch system Availability.

Confidentiality

- **Definition:** The property that information is not made available or disclosed to unauthorized individuals, entities, or processes.
- **About:** Secrecy, privacy.
- **Violated by:** Unauthorized access, interception, theft.

Confidentiality ensures that only authorized parties can read sensitive information. It's not just about external hackers; it's also about internal access controls. Does a customer service rep need to see *all* customer credit card numbers?

Integrity

- **Definition:** The property of accuracy, completeness, and non-alteration of assets.
- **Two Key Aspects:**
  - Data Integrity: Data is unchanged from its source.
  - System/Process Integrity: A system performs its intended function correctly.
- **Violated by:** Unauthorized modification, corruption, manipulation.

Integrity is about trustworthiness. If you receive an email from your boss, can you trust that it's exactly what they sent and hasn't been changed? If your car's braking system receives a "brake" command, can it trust the command is genuine?

Availability

- **Definition:** The property of a system or asset being accessible and usable upon demand by an authorized entity.
- **About:** Timely and reliable access.
- **Violated by:** Denial-of-Service (DoS) attacks, hardware failures, catastrophic events.

Often overlooked. An attacker doesn't need to steal or change your data to cause immense harm. Simply making your e-commerce website unreachable during the holiday shopping season can be catastrophic. Availability is a security requirement.

Beyond CIA: Additional Security Goals

- **Authenticity:** The property of being genuine and verifiable (e.g., proof of origin).
- **Accountability:** Actions can be traced uniquely to the responsible party.
- **Non-repudiation:** A party cannot falsely deny having taken an action (provides proof of origin and delivery).

CIA is the core, but modern systems need more. Authenticity asks, "Is this email really from the CEO?" Accountability ensures we know *who* made the change. Non-repudiation is crucial for e-commerce—you can't deny you placed an order if the log has your digital signature.

The Adversary's Perspective: Threats, Vulnerabilities, and Attacks

- **Asset:** Something of value to protect.
- **Threat:** A potential for violation of security. A *possible* danger.
- **Vulnerability:** A weakness in the system that could be exploited by a threat.
- **Attack (or Exploit):** The actualization of a threat; the act of exploiting a vulnerability.

This is the attack chain. A **threat** (e.g., a thief) targets an **asset** (your laptop). They succeed if the asset has a **vulnerability** (you left it unattended in a coffee shop) and they launch an **attack** (walk away with it). Security is about breaking this chain.

Control (or Countermeasure)

- **Definition:** An action, device, procedure, or technique that removes or reduces a vulnerability.
- **Function:** To block an attack, stop an ongoing attack, or reduce the impact of a successful attack.
- **Categories:** Preventive, Detective, Corrective, Deterrent, Recovery.

This is our defense. A **control** addresses a **vulnerability** to mitigate a **threat**. It's not just a firewall. A security policy is a control. Training is a control. A backup is a control. We will categorize these soon.

Threat Source / Threat Agent

- The entity (person, organization, natural event) that carries out a threat.
- **Motivations:** Financial gain, espionage, hacktivism, challenge, disruption.
- **Resources & Sophistication:** Varies widely from script kiddies to Advanced Persistent Threats (APTs).

Who is the adversary? Understanding their capability and intent is key to risk assessment. Defending against a nation-state APT requires different resources than defending against a casual scanner. Not all attackers are equal.

Risk: The Central Concept

- **Risk = Likelihood of a threat exploiting a vulnerability * Impact of the resulting loss.**
- Security is the practice of **Risk Management**.
- Goal: Reduce risk to an acceptable level, not to zero.

This is the most important business concept in security. We cannot prevent all bad things. We assess which risks are most likely and most damaging, and we apply our limited time and budget to control those first. This is a continuous process.

The "How" of Attacks: Methods and Techniques

- **Interception:** Threat to Confidentiality (eavesdropping).
- **Interruption:** Threat to Availability (DoS, destruction).
- **Modification:** Threat to Integrity (tampering).
- **Fabrication:** Threat to Authenticity & Integrity (spoofing, creation of false data).

