|  |  |
| --- | --- |
| **Concept** | **Definition** |
| **Authentication** | process of verifying the identity of a user, device, or system attempting to access a resource |
| **Authorization** | process of determining what level of access a user or system has to a resource or service once they have been authenticated. |
| **Accounting** | process of tracking and logging user and system activity to ensure that all actions are recorded and auditable. |
| **Least Privilege** | limiting user and system permissions to the minimum level necessary to perform their job functions. |
| **Need to Know** | limiting access to sensitive information to only those individuals who require it to perform their job functions. |
| **Segregation of Duties** | principle of dividing job functions and responsibilities among multiple individuals to prevent any single person from having too much control over a process or system. |
| **Threat** | it refers to any potential danger or harm that could exploit a vulnerability and cause damage or disruption to a system, application, network, or data. |
| **Vulnerability** | is a weakness in a system, application, network, or data that can be exploited by a threat |
| **Risk** | Risk: Risk is the likelihood or probability of a threat exploiting a vulnerability and causing harm or damage. **Risk = Vulnerability x Threat** |
| **Redefine the Perimeter** | focusing on securing data and applications rather than just the network perimeter. This means implementing security controls that can protect data and applications |
| **Implement Least Privilege:** | Like Least Privilege |
| **Never Trust**  **Always Verify** | assuming that all users, devices, and systems are potentially compromised, and verifying their identity and access permissions before granting access to resources. |
| **Assume Breach** | assuming that a security breach has already occurred or will occur, and focusing on detecting and responding to security incidents rather than just preventing them |
| **Defense in Depth** | implementing multiple layers of security controls to protect systems and data. |
| **Session** | a series of related browser requests that come from the same client during a certain time period. |
| **Session Hijacking** | It refers to the exploitation of a valid session assigned to a user. |
| **Cross-site scripting (XSS)** | type of security vulnerability that can be found in some web applications, XSS attacks enable attackers to inject client-side scripts into web pages viewed by other users. |

**Secure SDLC - Security Architecture**

**1 – Tier Architecture**

also known as a monolithic architecture, the entire application is contained in a single executable file or codebase.

**2 – Tier Architecture**

also known as a client-server architecture, the application is split into two components: a client component and a server component. The client component interacts with the user and sends requests to the server component, which processes those requests and sends responses back to the client.

**3 – Tier Architecture**

the application is split into three components: a presentation layer, a business logic layer, and a data storage layer. The presentation layer interacts with the user, the business logic layer processes requests and performs application-specific operations, and the data storage layer stores and retrieves data.

**Application Assessments**

1. **Static Application Security Testing - SAST:**

Used for static analysis will take your code as input and analyze each line for any insecure functions or coding practices.

* **Weakness**
* Cannot identify subjective or business logic related issues
* Extremely slow in adopting new versions of programming languages
* Requires more effort than dynamic analysis when dealing with tool results
* **Strength**
* Quick in identifying obvious coding flaws
* Can be run in parallel with development to reduce overhead at the end of the development life cycle

1. **Dynamic Application Security Testing – DAST:**

The software or individual tester sits between the server and the browser while modifying requests to identify flaws in how the server reacts to them.

* **Weakness**
* Depends heavily on the qualifications of the tester
* **Strength**
* Covers all of Application vulnerability testing.
* Can be leveraged into checking for more sophisticated attacks by doing additional manual checks.