1. **What is the difference between authentication and authorization?**

Authentication is the process of verifying who a user is, while authorization is the process of verifying what they have access to.

Here’s a quick overview of the differences between authentication and authorization:

|  |  |
| --- | --- |
| **Authentication** | **Authorization** |
| Determines whether users are who they claim to be | Determines what users can and cannot access |
| Challenges the user to validate credentials (for example, through passwords, answers to security questions, or facial recognition) | Verifies whether access is allowed through policies and rules |
| Usually done before authorization | Usually done after successful authentication |
| Generally, transmits info through an ID Token | Generally, transmits info through an Access Token |
| Generally governed by the OpenID Connect (OIDC) protocol | Generally governed by the OAuth 2.0 framework |

1. **What authorization approaches can you list? What is role-based access control?**

There are several approaches to authorization, including:

1. **Discretionary Access Control (DAC):** A type of access control where the owner of a resource decides who has access to it.
2. **Mandatory Access Control (MAC):** A type of access control where access to a resource is based on a set of rules defined by a system administrator or security policy.
3. **Role-Based Access Control (RBAC):** A type of access control where access to resources is based on the roles that users have in an organization. Users are assigned roles that define their permissions, and access is granted based on those roles.
4. **Attribute-Based Access Control (ABAC):** A type of access control where access to resources is based on attributes associated with users, resources, and the environment. Access is granted or denied based on a set of rules that consider the attributes.

Role-Based Access Control (RBAC) is a popular approach to authorization that is widely used in many organizations. In RBAC, access to resources is granted based on the roles that users have in the organization. Roles are defined based on the job functions or responsibilities of users, and access is granted or denied based on those roles. RBAC provides a centralized way to manage access control, making it easier to add or remove users and update their permissions. RBAC is also considered more secure than other approaches because it limits access to only what is necessary for a user's role, reducing the risk of accidental or intentional data breaches.

1. **What exactly is Identity Management (Identity and Access Management)?**

Identity Management (IDM) or Identity and Access Management (IAM) is a set of processes, technologies, and policies used to manage digital identities and their access to systems, applications, and data. The goal of IDM is to ensure that the right individuals have access to the right resources at the right timem while preventing unauthorized access to sensitive information.

IDM involves several key components, including:

1. Identity Provisioning. The process of creating, managing, and deactivating user accounts and credentials, such as usernames and passwords.
2. Authentication. The process of verifying the identity of a user, device, or system attempting to access a resource.
3. Authorization. The process of granting or denying access to resources based on the user’s identity, role, and privileges.
4. Access Management. The process of managing user access to resources, including monitoring, auditing, and revoking access as needed.
5. Identity Governance: The process of ensuring that user access is compliant with internal policies, regulations, and industry standards.

IDM systems typically include centralized directories, such as Active Directory or LDAP, which store user identities and access privileges. IDM solutions also include tools for identity provisioning, authentication, and authorization, as well as reporting and auditing capabilities to ensure compliance with regulations and standards.

1. **What authentication/authorization protocols do you know? What is the difference between OAuth & OpenID?**

There are several authentication and authorization protocols, including:

1. OAuth: An authorization protocol used to grant third-party applications access to resources on behalf of a user, without requiring the user to disclose their credentials to the third party. OAuth is commonly used to provide access to social media platforms, cloud storage services, and other web-based applications.
2. OpenID: An authentication protocol used to verify the identity of a user, allowing them to log in to different websites and applications using a single set of credentials. OpenID allows users to use their existing accounts from providers such as Google, Facebook, or LinkedIn to log in to other websites.
3. Security Assertion Markup Language (SAML): A standard protocol used for exchanging authentication and authorization data between parties, typically a user, an identity provider, and a service provider. SAML is commonly used for single sign-on (SSO) authentication in enterprise environments.
4. Kerberos: A network authentication protocol used to verify the identity of users or services over a network, providing mutual authentication and encryption of network communications.

The main difference between OAuth and Open ID is that OAuth is an authorization protocol, while OpenID is an authentication protocol. OAuth allows third-party applications to access resources on behalf of a user, while OpenID allows a user to use a single set of credentials to log in to different websites and applications. However, both protocols are often used together in a complementary way to provide authentication and authorization solution.

1. **What is Authentication/Authorization Token. What is JWT token? What other approaches except authentication/authorization, can we use with security token?**

An authentication or authorization token is a piece of data that is used to authenticate a user or authorize access to a resource.

Tokens are typically created by an authentication or authorization server and are passed between the client and server to establish and maintain a user’s session.

JWT, or JSON Web Token, is a type of token that is commonly used for authentication and authorization in web applications.