These are the four fundamental classes of attack actions. Think of them as verbs. An attacker can **intercept** your password, **interrupt** your server, **modify** a transaction amount, or **fabricate** a login request. All attacks fit into these categories.

Classifying Controls: By Function (1)

- **Preventive:** Designed to stop an incident from occurring.
    - *Example:* Firewall, access control lock, encryption.
- **Detective:** Designed to discover and identify incidents while they are occurring or after.
    - *Example:* Intrusion Detection System (IDS), audit logs, checksums.

Preventive controls are the first line of defense—they keep the bad thing from happening. But we assume they will fail. Detective controls are the alarm system that tells us a preventive control has failed. We need both.

Classifying Controls: By Function (2)

- **Corrective / Recovery:** Reduce the impact of an incident and restore normal operations.
  - *Example:* Backups, system restoration procedures.
- **Deterrent:** Discourage an attacker by reducing their willingness to attack.
  - *Example:* Warning banners, laws, visible security cameras.

Corrective controls are your "plan B." If the attacker gets through prevention and detection, how do you recover? Deterrent controls are psychological. They don't technically stop an attack, but they might make an attacker choose an easier target.

Classifying Controls: By Nature of Action

- **Technical (Logical):** Implemented in hardware, software, or firmware.
    - *Example:* Encryption, access control lists, network segmentation.
- **Administrative (Managerial):** Policies, procedures, and guidelines for people.
    - *Example:* Security training, background checks, incident response plan.
- **Physical:** Tangible, real-world measures.
    - *Example:* Locked doors, security guards, cable shielding.

Effective security requires a mix of all three. The strongest technical encryption is useless if an attacker can **physically** steal the server or if an **administrative** failure gives the password to a phisher. This is called "defense in depth."

Fundamental Security Design Principles

- Guidelines for building secure systems, not just adding security on later.
- First articulated by Saltzer and Schroeder in the 1970s.
- They remain remarkably relevant today.

These are the "rules of thumb" for architects and developers. They guide us to design security *into* the system from the very beginning, which is far more effective and cheaper than bolting it on afterwards.

Principle 1: Least Privilege

- **Definition:** Every entity (user, process, program) should operate using the minimal set of privileges necessary to complete its job.
- **Rationale:** Limits the damage from an accident or attack. Contains the "blast radius."

This is arguably the most important principle. A user account for browsing the web should not have administrator rights. A program that needs to read a file should not have write or delete access. Start with zero privilege and add only what is essential.

Principle 2: Fail-Safe Defaults

- **Definition:** The default access permission to any resource should be "deny" unless explicitly permitted.
- **Rationale:** It is easier to manage a list of exceptions (who *can* access) than to list everyone who cannot.

A new user should have access to nothing by default. A new file should be inaccessible to others by default. This "whitelisting" approach is inherently more secure than a "blacklisting" approach where everything is allowed except a few banned items.

Principle 3: Economy of Mechanism

- **Definition:** Security designs should be as simple and small as possible.
- **Rationale:** Complexity is the enemy of security. Simple mechanisms are easier to verify, test, and trust.

A huge, complex security system will have hidden bugs and unexpected interactions. Think of a massive firewall rule set vs. a simple, well-understood one. Keep it simple. This applies to code, policies, and network architecture.

Principle 4: Complete Mediation

- **Definition:** Every access to every resource must be checked for authority. Don't rely on cached permissions.
- **Rationale:** Prevents an entity from retaining access after its privilege has been revoked.

If you check a user's access to a file once and then remember "they're allowed," what happens when their permission is revoked? The system must **re-check** on every single access attempt. This is crucial for dynamic environments.

Principle 5: Open Design

- **Definition:** The security of a system should not depend on the secrecy of its design or implementation.
- **Rationale:** Allows for public scrutiny, which leads to more robust designs (see Kerckhoffs's Principle).
- **Security through obscurity is a weak, supplemental control.**

This does NOT mean you publish your passwords. It means the *algorithm* should be public and the security should lie in a secret *key*. Hiding how a system works ("security by obscurity") is a terrible primary defense because once discovered, it's useless.

