Security Principles
Day 18/365

To expand on the CIA triad that I went over in the arcX course we can add 2 more points

- **Authenticity**: Authentic means not fraudulent or counterfeit. Authenticity is about ensuring that the document/file/data is from the claimed source.
- **Nonrepudiation**: Repudiate means refusing to recognize the validity of something. Nonrepudiation ensures that the original source cannot deny that they are the source of a particular document/file/data. This characteristic is indispensable for various domains, such as shopping, patient diagnosis, and banking.

And 2 more described by Donn Parker in the Parkerian Hexad, 1998

- **Utility**: Utility focuses on the usefulness of the information. For instance, a user might have lost the decryption key to access a laptop with encrypted storage. Although the user still has the laptop with its disk(s) intact, they cannot access them. In other words, although still available, the information is in a form that is not useful, i.e., of no utility.
- **Possession**: This security element requires that we protect the information from unauthorized taking, copying, or controlling. For instance, an adversary might take a backup drive, meaning we lose possession of the information as long as they have the drive. Alternatively, the adversary might succeed in encrypting our data using ransomware; this also leads to the loss of possession of the data

The opposite of the CIA Triad would be the DAD Triad: Disclosure, Alteration, and Destruction.

- Disclosure: As in most modern countries, healthcare providers must maintain medical records' confidentiality. As a result, if an attacker succeeds in stealing some of these medical records and dumping them online to be viewed publicly, the health care provider will incur a loss due to this data disclosure attack.
- Alteration: Consider the gravity of the situation if the attacker manages to modify patient medical records. This alteration attack might lead to the wrong treatment being administered, and consequently, this alteration attack could be life-threatening.
- Destruction/Denial: Consider the case where a medical facility has gone completely paperless. If an attacker manages to make the database systems unavailable, the facility will not be able to function properly. They can go back to paper temporarily; however, the patient records won't be available. This denial attack would stall the whole facility.

Security Models

Bell-LaPadula Model

Aim: Achieve confidentiality

- **Simple Security Property**: This property is referred to as “no read up”; it states that a subject at a lower security level cannot read an object at a higher security level. This rule prevents access to sensitive information above the authorized level.
- **Star Security Property**: This property is referred to as “no write down”; it states that a subject at a higher security level cannot write to an object at a lower security level. This rule prevents the disclosure of sensitive information to a subject of lower security level.
- **Discretionary-Security Property**: This property uses an access matrix to allow read and write operations. An example access matrix is shown in the table below and used in conjunction with the first two properties.

Subjects	Object A	Object B
Subject 1	Write	No access
Subject 2	Read/Write	Read

The first two properties can be summarized as “write up, read down.” You can share confidential information with people of higher security clearance (write up), and you can receive confidential information from people with lower security clearance (read down).

The model was not designed to handle file-sharing

Biba Model

Aim: Achieve integrity

- **Simple Integrity Property**: This property is referred to as “no read down”; a higher integrity subject should not read from a lower integrity object.
- **Star Integrity Property**: This property is referred to as “no write up”; a lower integrity subject should not write to a higher integrity object.

Limitations: it does not handle internal threats (insider)

Clark-Wilson Model

Aim: Achieve integrity

- **Constrained Data Item (CDI)**: This refers to the data type whose integrity we want to preserve.
- **Unconstrained Data Item (UDI)**: This refers to all data types beyond CDI, such as user and system input.
- **Transformation Procedures (TPs)**: These procedures are programmed operations, such as read and write, and should maintain the integrity of CDIs.
- **Integrity Verification Procedures (IVPs)**: These procedures check and ensure the validity of CDIs.

Other models

- Brewer and Nash model
- Goguen-Meseguer model
- Sutherland model
- Graham-Denning model
- Harrison-Ruzzo-Ullman model

Defence In Depth

refers to creating a security system of multiple levels

Consider the following analogy: you have a locked drawer where you keep your important documents and pricey stuff. The drawer is locked; however, do you want this drawer lock to be the only thing standing between a thief and your expensive items? If we think of multi-level security, we would prefer that the drawer be locked, the relevant room be locked, the main door of the apartment be locked, the building gate be locked, and you might even want to throw in a few security cameras along the way. Although these multiple levels of security cannot stop every thief, they would block most of them and slow down the others.

ISO/ IEC 19249

Created by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC)

'ISO/IEC 19249 :2017 *Information technology - Security techniques - Catalogue of architectural and design principles for secure products, systems and applications*'

Five architectural principles:

- Domain Separation: Every set of related components is grouped as a single entity, they can be apps, data, or others. Each will have its own domain and be assigned a common set of security attributes.
- Layering: System is structured into many abstract levels, it becomes possible to impose security policies at different levels. i.e. OSI
- Encapsulation: OOP principle, data should not leave the object responsible for it.
- Redundancy: Ensures availability and integrity, you WANT redundancy, i.e. a hardware server with two built-in power supplies
- Virtualization: remote machines.

Five design principles:

- Least Privilege: need-to-know basis, answer the question "who can access what?". You should provide the least amount of permissions for someone to carry out a task and nothing more. i.e. if a user needs to be able to view a document, you should give them read rights and no write rights.
- Attack Surface Minimisation: Explained in arcX module.
- Centralized Parameter Validation: Many threats are due to the system receiving input. Invalid inputs can be used to exploit vulnerabilities in the system. A parameter validation is necessary to ensure the correct system state
- Centralized General Security Services: As a security principle, we should aim to centralize all security services. For example, we would create a centralized server for authentication. Of course, you might take proper measures to ensure availability and prevent creating a single point of failure.
- **Preparing for Error and Exception Handling**: Whenever we build a system, we should take into account that errors and exceptions do and will occur. For instance, in a shopping application, a customer might try to place an order for an out-of-stock item. A database might get overloaded and stop responding to a web application. This principle teaches that the systems should be designed to fail safe; for example, if a firewall crashes, it should block all traffic instead of allowing all traffic. Moreover, we should be careful that error messages don't leak information that we consider confidential, such as dumping memory content that contains information related to other customers.

Zero Trust versus Trust but Verify

We start from the fact that you cannot distrust everyone

Trust but Verify: This principle teaches that we should always verify even when we trust an entity and its behaviour. An entity might be a user or a system. Verifying usually requires setting up proper logging mechanisms; verifying indicates going through the logs to ensure everything is normal. In reality, it is not feasible to verify everything; just think of the work it takes to review all the actions taken by a single entity, such as Internet pages browsed by a single user. This requires automated security mechanisms, such as proxy, intrusion detection, and intrusion prevention systems.

Zero Trust: This principle treats trust as a vulnerability, and consequently, it caters to insider-related threats. After considering trust as a vulnerability, zero trust tries to eliminate it. It is teaching indirectly, "never trust, always verify." In other words, every entity is considered adversarial until proven otherwise. Zero trust does not grant trust to a device based on its location or ownership. This approach contrasts with older models that would trust internal networks or enterprise-owned devices. Authentication and authorization are required before accessing any resource. As a result, if any breach occurs, the damage would be more contained if a zero trust architecture had been implemented.

Threat vs Risk

Don't confuse ->

- **Vulnerability**: Vulnerable means susceptible to attack or damage. In information security, a vulnerability is a weakness.
- **Threat**: A threat is a potential danger associated with this weakness or vulnerability.
- **Risk**: The risk is concerned with the likelihood of a threat actor exploiting a vulnerability and the consequent impact on the business.