TryHackMe Walkthrough: Security Principles

Introduction

Security has become a buzzword; every company wants to claim its product or service is secure. But is it?

Before we start discussing the different security principles, it is vital to know the adversary against whom we are protecting our assets. Are you trying to stop a toddler from accessing your laptop? Or are you trying to protect a laptop that contains technical designs worth millions of dollars? Using the exact protection mechanisms against toddlers and industrial espionage actors would be ludicrous. Consequently, knowing our adversary is a must so we can learn about their attacks and start implementing appropriate security controls.

It is impossible to achieve perfect security; no solution is 100% secure. Therefore, we try to improve our security posture to make it more difficult for our adversaries to gain access.

The objective of this room is to:

- Explain the security functions: Confidentiality, Integrity and Availability (CIA).
- Present the opposite of the security triad, CIA: Disclosure, Alteration, and Destruction/Denial (DAD).
- Introduce the fundamental concepts of security models, such as the Bell-LaPadula model.
- Explain security principles such as Defence-in-Depth, Zero Trust, and Trust but Verify.
- Introduce ISO/IEC 19249.
- Explain the difference between Vulnerability, Threat, and Risk.

Answer the questions below

Think how you would describe something as secure. (no answer)

CIA

Before we can describe something as *secure*, we need to consider better what makes up security. When you want to judge the security of a system, you need to think in terms of the security triad: confidentiality, integrity, and availability (CIA).

- **Confidentiality** ensures that only the intended persons or recipients can access the data.
- **Integrity** aims to ensure that the data cannot be altered; moreover, we can detect any alteration if it occurs.
- **Availability** aims to ensure that the system or service is available when needed.

Let's consider the CIA security triad in the case of placing an order for online shopping:

- **Confidentiality**: During online shopping, you expect your credit card number to be disclosed only to the entity that processes the payment. If you doubt that your credit card information will be disclosed to an untrusted party, you will most likely refrain from continuing with the transaction. Moreover, if a data breach results in the disclosure of personally identifiable information, including credit cards, the company will incur huge losses on multiple levels.
- **Integrity**: After filling out your order, if an intruder can alter the shipping address you have submitted, the package will be sent to someone else. Without data integrity, you might be very reluctant to place your order with this seller.
- **Availability**: To place your online order, you will either browse the store's website or use its official app. If the service is unavailable, you won't be able to browse the products or place an order. If you continue to face such technical issues, you might eventually give up and start looking for a different online store.

Let's consider the CIA as it relates to patient records and related systems:

- **Confidentiality**: According to various laws in modern countries, healthcare providers must ensure and maintain the confidentiality of medical records. Consequently, healthcare providers can be held legally accountable if they illegally disclose their patients' medical records.
- **Integrity**: If a patient record is accidentally or maliciously altered, it can lead to the wrong treatment being administered, which, in turn, can lead to a life-threatening situation. Hence, the system would be useless and potentially harmful without ensuring the integrity of medical records.
- **Availability**: When a patient visits a clinic to follow up on their medical condition, the system must be available. An unavailable system would mean that the medical practitioner cannot access the patient's records and consequently won't know if any current symptoms are related to the patient's medical history. This situation can make the medical diagnosis more challenging and error-prone.

The emphasis does not need to be the same on all three security functions. One example would be a university announcement; although it is usually not confidential, the document's integrity is critical.

Beyond CIA

Going one more step beyond the CIA security triad, we can think of:

- **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.

These two requirements are closely related. The need to tell authentic files or orders from fake ones is indispensable. Moreover, ensuring that the other party cannot deny being the source is vital for many systems to be usable.

In online shopping, depending on your business, you might tolerate attempting to deliver a t-shirt with cash-on-delivery and learn later that the recipient never placed such an order. However, no company can tolerate shipping 1000 cars to discover that the order is fake. In the example of a shopping order, you want to confirm that the said customer indeed placed this order; that's authenticity. Moreover, you want to ensure they cannot deny placing this order; that's nonrepudiation.

As a company, if you receive a shipment order of 1000 cars, you need to ensure the authenticity of this order; moreover, the source should not be able to deny placing such an order. Without authenticity and nonrepudiation, the business cannot be conducted.

Parkerian Hexad

In 1998, Donn Parker proposed the Parkerian Hexad, a set of six security elements. They are:

1. Availability
2. Utility
3. Integrity
4. Authenticity
5. Confidentiality
6. Possession

We have already covered four of the above six elements. Let's discuss the remaining two elements:

- **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.