Principle 6: Separation of Privilege

- **Definition:** Where feasible, require two or more conditions to grant access (like multi-factor authentication).
- **Rationale:** Makes compromise more difficult. An attacker must defeat multiple, independent controls.

This is the principle behind "two keys to launch a missile" or "password + SMS code" for login. It's a specific application of "defense in depth." Don't rely on a single check or a single key.

Principle 7: Least Common Mechanism

- **Definition:** Minimize the amount of mechanism common to more than one user or system.
- **Rationale:** Limits the potential for unintended information flows and reduces the impact of a compromise in the shared mechanism.

Shared resources are a risk. If ten critical applications all depend on one central authentication server, compromising that one server takes down all ten. Where possible, isolate systems to prevent a single point of failure from becoming a catastrophe.

Principle 8: Psychological Acceptability

- **Definition:** Security mechanisms should not make the resource more difficult to access than if the mechanism were absent.
- **Rationale:** If security is too cumbersome, users will bypass it, creating a bigger vulnerability.

This is the human-factors principle. A password policy requiring a 30-character random string changed daily will lead to passwords on sticky notes. Security must be usable. The user interface is part of the security design.

The Principle of Weakest Link

- **Not an original Saltzer & Schroeder principle, but critical.**
- **Definition:** A system is only as secure as its weakest component.
- **Implication:** You must protect all parts of the system chain.

You can have a vault door, but if the wall is made of plywood, the system is weak. In cybersecurity, the "weakest link" is often the human user (susceptible to phishing), an unpatched application, or a misconfigured server. Attackers target the weakest point.

Why Is Security So Hard? Part 1: The Problem Space

- **The Defender's Dilemma:** Defender must protect *all* points; attacker needs only *one* weakness.
- **Asymmetry of Effort:** Defense is often more costly (time, money, performance) than attack.
- **Evolving Threat:** Attackers adapt quickly. Today's effective control may be obsolete tomorrow.

This is the fundamental challenge. We are playing a game where the attacker has the initiative and only has to succeed once. We have to be right every single time. This is why a purely preventive mindset is doomed to fail.

Why Is Security So Hard? Part 2: System Complexity

- Modern systems are staggeringly complex (hardware, OS, libraries, applications, network).
- Complexity leads to unintended interactions and hidden vulnerabilities (bugs).
- It is impossible to fully test or verify a complex system.

The codebase of a modern OS or web browser runs into tens of millions of lines. No human or team can understand it all. In that complexity, bugs—many of which are security flaws—are inevitable. We are building fortresses out of imperfect bricks.

Why Is Security So Hard? Part 3: The Human Factor

- Users are not security experts. They prioritize convenience.
- Social engineering attacks bypass technical controls entirely.
- Developers make mistakes that lead to vulnerabilities.
- Administrators misconfigure systems.

This is often the weakest link. A brilliant technical defense can be undone by one employee clicking a malicious link. Security must account for predictable human error and malicious manipulation of human psychology.

The "Gold Standard" of Evaluation: Assurance

- **Definition:** The degree of confidence that the security features and architecture of a system accurately mediate and enforce the security policy.
- **How?** Through rigorous analysis, testing, and formal verification.
- It answers: "How much do you trust this system to be secure?"

Assurance is about evidence. Saying "we have a firewall" provides little assurance. Providing independent test results, formal models, and verification of its configuration provides high assurance. It's the difference between a claim and a warranty.

Operational vs. Assurance Aspects

- **Operational (Features):** *What* the system does. (Encryption, access control, auditing).
- **Assurance (Trust):** *How well* it does it, and how confident we are that it cannot be bypassed.
- You need both. Features without assurance are untrustworthy.

