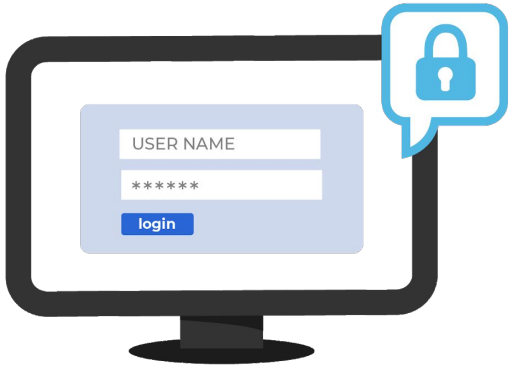
Module 2 - Lecture 16

# Authentication & Authorization

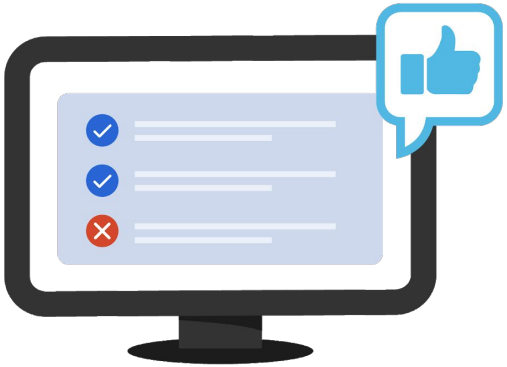
# Authentication vs Authorization

## Authentication




Confirms users
are who they say they are.

## Authorization



Gives users permission
to access a resource.

okta

# Authentication vs Authorization

- Authentication should be centralized.
  - Implement it once and forget about it.
  - Use verified techniques. Do not invent your own way.

- Authorization is often distributed.
  - Different segments of your application will be accessible by different users.
  - More room for user (software developer) error.
  - Always consider the data.
    - Who should have access? Read-only or write?

# Authentication vs Authorization

HTTP Status codes

| Status Code | Description | Cause |
| --- | --- | --- |
| `401` | `Unauthorized` | Occurs when you don't supply authentication credentials. |
| `403` | `Forbidden` | Occurs when your credentials aren't authorized for a resource. |

# The Roles

- **Identity Provider:** The provider of authentication. Also is the keeper of the authorization roles/claims that the end-user has.

- **Service Provider:** The provider of the service that the end-user is looking to access.

- In the past, these were always the same software development team.
- Nowadays, we often use third-party identity providers, such as Google, GitHub, Facebook, etc.
- Even large organizations like to use a single identity provider for handling auth.

# Authentication


Multifactor authentication
Time
Something you have
Something you are
Something you know
Location
*****
55984

# Best Practices: Authentication Failures

- Don’t tip your hand.
  - Bad: The username is incorrect.
  - Good: Login failed.

- Make them wait.
  - After 5 failed attempts. Lock the user out for a period of time.
  - Each attempt takes longer to return an error.

- Log the user out if they are idle.
  - Avoid them leaving their device and having their session hijacked.

# HTTP Authentication Mechanisms

- HTTP Basic Authentication
  - Username and password sent in HTTP Authorization Header.

- HTTP Bearer Token Authentication
  - Token issued by server.
  - Token sent in HTTP Authorization Header for subsequent requests.

- Session
  - Server generates and issues Session ID.
  - Typically stored in a cookie.

- Single-Sign-On / Federated (SAML, OpenID Connect)
  - One service provides authentication for many other services.
  - A token is issued by the server and the client sends it in subsequent requests (sometimes uses HTTP Bearer)

# Authorization

# Authorization Techniques

Authorization rules are customized by the server-side program.

Two techniques are commonly used.

- **Claims**
  - A claim is a customized set of key-value pairs to define attributes of your users. Claims can then be used to determine what a user is capable of doing.

- **Roles**
  - A user has zero to many roles, such as Administrator, User, HR, Engineer, etc.
  - Used to provide an umbrella of functionality to a type of user.

# Authorization Techniques

Less common

- **Whitelisting**
  - List of users (often IP addresses) that are permitted to make requests to a service. Typically used between trusted applications with static IP addresses.

# Authorization Mechanisms

- JSON Web Tokens (Client-side)
  - Digitally signed, encoded, and, optionally, encrypted JSON.
  - Should be stored securely.
    - Stored in the client's browser via a cookie or in-memory.
  - Subsequent HTTP requests send the token via HTTP Authorization Header.
  - Cons: Large when compared to session IDs.

- Session (Server-side)
  - Unique identifier stored in a cookie in the client's browser.
  - Subsequent HTTP requests send the cookie for the server to use.
  - Cons: Not as scalable when compared to JWTs. Cannot be used across domains.

# Cookies

- An HTTP cookie (web cookie, browser cookie) is a small piece of data that a server sends to the user's web browser.

- Remembers stateful information.
- Sent with every HTTP request.
- Specific to the browser. Private to the domain.
- 4KB maximum.
- Better mechanisms for general storage nowadays (Web Storage API)
- Viewable in DevTools

- Typical use cases:
  - Session management
  - Tracking user behavior
  - Personalization

# QUESTIONS?