**Q1: What is a JSON Web Token (JWT)?**  
**A:** A JSON Web Token (JWT) is a compact, URL-safe token that represents a set of claims between two parties. It is commonly used for authentication and authorization in web applications. The claims within a JWT are encoded as a JSON object and can be signed or encrypted to ensure their integrity and confidentiality.

**Q2: What are the main components of a JWT?**  
**A:** A JWT consists of three main parts:

**Header:** Contains metadata about the token, including the type of token (JWT) and the signing algorithm used (e.g., HMAC SHA256 or RSA).

**Payload:** Contains the claims, which are statements about an entity (typically, the user) and additional data. Claims can be public, private, or registered.

**Signature:** Used to verify the authenticity of the token. It is created by encoding the header and payload, and then signing it using the specified algorithm and a secret or private key.

**Q3: How is a JWT typically used in authentication?**  
**A:** In authentication, a JWT is typically issued by an authentication server after a user successfully logs in. The JWT is then sent back to the client (usually a web browser or mobile app), where it is stored (commonly in local storage or a cookie). For subsequent requests, the client includes the JWT in the Authorization header, usually as a Bearer token. The server then verifies the token to authenticate the user and grant access to protected resources.

**Q4: What is the difference between JWT and traditional session-based authentication?**  
**A:** In traditional session-based authentication, the server stores session data in memory or a database and identifies the user with a session ID stored in a cookie. This requires the server to maintain state. In contrast, JWT is stateless; the token itself contains all the necessary information to identify the user, and no server-side session data is required. This makes JWT more scalable, especially for distributed systems or microservices.

**Q5: What are the common use cases for JWT?**  
**A:** Common use cases for JWT include:

**Authentication:** JWT is widely used in web and mobile applications to authenticate users and manage their sessions.

**Authorization:** After a user is authenticated, JWT can be used to grant access to specific resources by including user roles or permissions in the token.

**Information Exchange:** JWT can securely transmit information between parties, ensuring that the data is not tampered with.

**Q6: How is the security of a JWT ensured?**  
**A:** The security of a JWT is ensured through its signature. The signature is generated by encoding the header and payload and then signing them using a secret key or private key. The recipient of the JWT can verify the signature using the corresponding secret or public key, ensuring that the token has not been altered. Additionally, using HTTPS is recommended to protect the JWT during transmission.

**Q7: What are some common vulnerabilities of JWT, and how can they be mitigated?**  
**A:** Some common vulnerabilities of JWT include:

**Token theft:** If a JWT is stolen (e.g., from local storage or a compromised network), an attacker can use it to gain unauthorized access. This can be mitigated by implementing token expiration, using secure storage (e.g., cookies with the HttpOnly flag), and employing refresh tokens.

**Weak signing algorithms:** Some JWTs use weak signing algorithms, which can be exploited. Always use strong, industry-standard algorithms like RS256 or HS256.

**Lack of encryption:** If the payload contains sensitive information, it should be encrypted to prevent exposure. This can be achieved using JWE (JSON Web Encryption).

**Q8: What is the difference between symmetric and asymmetric signing in JWT?**  
**A:** In symmetric signing (e.g., HS256), the same secret key is used to sign and verify the JWT. Both the issuer and the recipient must share the secret key. In asymmetric signing (e.g., RS256), a private key is used to sign the JWT, and a corresponding public key is used to verify it. This allows the issuer to keep the private key secure while distributing the public key to any number of recipients.

**Q9: How can you invalidate a JWT?**  
**A:** Since JWTs are stateless, invalidating them can be challenging. Some common methods include:

**Short-lived tokens:** Use short expiration times to limit the window of validity.

**Token revocation list:** Maintain a list of revoked tokens on the server and check this list before accepting any JWT.

**Rotating tokens:** Issue new tokens frequently (e.g., using refresh tokens), making older tokens obsolete.

**Q10: What are the advantages of using JWT over other token-based authentication methods?**  
**A:** The advantages of using JWT include:

**Statelessness:** JWT does not require server-side storage, making it ideal for distributed systems and microservices.

**Scalability:** Because no session state is stored on the server, JWT can scale easily across multiple servers and services.

**Flexibility:** JWT can contain any kind of claim, allowing it to be used for various purposes such as authentication, authorization, and data exchange.

**Compactness:** JWTs are compact and can be efficiently transmitted over the network, making them suitable for mobile and IoT applications.