A lock on a door is a feature. Assurance is knowing the lock is made of hardened steel, installed correctly, and that the doorframe is equally strong. In software, features are the security functions; assurance is the rigorous process that builds confidence they work correctly.

The Role of Policy

- **Security Policy:** A formal statement of the rules that asset owners must follow to protect the assets.
- It defines the "what" and "why" of security for an organization.
- All controls, mechanisms, and procedures are implementations of the policy.

Everything starts with policy. It is the high-level document that says, "Customer data shall be confidential." The technical controls (encryption) and administrative controls (training) are how we make that policy happen. No policy = ad-hoc, ineffective security.

Ethical Issues in Security

- Security professionals have immense power (access to data, ability to monitor).
- Key Questions:
  - What are the boundaries of defensive monitoring?
  - What is the responsibility for finding vulnerabilities?
  - What are the ethics of "hacking back"?
- **Core Principle:** Do no harm. Respect privacy and law.

This is a critical part of professional practice. Just because you *can* monitor every keystroke of an employee doesn't mean you *should*. Vulnerability disclosure has its own ethics—do you tell the vendor quietly or announce it publicly? These are real dilemmas.

A Historical Perspective: Security Waves

- **1st Wave (1970s-80s):** Physical and OS security for multi-user mainframes.
- **2nd Wave (1990s):** Network security and the rise of the firewall.
- **3rd Wave (2000s):** Application security (buffer overflows, SQLi).
- **4th Wave (Now):** Pervasive computing, supply chain, human-centric security.

Security problems evolve as technology evolves. We solved (some) problems from a previous wave, only to create new ones. We are now in an era where the attack surface includes cloud APIs, smartphone apps, and the software components we import from the internet.

The Total Cost of Security

- **Direct Costs:** Hardware, software, personnel.
- **Indirect Costs:** Performance overhead, usability impact, training.
- **Trade-off:** Investment vs. Risk Reduction.
- Goal is **cost-effective** security, not absolute security.

Security is an investment with a return (risk reduction). Spending $10 million to protect against a $100,000 risk is bad business. This is why risk analysis is the foundation of all sensible security management.

The Lifecycle of a Vulnerability

- 1. Vulnerability introduced (e.g., in design or code).
- 2. Vulnerability exists in deployed system.
- 3. Vulnerability discovered (by defender or attacker).
- 4. Exploit is created.
- 5. Attack occurs.
- 6. Patch created and deployed.
- Security aims to shorten this timeline and reduce impact.

The "window of exposure" is the time between step 3 (discovery) and step 6 (patch deployment). The defender's goal is to discover it first (through auditing/analysis) and patch it before attackers can widely exploit it. Zero-day attacks occur when the attacker finds it first (step 3) and uses it before the defender knows.

The Importance of a Holistic Approach

- Security is not just a technology problem.
- It requires a blend of:
  - **Technology** (Tools & Mechanisms)
  - **Processes** (Procedures & Management)
  - **People** (Training & Culture)
- Failure in any one area compromises the whole.

This is the mantra. You can buy the best security software, but if you have no process to manage it and your people are untrained, it will fail. This holistic view is often called the "people, process, technology" triad.

Key Terminology Review: Assets & Goals

- **Asset:** What we protect.
- **Confidentiality, Integrity, Availability (CIA):** The core goals.
- **Authenticity, Accountability, Non-repudiation:** Extended goals.

Ensure you are completely comfortable with these terms. They are the language of security. Every future discussion will refer back to protecting an asset's CIA properties.

Key Terminology Review: The Attack Model

- **Threat:** A potential cause of an unwanted incident.
- **Vulnerability:** A weakness that can be exploited.
- **Attack/Exploit:** The act of exploiting the vulnerability.
- **Risk:** The potential for loss (Likelihood * Impact).

These terms have specific meanings. Don't use them interchangeably. A hurricane is a *threat*. A building with no storm shutters has a *vulnerability*. The hurricane hitting is the *attack*. The *risk* is the probability of the hit times the cost of damage.