Answer the questions below

1. Two companies are negotiating a certain agreement; however, they want to keep the details of the agreement secret. Which security pillar are they emphasizing? **Confidentiality**
2. As the troops got deployed, the leader stressed that they should not communicate their location to anyone while the mission was ongoing. Which security function did the leader want to have? **Confidentiality**
3. You went to cash out a check, and the bank teller made you wait for five minutes as they confirmed the signature of the check's issuer. Which security function is the bank teller checking? **Integrity**
4. One hotel is stressing that the Internet over its WiFi network must be accessible 24 hours a day, seven days a week. Which security pillar is the hotel requiring? **Availability**
5. At a police checkpoint, the police officer suspected that the vehicle registration papers were fake. Which security function does the officer think is lacking? **Integrity**

Click on "View Site" and answer the five questions. What is the flag that you obtained at the end?
THM{CIA_TRIAD}

DAD

The security of a system is attacked through one of several means. It can be via the disclosure of secret data, alteration of data, or destruction of data.

- **Disclosure** is the opposite of confidentiality. In other words, disclosure of confidential data would be an attack on confidentiality.
- **Alteration** is the opposite of Integrity. For example, the integrity of a check is indispensable.
- **Destruction/Denial** is the opposite of Availability.

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

Consider the previous example of patient records and related systems:

- 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.

Protecting against disclosure, alteration, and destruction/denial is of utter significance. This protection is equivalent to working to maintain confidentiality, integrity and availability.

Protecting confidentiality and integrity to an extreme can restrict availability, and increasing availability to an extreme can result in losing confidentiality and integrity. Good security principles implementation requires a balance between the three.

Answer the questions below

1. The attacker managed to gain access to customer records and dumped them online. What is this attack? **Disclosure**
2. A group of attackers were able to locate both the main and the backup power supply systems and switch them off. As a result, the whole network was shut down. What is this attack? **Destruction/Denial**

Fundamental Concepts Of Security Models

We have learned that the security triad is represented by Confidentiality, Integrity, and Availability (CIA). One might ask, how can we create a system that ensures one or more security functions? The answer would be in using security models. In this task, we will introduce three foundational security models:

- Bell-LaPadula Model
- The Biba Integrity Model
- The Clark-Wilson Model

Bell-LaPadula Model

The Bell-LaPadula Model aims to achieve **confidentiality** by specifying three rules:

- **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).

There are certain limitations to the Bell-LaPadula model. For example, it was not designed to handle file-sharing.

Biba Model

The Biba Model aims to achieve **integrity** by specifying two main rules:

- **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.

These two properties can be summarized as “read up, write down.” This rule is in contrast with the Bell-LaPadula Model, and this should not be surprising as one is concerned with confidentiality while the other is with integrity.

Biba Model suffers from various limitations. One example is that it does not handle internal threats (insider threat).

Clark-Wilson Model

The Clark-Wilson Model also aims to achieve integrity by using the following concepts:

- **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.

We covered only three security models. The reader can explore many additional security models. Examples include:

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

Answer the questions below

1. Which model dictates “no read down”? **Biba**
2. Which model forces “no write up”? **Biba**
3. Which model states “no read up”? **Bell-LaPadula**
4. Which model teaches “no write down”? **Bell-LaPadula**

Click on "View Site" and answer the four questions. What is the flag that you obtained at the end? THM{SECURITY_MODELS}

Defense In Depth

Defence-in-Depth refers to creating a security system of multiple levels; hence it is also called Multi-Level Security.

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

The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) have created the ISO/IEC 19249. In this task, we will brush briefly upon ISO/IEC 19249:2017 *Information technology - Security techniques - Catalogue of architectural and design principles for secure products, systems and applications*. The purpose is to have a better idea of what international organizations would teach regarding security principles.

ISO/IEC 19249 lists five *architectural* principles:

1. **Domain Separation**: Every set of related components is grouped as a single entity; components can be applications, data, or other resources. Each entity will have its own domain and be assigned a common set of security attributes. For example, consider the x86 processor privilege levels: the operating system kernel can run in ring 0 (the most privileged level). In contrast, user-mode applications can run in ring 3 (the least privileged level). Domain separation is included in the Goguen-Meseguer Model.
2. **Layering**: When a system is structured into many abstract levels or layers, it becomes possible to impose security policies at different levels; moreover, it would be feasible to validate the operation. Let's consider the OSI (Open Systems Interconnection) model with its seven layers in networking. Each layer in the OSI model provides specific services to the layer above it. This layering makes it possible to impose security policies and easily validate that the system is working as intended. Another example from the programming world is disk operations; a programmer usually uses the disk read and write functions provided by the chosen high-level programming language. The programming language hides the low-level system calls and presents them as more user-friendly methods. Layering relates to Defence in Depth.
3. **Encapsulation**: In object-oriented programming (OOP), we hide low-level implementations and prevent direct manipulation of the data in an object by providing specific methods for that purpose. For example, if you have a clock object, you would provide a method `increment()` instead of giving the user

direct access to the `seconds` variable. The aim is to prevent invalid values for your variables. Similarly, in larger systems, you would use (or even design) a proper Application Programming Interface (API) that your application would use to access the database.

