Week 3: CSRF & CORS

- **Cross-Site Request Forgery (CSRF)**

What is it?

- CSRF occurs when an attacker tricks a user into making an **unauthorized request** to a web application in which the user is authenticated.
- This can perform actions like transferring funds, changing account settings, or submitting forms without the user's knowledge.

How to find it?

- Test forms (login, password reset, etc.) by sending forged requests to see if they process actions without authentication.
- Look for **state-changing requests** that don't have additional protection (e.g., tokens).

How to prevent it?

- Use **Anti-CSRF tokens** (e.g., in form submissions or AJAX requests).
- Check the **Referer** header to ensure requests are coming from valid sources.
- Implement **SameSite** cookie attribute to restrict cross-origin requests.

🔹 Cross-Origin Resource Sharing (CORS)

What is it?

- CORS is a security feature that allows or restricts resources to be requested from a different domain.
- Misconfigured CORS can lead to unauthorized domains accessing sensitive data or making requests on behalf of a user.

How to find it?

- Test by making **cross-origin** requests (e.g., from one domain to another) and check if sensitive data is exposed.
- Look for **wildcard (*)** in CORS headers that allow requests from any origin.

How to prevent it?

- Set strict CORS policies by allowing only trusted domains (e.g., Access-Control-Allow-Origin: https://trusted.com).
- Avoid using **wildcard** for Access-Control-Allow-Origin.
- Ensure that **credentials** (cookies, HTTP authentication) are handled properly in CORS requests.