**Confidentiality:-**

Confidentiality is part of information security and keeping data and resources secret and hidden. Cryptography and access control mechanism supports confidentiality. Information possesses confidentiality when it is accessible only to those who are authorized to access it. Conversely, information lacks confidentiality to the extent that it is available or disclosed to unauthorized persons or processes. The need for keeping information secret arises from the use of computers in sensitive fields:- such as government , industry, Bank , Telecommunication ..etc. The first formal work in computer security was motivated by the military’s attempt to implement controls to enforce a “need to know” principal.

A good example is cryptography, which traditionally is used to protect secret messages. But cryptography is traditionally used to protect data, not resources. Resources are protected by limiting information, for example by using firewalls or address translation mechanism.

Resource hiding is important aspect of confidentiality. Sites often wish to conceal their configuration as well as what systems they are using; organizations may not wish others to know about specific equipment (because it could be used without authorization or in inappropriate ways), and a company renting time from a service provider may not want others to know what resources it is using.

Access control mechanisms provide these capabilities as well.

**Integrity :-**

Integrity refers to the trustworthiness of data or resources, and it is usually phrased in terms of preventing improper or unauthorized change. Integrity includes data integrity ( the content of the information) and origin integrity( the source of the data, often called authentication). Integrity mechanism fall into two classes:

* Prevention mechanism and
* Detection mechanism

Prevention mechanism seek to maintain the integrity of the data by blocking any unauthorized attempts to change the data or any attempts to change the data in unauthorized ways.

Detection mechanisms do not try to prevent violations of integrity; they simply report that the data’s integrity is no longer trustworthy. Detection mechanisms may analyze system events (user or system actions) to detect problems or (more commonly) may analyze the data itself to see if required or expected constraints still hold. The mechanisms may report the actual cause of the integrity violation (a specific part of a file was altered), or they may simply report that the file is now corrupt. Working with integrity is very different from working with confidentiality.

With confidentiality, the data is either compromised or it is not, but integrity includes both the correctness and the trustworthiness of the data. The origin of the data (how and from whom it was obtained), how well the data was protected before it arrived at the current machine, and how well the data is protected on the current machine all affect the integrity of the data. Thus, evaluating integrity is often very difficult, because it relies on assumptions about the source of the data and about trust in that source—two underpinnings of security that are often overlooked.

Note:- Ensuring authorized modifications; Includes correctness and trustworthiness. May refer to: Data integrity , Origin integrity

Integrity: a good example here is that of an interrupted database transaction, leaving the database in an inconsistent state (this foreshadows the Clark-Wilson model). Trustworthiness of both data and origin affects integrity.That integrity is tied to trustworthiness makes it much harder to quantify than confidentiality. Cryptography provides mechanisms for detecting violations of integrity, but not preventing them (e.g., a digital signature can be used to determine if data has changed).

**Availability**

Availability refers to the ability to use the information or resource desired. Availability is an important aspect of reliability as well as of system design because an unavailable system is at least as bad as no system at all. The aspect of availability that is relevant to security is that someone may deliberately arrange to deny access to data or to a service by making it unavailable. System designs usually assume a statistical model to analyze expected patterns of use, and mechanisms ensure availability when that statistical model holds. Someone may be able to manipulate use (or parameters that control use, such as network traffic) so that the assumptions of the statistical model are no longer valid. This means that the mechanisms for keeping the resource or data available are working in an environment for which they were not designed. As a result, they will often fail.

* + Note: Ensuring authorized access to data and resources when desired, this is usually defined in terms of “quality of service,” in which authorized users are expected to receive a specific level of service (stated in terms of a metric). Denial of service attacks are attempts to block availability.