4. **Redundancy**: This principle ensures availability and integrity. There are many examples related to redundancy. Consider the case of a hardware server with two built-in power supplies: if one power supply fails, the system continues to function. Consider a RAID 5 configuration with three drives: if one drive fails, data remains available using the remaining two drives. Moreover, if data is improperly changed on one of the disks, it would be detected via the parity, ensuring the data's integrity.
5. **Virtualization**: With the advent of cloud services, virtualization has become more common and popular. The concept of virtualization is sharing a single set of hardware among multiple operating systems. Virtualization provides sandboxing capabilities that improve security boundaries, secure detonation, and observance of malicious programs.

ISO/IEC 19249 teaches five *design* principles:

1. **Least Privilege**: You can also phrase it informally as "need-to basis" or "need-to-know basis" as you answer the question, "who can access what?" The principle of least privilege teaches that you should provide the least amount of permissions for someone to carry out their task and nothing more. For example, if a user needs to be able to view a document, you should give them read rights without write rights.
2. **Attack Surface Minimisation**: Every system has vulnerabilities that an attacker might use to compromise a system. Some vulnerabilities are known, while others are yet to be discovered. These vulnerabilities represent risks that we should aim to minimize. For example, in one of the steps to harden a Linux system, we would disable any service we don't need.
3. **Centralized Parameter Validation**: Many threats are due to the system receiving input, especially from users. Invalid inputs can be used to exploit vulnerabilities in the system, such as denial of service and remote code execution. Therefore, parameter validation is a necessary step to ensure the correct system state. Considering the number of parameters a system handles, the validation of the parameters should be centralized within one library or system.
4. **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.

5. **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 safely; 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.

In the following questions, refer to the ISO/IEC 19249 five design principles above. Answer with a number between 1 and 5, depending on the number of the design principle.

Answer the questions below

1. Which principle are you applying when you turn off an insecure server that is not critical to the business? **2**
2. Your company hired a new sales representative. Which principle are they applying when they tell you to give them access only to the company products and prices? **1**
3. While reading the code of an ATM, you noticed a huge chunk of code to handle unexpected situations such as network disconnection and power failure. Which principle are they applying? **5**

Zero Trust against Trust But Verify

Trust is a very complex topic; in reality, we cannot function without trust. If one were to think that the laptop vendor has installed spyware on the laptop, they would most likely end up rebuilding the system. If one were to mistrust the hardware vendor, they would stop using it completely. If we think of trust on a business level, things only become more sophisticated; however, we need some guiding security principles. Two security principles that are of interest to us regarding trust:

- Trust but Verify
- Zero Trust

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.

Microsegmentation is one of the implementations used for Zero Trust. It refers to the design where a network segment can be as small as a single host. Moreover, communication between segments requires authentication, access control list checks, and other security requirements.

There is a limit to how much we can apply zero trust without negatively impacting a business; however, this does not mean that we should not apply it as long as it is feasible.

Threat against Risk

There are three terms that we need to take note of to avoid any confusion.

- **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.

Away from information systems, a showroom with doors and windows made of standard glass suffers a weakness, or *vulnerability*, due to the nature of glass. Consequently, there is a *threat* that the glass doors and windows can be broken. The showroom owners should contemplate the *risk*, i.e. the likelihood that a glass door or window gets broken and the resulting impact on the business.

Consider another example directly related to information systems. You work for a hospital that uses a particular database system to store all the medical records. One day, you are following the latest security news, and you learn that the used database system is not only vulnerable but also a proof-of-concept working exploit code has been released; the released exploit code indicates that the threat is real. With this knowledge, you must consider the resulting risk and decide the next steps.

Conclusion

By now, you should be very familiar with CIA and DAD and other terms such as authenticity, repudiation, vulnerability, threat, and risk. We visited three security models and the ISO/IEC 19249. We have covered different security principles such as defence in depth, trust but verify, and zero trust.

Finally, the Shared Responsibility Model is worth mentioning, especially with the increased reliance on cloud services. Various aspects are required to ensure proper security. They include hardware, network infrastructure, operating systems, applications, etc. However, customers using cloud services have different access levels depending on the cloud services they use. For example, an Infrastructure as a Service (IaaS) user has complete control (and responsibility) over the operating system.

On the other hand, a Software as a Service (SaaS) user has no direct access to the underlying operating system. Consequently, achieving security in a cloud environment necessitates both the cloud service provider and the user to do their parts. The Shared Responsibility Model is a cloud security framework to ensure that each party is aware of its responsibility.