|  |  |  |
| --- | --- | --- |
| **S.No.** | **Questions** | **Expected answers** |
| 1 | Do you have knowledge on implementing secure communication protocols, such as SSL/TLS, in frontend applications? | Using HTTPS: Always make API requests over https:// to encrypt data in transit. This is crucial for protecting user data. Secure Headers: Use security headers like Content-Security-Policy, X-Content-Type-Options, and Strict-Transport-Security to enhance security. |
| 2 | What is encryption and decryption and how do you implement them? When do we need them? | Encryption is the process of converting data into a secure format. In the frontend. Decryption: Reverse the process to retrieve the original data. Libraries: Use libraries like Crypto JS or the Web Crypto API for encryption. Use encryption when handling sensitive data (e.g., user credentials) before sending it to the server. |
| 3 | What is Encoding and how do you implement them? When do we need it? | Encoding transforms data into a specific format for safe transmission. In the frontend. URL Encoding: Use encode URI Component to encode special characters in URLs. HTML Encoding: Use libraries like DOM Purify to sanitize HTML content before rendering it.  Use encoding when sending data in URLs or rendering user-generated content to prevent misinterpretation by browsers. |
| 4 | What is a cookie? When and how do we use it? | Cookies are small pieces of data stored in the browser, typically used for: Session Management: Storing user sessions to maintain logged-in states. Preferences: Saving user settings like themes. |
| 5 | What is CSRF? How do you prevent it? | CSRF is an attack that tricks users into executing unwanted actions. To prevent it: Anti-CSRF Tokens: Include a unique token in each form submission that must be validated on the server. Same Site Cookies: Set the Same Site attribute to Strict or Lax to limit how cookies are sent with cross-site requests. |
| 6 | What is SQL Injection? How do you prevent it? | Input Validation: Validate and sanitize all user inputs before sending them to the backend. |
| 7 | What is DDoS attack? How do you prevent it? | A DDoS attack overwhelms a service with traffic. To prevent it: Implementing Captcha Enabling Rate Limiters Using a CDN: Leverage a Content Delivery Network to absorb traffic spikes. Optimizing Frontend Assets: Minimize and compress files to reduce server load. |
| 8 | What is Cross Site Scripting(XSS) ? How do you prevent it? | XSS allows attackers to inject scripts into web pages. To prevent it: Output Encoding: Always encode user input when rendering on the page. Content Security Policy (CSP): Implement a CSP to restrict sources of scripts and resources. |
| 9 | How do you handle input validations and prevent common security vulnerabilities, such as SQL injection or cross-site scripting (XSS)? | Client-Side Validation: Use HTML5 validation attributes (like required, pattern) and JavaScript validation for immediate feedback. Sanitization: Use libraries to sanitize input before rendering (e.g., DOM Purify for HTML). Server-Side Validation: Always validate and sanitize data on the server side to provide an additional layer of security. |
| 10 | Can you describe how you would conduct a security review or vulnerability assessment of a frontend application? What tools or techniques would you use? | Static Code Analysis: Use tools like ESL int with security plugins to identify issues. Dynamic Testing: Conduct penetration tests using tools like OWASP ZAP. Dependency Scanning: Use tools like npm audit or Snyk to find vulnerabilities in dependencies. |
| 11 | Have you ever encountered a security incident or breach in a the frontend application? How did you handle it, and what measures did you take to prevent similar incidents in the future? | Immediate Response: Assess the situation to understand the breach. Patching: Fix the vulnerabilities exploited during the incident. Documentation: Document the incident and the steps taken to prevent recurrence, including updates to coding practices or libraries used. |
| 12 | What are some common techniques or best practices you follow to secure frontend applications? | Input Validation Use Security Headers like X-Content-Type-Options, X-Frame-Options, and CSP Strong Authentication Access Control Secure protocols (HTTPS)  Error Handling Regular Tech stack Updates |