(Additional from NIST)

* Accountability
  + Ensuring that an entity’s action is traceable uniquely to that entity
* Security assurance
  + Assurance that all four objectives are met
* A threat is a “potential” violation of security
  + The violation need not actually occur
  + The fact that the violation *might* occur makes it a threat
  + It is important to guard against threats and be prepared for the actual violation

The actual violation of security is called an attack

**Common Security Attacks** :-

* Interruption, delay, denial of receipt or denial of service
  + System assets or information become unavailable or are rendered unavailable
* Interception or snooping
  + Unauthorized party gains access to information by browsing through files or reading communications
* Modification or alteration
  + Unauthorized party changes information in transit or information stored for subsequent access
* Fabrication, masquerade, or spoofing
  + Spurious information is inserted into the system or network by making it appear as if it is from a legitimate entity
* Repudiation of origin
  + False denial that an entity created something

Classes of Threat :-

* Disclosure: *unauthorized access to information*
  + Snooping
* Deception: *acceptance of false data*
  + Modification, masquerading/spoofing, repudiation of origin, denial of receipt
* Disruption: *interruption/prevention of correct operation*
  + Modification
* Usurpation: *unauthorized control of a system component*
  + Modification, masquerading/spoofing, delay, denial of service

Snooping : an example is passive wiretapping, where the attacker monitors communications.

Modification: an example is active wiretapping, where the attacker injects something into a communication or modifies parts of the communication. Modification is sometimes called alteration.

Spoofing: delegation is basically authorized spoofing. The difference is that the ones to which authority is delegated does not impersonate the delegator; she simply asserts authority to act as an agent for the delegator.

Delay:

Denial of service: this may not be due to an attack, but due to limits of resources. However, the effect here is critical. If you define security in terms of what users need to access, the inability to access is a security problem regardless of whether the reason is intentional (an attack) or unintentional (not an attack).

**Policy and Mechanism:-**

* A security policy states what is, and is not, allowed
  + This defines “security” for the site/system/*etc*.
  + Policy definition: Informal? Formal?

Policy: may be expressed in

* natural language, which is usually imprecise but easy to understand;
* mathematics, which is usually precise but hard to understand;
* policy languages, which look like some form of programming language and try to balance precision with ease of understanding
* Mechanisms enforce policies
* Composition of policies
  + If policies conflict, discrepancies may create security vulnerabilities

Mechanisms: may be

* technical, in which controls in the computer enforce the policy; for example, the requirement that a user supply a password to authenticate herself before using the computer
* procedural, in which controls outside the system enforce the policy; for example, firing someone for ringing in a disk containing a game program obtained from an untrusted source

The composition problem requires checking for inconsistencies among policies. If, for example, one policy allows students and faculty access to all data, and the other allows only faculty access to all the data, then they must be resolved (e.g., partition the data so that students and faculty can access some data, and only faculty access the other data).

**Goal of Security :-**

* Prevention
  + To prevent someone from violating a security policy

Prevention is ideal, because then there are no successful attacks.

* Detection
  + To detect activities in violation of a security policy
  + Verify the efficacy of the prevention mechanism

Detection occurs after someone violates the policy. The mechanism determines that a violation of the policy has occurred (or is underway), and reports it. The system (or system security officer) must then respond appropriately.

* Recovery
  + Stop policy violations (attacks)
  + Assess and repair damage
  + Ensure availability in presence of an ongoing attack
  + Fix vulnerabilities for preventing future attack
  + Retaliation against the attacker

Recovery means that the system continues to function correctly, possibly after a period during which it fails to function correctly. If the system functions correctly always, but possibly with degraded services, it is said to be intrusion tolerant. This is very difficult to do correctly; usually, recovery means that the attack is stopped, the system fixed (which may involve shutting down the system for some time, or making it unavailable to all users except the system security officers), and then the system resumes correct operations.

**Assumptions and Trust:-**

* Policies and mechanisms have implicit assumptions
* Assumptions regarding policies
  + Unambiguously partition system states into “secure” and “nonsecure” states
  + Correctly capture security requirements

All security policies and mechanisms rest on assumptions; we’ll examine some in later chapters, Malicious Logic. Here is a taste of the assumptions.

Policies: as these define security, they have to define security correctly for the particular site. For example, a web site has to be available, but if the security policy does not mention availability, the definition of security is inappropriate for the site. Also, a policy may not specify whether a particular state is “secure” or “non-secure.” This ambiguity causes problems.

* Mechanisms
  + Assumed to enforce policy; i.e., ensure that the system does not enter “nonsecure” state
  + Support mechanisms work correctly

Mechanisms: as these enforce policy, they must be appropriate. For example, cryptography does not assure availability, so using cryptography in the above situation won’t work. Further, security mechanisms rely on supporting infrastructure, such as compilers, libraries, the hardware, and networks to work correctly.