Key Terminology Review: Our Defenses

- **Control/Countermeasure:** What we put in place to reduce risk.
- **Preventive, Detective, Corrective, Deterrent:** Control functions.
- **Technical, Administrative, Physical:** Control natures.

Classifying controls helps us build a balanced, layered defense (defense in depth). We need controls that function in different ways and are implemented across different domains.

Key Principles Review (1-4)

- 1. **Least Privilege:** Minimal rights needed.
- 2. **Fail-Safe Defaults:** Default = deny.
- 3. **Economy of Mechanism:** Keep it simple.
- 4. **Complete Mediation:** Check every access, every time.

These four principles are often the most directly applicable to system design. When reviewing any design, ask: Are privileges minimized? Are defaults secure? Is it unnecessarily complex? Does it check permissions consistently?

Key Principles Review (5-8)

- 5. **Open Design:** Security not in secrecy of design.
- 6. **Separation of Privilege:** Require multiple conditions.
- 7. **Least Common Mechanism:** Minimize shared resources.
- 8. **Psychological Acceptability:** Usable security.
- + **Weakest Link:** A chain is only as strong as its weakest link.

These principles cover trust, redundancy, isolation, and human factors. Psychological Acceptability is critical for real-world success. And always remember the Weakest Link—find and fortify it.

The Security Mindset

- Think like an adversary. Ask: "How can this be broken?"
- Assume failures will happen. Plan for detection and recovery.
- Embrace paranoia, tempered by risk management.
- Trust, but verify.

This is the core takeaway. Adopt a skeptical, inquisitive, and resilient mindset. Don't just see how a system works; see how it *fails*. This mindset will guide you through the technical details of the coming chapters.

Chapter 1 Summary: The Big Picture

- Security is about **managing risk** to protect **assets**.
- Goals are defined by **CIA+**.
- We defend against **threats** that exploit **vulnerabilities** via **attacks** using **controls**.
- Design is guided by **time-tested principles**.
- It is hard due to **asymmetry, complexity, and human factors**.

You now have the framework. Everything else in this book—cryptography, software security, networks, management—fills in the details of this framework. You understand the *what* and *why*. Now we will dive into the *how*.

Looking Ahead: What's Next?

- **Chapter 2: Toolbox: Authentication, Access Control, and Cryptography.**
- We will start building our toolkit of security mechanisms.
- These are the foundational *technical controls* that implement our security goals.

With our overview complete, we move to the essential tools. Chapter 2 covers the bedrock: proving identity (authentication), controlling what they can do (access control), and protecting data with math (cryptography).

Critical Thinking Questions (1)

- For a public website like a news outlet, rank the importance of Confidentiality, Integrity, and Availability. Justify your ranking.
- Describe a simple system (e.g., a library). Identify one asset, one threat, one vulnerability, and a possible control.

Think about these. For the news site, availability is paramount (people need to read it), integrity is critical (you can't let someone change the news), confidentiality is lower (most news is public). This prioritization drives security investments.

Critical Thinking Questions (2)

- How does the principle of "Psychological Acceptability" explain the failure of very strict password policies?
- Give an example of a preventive, a detective, and a corrective control for protecting the data on your personal laptop.

Strict policies lead to writing down passwords, which is worse. For your laptop: Preventive = Full disk encryption. Detective = Anti-virus scan. Corrective = Restore files from backup.

Resources and Further Reading

- Pfleeger, C. P., Pfleeger, S. L., & Margulies, J. (2015). *Security in Computing, 5th Ed.* Prentice Hall.
- Saltzer, J. H., & Schroeder, M. D. (1975). The Protection of Information in Computer Systems.
- CERT Coordination Center: www.sei.cmu.edu/about/divisions/cert/index.cfm

The Saltzer & Schroeder paper is a classic, still worth reading. CERT is a phenomenal real-world resource for current vulnerabilities and best practices.

End of Chapter 1

Introduction and Overview

Questions? Next: Chapter 2 - Toolbox

Thank you!