JWT tokens are encoded using JSON and are-self-contained, meaning that they contain all necessary information for authentication and authorization in a single token.

In addition to using authentication and authorization tokens, there are other security token approaches that can be used to secure applications and systems, including:

1. One-Time Password (OTP) Tokens: OTP tokens generate a unique, one-time code that is used for authentication. The code is typically generated by a hardware or software token and is valid for a short period of time.
2. X.509 Certificates: X.509 certificates are digital certificates used to verify the identity of a user or system. They are commonly used for authentication in secure web applications and other systems.
3. Kerberos Tickets: Kerberos is a network authentication protocol that uses tickets to verify the identity of users and services. Kerberos tickets are used to provide mutual authentication and encryption of network communications.
4. Biometric tokens: Biometric tokens use biometric data, such as fingerprints, facial recognition, or iris scans, to authenticate users. They are commonly used in high-security applications and systems.

Each of these security token approaches provides a different level of security and usability, and the choice of which to use depends on the requirements of the application or system.

1. **What is Single Sign-On (SSO)? Name the steps to implement SSO. What are the benefits of SSO?**

Single Sign-ON (SSO) is an authentication process that allows users to log in once and gain access to multiple applications or systems without needing to enter their credentials again. With SSO, users can authenticate once and access multiple resources, simplifying the login process and reducing the need to remember multiple usernames and passwords.

The steps to implement SSO are follows:

1. Planning: Identify the resources that need to be included in the SSO solution and define the authentication and authorization policies.
2. Configuration: configure the SSO server, including and connecting to the user directory.
3. Integration: Integrate the SSO solution with the applications or systems that need to be accessed using SSO. This typically involves configuring the applications or systems to recognize and trust the SSO server.
4. Testing: Test the SSO solution to ensure that it is working as expected and that users can access the resources they need.

The benefits of SSO include:

1. Improved User Experience: SSO simplifies the login process, reducing the need to remember multiple usernames and passwords and improving the user experience.
2. Increased Security: SSO eliminates the need for users to store passwords for multiple applications, reducing the risk of password-based attacks and improving overall security.
3. Simplified Access Management: SSO simplifies the management of user access to resources, making it easier to revoke access when necessary and ensuring that users have access to the resources they need.
4. Cost Savings: SSO can help reduce the costs associated with managing user access to resources, including password resets and help desk support.
5. **What is the difference between Two-Factor Authentication and Multi-Factor Authentication?**

Two-Factor Authentication (2FA) and Multi-Factor Authentication (MFA) are both security mechanisms used to authenticate users and protect against unauthorized access.

2FA is a security process that requires users to provide two different types of authentication factors in order to access a resource. Typically, these factors are something the user has, such as a mobile phone. 2FA provides an extra layer of security beyond just a password, making it more difficult for attackers to gain access to a user’s account.

MFA, on the other hand, is a security process that requires users to provide more than two different types of authentication factors to access a resource. In addition to something the user knows and something the user has, MFA may also require something the user is, such as a fingerprint or other biometric data. MFA provides an even higher level of security than 2FA, making it more difficult for attackers to gain access to a user’s account.

In summary, the main difference between 2FA and MFA is the number of authentication factors required. 2FA requires two factors, while MFA requires more than two. While both mechanisms provide extra layers of security beyond just a password, MFA provides an even higher level of security than 2FA.

1. **Which of the OAuth flows can be used for user (customer) and which for client (server) authentication?**

OAuth is an authorization protocol that allows a client application to access resources on behalf of a user. There are several OAuth flows defined, each with a different set of requirements and use cases.

The flows that can be used for user (customer) authentication are:

1. Authorization Code Flow: This flow is typically used for server-side web applications where the client application is trusted and can maintain the client secret. The user is redirected to the authorization server to authenticate and grant access to the client application.
2. Implicit Flow. This flow is typically used for client-side web applications or mobile applications where the client application cannot maintain the client secret. The user is redirected to the authorization server to authenticate and grant access to the client application.
3. Resource Owner Password Credentials Flow. This flow is typically used for trusted applications where the client application can securely store and use the user’s credentials. The user provides their credentials directly to the client application, which the uses them to obtain an access token from the authorization server.

The flow that can be used for client (server) authentication is:

1. Client Credentials Flow: This flow is typically used for server-to-server communication where the client application needs to access resources on its own behalf, rather than on behalf of a user. The client application provides its own credentials to authorization server to obtain an access token.

The Authorization Code Flow, Implicit Flow, and Resource Owner Password Credentials Flow can be used for user (customer) authentication, while the Client Credentials Flow can be used for client (server) authentication.