**Types of Mechanism**

* Let *P* be the set of all the reachable states
* Let *Q* be a set of secure states identified by a policy: *Q* ⊆ *P*
* Let the set of states that an enforcement mechanism restricts a system to be *R*
* The enforcement mechanism is
  + Secure if *R* ⊆ *Q*
  + Precise if *R* = *Q*
  + Broad if there are some states in R that are not in Q

A reachable state is one that the computer can enter. A secure state is a state defined as allowed by the security policy.

The left figure shows a secure system: all reachable states are in the set of secure states. The system can never enter (reach) a non-secure state, but there are secure states that the system cannot reach.

The middle figure shows a precise system: all reachable states are secure, and all secure states are reachable. Only the non-secure states are unreachable.

The right figure shows a broad system. Some non-secure states are reachable. This system is also not secure.

**Information Assurance:-**

* *Information Assurance Advisory Council (IAAC)*:

“Operations undertaken to protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality and non-repudiation”

* National Institute of Standards Technology

“Assurance is the basis for confidence that the security measures, both technical and operational, work as intended to protect the system and the information it processes”

**Assurance :-**

* Assurance is to indicate “how much” to trust a system and is achieved by ensuring that
  + The required functionality is present and correctly implemented
  + There is sufficient protection against unintentional errors
  + There is sufficient resistance to intentional penetration or by-pass
* Basis for determining this aspect of trust
  + Specification
    - Requirements analysis
    - Statement of desired functionality
  + Design
    - Translate specification into components that satisfy the specification
  + Implementation
    - Programs/systems that satisfy a design

Assurance is a measure of how well the system meets its requirements; more informally, how much you can trust the system to do what it is supposed to do. It does not say what the system is to do; rather, it only covers how well the system does it.

Specifications arise from requirements analysis, in which the goals of the system are determined. The specification says what the system must do to meet those requirements. It is a statement of functionality, not assurance, and can be very formal (mathematical) or informal (natural language). The specification can be high-level or low-level (for example, describing what the system as a whole is to do vs. what specific modules of code are to do).

The design architects the system to satisfy, or meet, the specifications. Typically, the design is layered by breaking the system into abstractions, and then refining the abstractions as you work your way down to the hardware. An analyst also must show the design matches the specification.

The implementation is the actual coding of the modules and software components. These must be correct (perform as specified), and their aggregation must satisfy the design.

Note the assumptions of correct compilers, hardware, *etc*.

**Operational Issues :-**

* Cost-Benefit Analysis
  + Benefits vs. total cost
  + Is it cheaper to prevent or recover?
* Risk Analysis
  + Should we protect something?
  + How much should we protect this thing?
  + Risk depends on environment and change with time
* Laws and Customs
  + Are desired security measures illegal?
  + Will people do them?
  + Affects availability and use of technology

Security does not end when the system is completed. Its operation affects security. A “secure” system can be breached by improper operation (for example, when accounts with no passwords are created). The question is how to assess the effect of operational issues on security.

Cost-Benefit Analysis: this weighs the cost of protecting data and resources with the costs associated with losing the data. Among the considerations are the overlap of mechanisms’ effects (one mechanism may protect multiple services, so its cost is amortized), the non-technical aspects of the mechanism (will it be impossible to enforce), and the ease of use (if a mechanism is too cumbersome, it may cost more to retrofit a decent user interface than the benefits would warrant).

Risk Analysis: what happens if the data and resources are compromised? This tells you what you need to protect and to what level. Cost-benefit analyses help determine the risk here, but there may be other metrics involved (such as customs).

Laws and Customs: these constrain what you can do. Encryption used to be the biggie here, as the text indicates. How much that has changed is anybody’s guess. Customs involve non-legislated things, like the use of urine specimens to determine identity. That is legal, at least in the US in some cases; but it would never be widely accepted as an alternative to a password.

