1. Describe the differences between authentication and authorization, and explain why they are both important in web application security.

* **Authentication** is the technique of verifying the identity of a consumer or entity attempting to access an internet software. It usually includes using credentials which include usernames, passwords, or other authentication factors (fingerprint, face recognition) to validate the claimed identification.
* **Authorization** is the system of granting or denying get right of entry to to specific resources or moves inside a web utility primarily based at the authenticated consumer's privileges or permissions.

1. What are some common techniques used for authentication and authorization in Node.js applications? Provide examples of each.

**Authentication Techniques**

* + **Username/Password Authentication**
  + **Social Media Authentication**

**Authorization Techniques**

* + **Role-Based Access Control**
  + **JSON Web Tokens (JWT)**

1. Explain what JSON Web Tokens (JWTs) are and how they are used in Node.js applications. How do they improve security compared to other authentication methods?

* JSON internet Tokens (JWTs) are a kind of token-based total authentication mechanism used in internet programs, along with Node.Js packages. JWTs are a compact, URL-secure approach of representing claims among parties, generally used for authentication and authorization.

1. What is OAuth 2.0, and how is it used for authentication and authorization in Node.js applications? Describe the different OAuth 2.0 flows and how they are used in practice.

* OAuth 2.0 is an authorization framework that lets in customers to furnish get right of entry to their resources to 0.33-celebration programs, without sharing their login credentials.

1. What is role-based access control (RBAC), and how is it used for authorization in Node.js applications? Provide an example of how you might use RBAC to control access to a specific resource or functionality.

* Role-based get entry to manage (RBAC) is a common authorization mechanism used in Node.Js applications to control and put in force get right of entry to manipulate based totally on the jobs or permissions assigned to users.

1. Describe the different techniques used for secure password storage in Node.js applications, and explain why secure password storage is important.

* Hashing
* Salting
* Key stretching
* secure password storage is essential in Node.Js applications because it facilitates guard person accounts from unauthorized get admission to

1. What is two-factor authentication (2FA), and how can it be implemented in Node.js applications? Explain the benefits and potential drawbacks of 2FA.
   * Something the user knows
   * Something the user has
   * Something the user is
2. How can session management be used to improve authentication and authorization in Node.js applications? Describe some common techniques used for session management, and explain why they are important for security.

* Session management is a critical factor of authentication and authorization in net applications.
* Server-side sessions
* JSON Web Tokens (JWTs)
* Client-side sessions

1. Describe a specific security vulnerability that could be exploited in a Node.js application that does not properly handle authentication or authorization. How might an attacker exploit this vulnerability, and what steps can be taken to prevent it?

* A specific security vulnerability that might be exploited in a Node.Js application that doesn't properly cope with authentication or authorization is the "Insecure Direct item Reference" (IDOR) vulnerability.
* Proper authorization
* Use of indirect references
* Input validation

1. Explain why it is important to regularly review and update the authentication and authorization mechanisms used in a Node.js application. What are some common issues that can arise if these mechanisms are not kept up-to-date?

* Often reviewing and updating authentication and authorization mechanisms in a Node.Js application is important for maintaining protection and protective against capacity vulnerabilities.
* It allows to patch security vulnerabilities,
* observe safety standards,
* adapt to converting enterprise necessities,
* and avoid troubles including records breaches, compliance violations, unauthorized get admission to, and person revel in issues.
* Failure to keep those mechanisms up-to-date can bring about security dangers, facts breaches, compliance violations, unauthorized get right of entry to, and poor user experience.