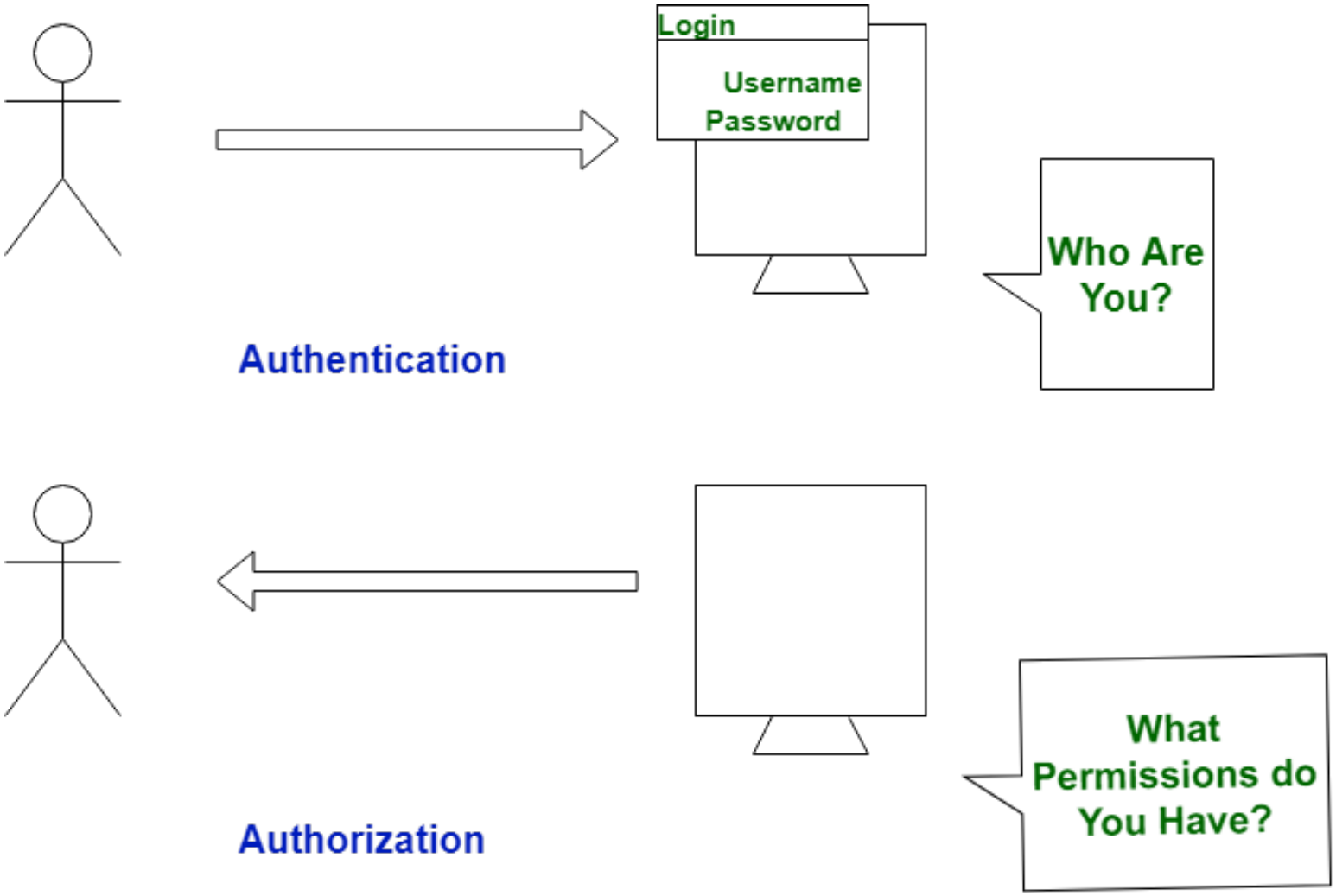
Authentication

- Authentication (AuthN) is a process that verifies that someone or something is who they say they are. Technology systems typically use some form of authentication to secure access to an application or its data. Verifies credentials
- It is used by both server and client. The server uses authentication when someone wants to access the information, and the server needs to know who is accessing the information mostly using username and password. The client uses it when he wants to know that it is the same server that it claims to be.
- Authentication works through passwords, one-time pins(Grant access for only one session or transaction), biometric information(A user presents a fingerprint or eye scan to gain access to the system), and other information provided or entered by the user.
- Two-factor Authentication: it needs two-step verification to authenticate a user. It does not require only a username and password but also needs the unique information that only the particular user knows, such as first school name, a favorite destination. Apart from this, it can also verify the user by sending the OTP or a unique link on the user's registered number or email address.
- Single Sign-on: a way to enable access to multiple applications with a single set of credentials.