**Human Issues :-**

* Organizational Problems
  + Power and responsibility
  + Financial benefits
* People problems
  + Outsiders and insiders
  + Social engineering

Organizations: the key here is that those responsible for security have the power to enforce security. Otherwise there is confusion, and the architects need not worry if the system is secure because they won’t be blamed if someone gets in. This arises when system administrators, for example, are responsible for security, but only security officers can make the rules. Preventing this problem (power without responsibility, or vice versa) is tricky and requires capable management. What’s worse is that security is not a direct financial incentive for most companies because it doesn’t bring in revenue. It merely prevents the loss of revenue obtained from other sources.

People problems: are by far the main source of security problems. Outsiders are attackers from without the organization; insiders are people who have authorized access to the system and, possibly, are authorized to access data and resources, but use the data or resources in unauthorized ways. It is speculated that insiders account for 80-90% of all security problems, but the studies generally do not disclose their methodology in detail, so it is hard to know how accurate they are. (Worse, there are many slightly different definitions of the term “insider,” causing the studies to measure slightly different things!) Social engineering, or lying, is quite effective, especially if the people gulled are inexperienced in security (possibly because they are new, or because they are tired).

**Security policy :-**

* Policy partitions system states into:
  + Authorized (secure)
    - These are states the system can enter
  + Unauthorized (nonsecure)
    - If the system enters any of these states, it’s a security violation
* Secure system
  + Starts in authorized state
  + Never enters unauthorized state

**Confidentiality :-**

* *X* set of entities, *I* information
* *I* has *confidentiality* property with respect to *X* if no *x* ∈ *X* can obtain information from *I*
* *I* can be disclosed to others
* Example:
  + *X* set of students
  + *I* final exam answer key
  + *I* is confidential with respect to *X* if students cannot obtain final exam answer key

**Integrity :-**

* *X* set of entities, *I* information
* *I* has *integrity* property with respect to *X* if all *x* ∈ *X* trust information in *I*
* Types of integrity:
  + trust *I*, its conveyance and protection (data integrity)
  + *I* information about origin of something or an identity (origin integrity, authentication)
  + *I* resource: means resource functions as it should (assurance)

**Availability :-**

* *X* set of entities, *I* resource
* *I* has *availability* property with respect to *X* if all *x* ∈ *X* can access *I*
* Types of availability:
  + traditional: *x* gets access or not
  + quality of service: promised a level of access (for example, a specific level of bandwidth) and not meet it, even though some access is achieved

**Policy Models:-**

* Abstract description of a policy or class of policies
* Focus on points of interest in policies
  + Security levels in multilevel security models
  + Separation of duty in Clark-Wilson model
  + Conflict of interest in Chinese Wall model

**Types of Security Policies:-**

* Military (governmental) security policy
  + Policy primarily protecting confidentiality
* Commercial security policy
  + Policy primarily protecting integrity
* Confidentiality policy
  + Policy protecting only confidentiality
* Integrity policy
  + Policy protecting only integrity

**Integrity and Transactions:-**

* Begin in consistent state
  + “Consistent” defined by specification
* Perform series of actions (*transaction*)
  + Actions cannot be interrupted
  + If actions complete, system in consistent state
  + If actions do not complete, system reverts to beginning (consistent) state

**Trust:-**

Administrator installs patch

1. Trusts patch came from vendor, not tampered with in transit
2. Trusts vendor tested patch thoroughly
3. Trusts vendor’s test environment corresponds to local environment
4. Trusts patch is installed correctly

**Trust in Formal Verification**

* Gives formal mathematical proof that given input *i*, program *P* produces output *o* as specified
* Suppose a security-related program *S* formally verified to work with operating system *O*
* What are the assumptions?

**Trust in Formal Methods:-**

1. Proof has no errors
   * Bugs in automated theorem provers
2. Preconditions hold in environment in which *S* is to be used
3. *S* transformed into executable *S′* whose actions follow source code
   * Compiler bugs, linker/loader/library problems
4. Hardware executes *S′* as intended
   * Hardware bugs (Pentium f00f bug, for example)

**Types of Access Control :-**

* Discretionary Access Control (DAC, IBAC)
  + individual user sets access control mechanism to allow or deny access to an object
* Mandatory Access Control (MAC)
  + system mechanism controls access to object, and individual cannot alter that access
* Originator Controlled Access Control (ORCON)
  + originator (creator) of information controls who can access information