1. **Broken object level Authorization.**

It is a security vulnerability that occurs when an application or application programming interface (API) provides access to data objects based on the user's role, but fails to verify if the user is authorized to access those specific data objects.

1. **Broken User Authentication.**

Authentication is “broken” when attackers are able to compromise passwords, keys or session tokens, user account information, and other details to assume user identities. This occurs due to poor design and implementation of identity and access controls.

1. **Excessive Data Exposure.**

It is when APIs reveal more fields, data, and information than the client requires through the API response. Excessive data exposure flaws expose all object properties to API calls rather than what the user needs to act on without considering the object's sensitivity level.

1. **Lack of Resource and Rate limit.**

Lack of Resources & Rate Limiting occurs when developers/organizations don't impose restrictions on the size and number of resources requested by the user/ client in an API call or on the number of API calls a client/user can make within a given time frame.

1. **Broken Function Level Authorization.**

Itrefers to the user hierarchical permissions system being incomplete or broken. APIs that involve complex permissions and user roles that can span the hierarchy in different ways are more prone to having broken function-level authorizations.

1. **Mass Assignment.**

Mass assignment vulnerabilities occur when a user is able to initialize or overwrite server-side variables for which are not intended by the application.

1. **Security Misconfigurations.**

A security misconfiguration occurs when system or application configuration settings are missing or are erroneously implemented, allowing unauthorized access.

1. **Injections.**

Injection flaws, such as SQL, NoSQL, Command Injection, etc., occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's malicious data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

1. **Improper Asset Management.**

Improper asset management is a vulnerability caused by lack of a technical overview of deployed API assets where these assets may be vulnerable to exploits due to stagnation and lack of oversight and ownership.

1. **Insufficient Logging and Monitoring.**

insufficient logging and monitoring, it is almost impossible to track suspicious activities and respond to them in a timely fashion. Without visibility over on-going malicious activities, attackers have plenty of time to fully compromise systems.