Authorization

- Authorization: Authorization is the security process that determines a user or service's level of access. It defines that what data and information one user can access. It is also said as AuthZ. Authorization always takes place after authentication. Grants or denies permissions.
- Role-based access control: RBAC or Role-based access control technique is given to users as per their role or profile in the organization. It can be implemented for system-system or user-to-system.
- Oauth: OAuth is an authorization protocol, which enables the API to authenticate and access the requested resources.
- Authorization permissions cannot be changed by the user. The permissions are given to a user by the owner/manager of the system, and he can only change it.
- Access Control Lists (ACLs) determine which users or services can access a particular digital environment. They accomplish this access control by enforcing allow or deny rules based on the user's authorization level. Eg- users and admin role.
- we can say Authentication verifies the user's identity, and Authorization verifies the user's access and permissions. If the user can't prove their identity, they cannot access the system. And if you are authenticated by proving the correct identity, but you are not authorized to perform a specific function, you won't be able to access that. However, both security methods are often used together.



OAuth 2.0

- “Open Authorization”, is a standard designed to allow a website or application to access resources hosted by other web apps on behalf of a user.
- It replaced OAuth 1.0 in 2012 and is now the de facto industry standard for online authorization. OAuth 2.0 provides consented access and restricts actions of what the client app can perform on resources on behalf of the user, without ever sharing the user's credentials.
- OAuth 2.0 is an authorization protocol and NOT an authentication protocol. As such, it is designed primarily as a means of granting access to a set of resources, for example, remote APIs or user’s data.
- OAuth 2.0 uses Access Tokens, a piece of data that represents the authorization to access resources on behalf of the end-user. OAuth 2.0 doesn’t define a specific format for Access Tokens. However, in some contexts, the JSON Web Token (JWT) format is often used. This enables token issuers to include data in the token itself. Also, for security reasons, Access Tokens may have an expiration date.

OAuth2.0 Roles

- **Resource Owner**: The user or system that owns the protected resources and can grant access to them.
- **Client**: The client is the system that requires access to the protected resources. To access resources, the Client must hold the appropriate Access Token.
- **Authorization Server**: This server receives requests from the Client for Access Tokens and issues them upon successful authentication and consent by the Resource Owner. The authorization server exposes two endpoints: the Authorization endpoint, which handles the interactive authentication and consent of the user, and the Token endpoint, which is involved in a machine to machine interaction.
- **Resource Server**: A server that protects the user's resources and receives access requests from the Client. It accepts and validates an Access Token from the Client and returns the appropriate resources to it.

Grant Types in OAuth 2.0

- **Authorization Code grant**: The Authorization server returns a single-use Authorization Code to the Client, which is then exchanged for an Access Token.
- **Implicit Grant**: A simplified flow where the Access Token is returned directly to the Client.
- **Authorization Code Grant with Proof Key for Code Exchange (PKCE):** This authorization flow is similar to the Authorization Code grant, but with additional steps that make it more secure for mobile/native apps and SPAs.
- **Resource Owner Credentials Grant Type**: This grant requires the Client first to acquire the resource owner's credentials, which are passed to the Authorization server. it is applicable in the use cases where a redirect is infeasible.
- **Client Credentials Grant Type**: Used for non-interactive applications e.g., automated processes, microservices, etc. In this case, the application is authenticated per se by using its client id and secret.
- **Refresh Token Grant**: The flow that involves the exchange of a Refresh Token for a new Access Token.

How OAuth 2.0 Work?

- The Client must acquire its own credentials, a client id and client secret, from the Authorization Server in order to identify and authenticate itself when requesting an Access Token.
- Using OAuth 2.0, access requests are initiated by the Client, e.g., a mobile app, website, smart TV app, desktop application, etc. The token request, exchange, and response follow this general flow:

1. The Client requests authorization (authorization request) from the Authorization server, supplying the client id and secret to as identification; it also provides the scopes and an endpoint URI (redirect URI) to send the Access Token or the Authorization Code to.
2. The Authorization server authenticates the Client and verifies that the requested scopes are permitted.
3. The Resource owner interacts with the Authorization server to grant access.
4. The Authorization server redirects back to the Client with either an Authorization Code or Access Token, depending on the grant type, as it will be explained in the next section. A Refresh Token may also be returned.
5. With the Access Token, the Client requests access to the resource from the Resource server.