Cross-Site Request Forgery (CSRF) Overview

CSRF stands for Cross-Site Request Forgery. It is a type of web security vulnerability where an attacker tricks a logged-in user into performing unwanted actions on a web application without their knowledge.

■ Real-world Example:

Suppose a vulnerable banking app has a transfer endpoint: POST /transfer?amount=1000&toAccount;=12345 Cookie: session=xyz An attacker can craft a malicious request, for example by embedding a link or hidden request in another site. If you are logged in, your session cookie is sent, and the transfer goes through.

■ How to Prevent CSRF in ASP.NET Core:

ASP.NET Core has built-in Antiforgery protection. 1. Generate Tokens - A token is issued by the server and stored in a cookie + sent in response. - The frontend must include it in a header or hidden form field. 2. Validate Tokens - Apply [ValidateAntiForgeryToken] on actions. - The framework compares the header token and the cookie token. If they don’t match, the request is rejected.

■ Example in ASP.NET Core MVC:

[HttpPost] [ValidateAntiForgeryToken] public IActionResult UpdateProfile(UserModel model) { // CSRF protected return Ok(); }

■ Example in ASP.NET Core Web API:

In Program.cs: builder.Services.AddAntiforgery(options => { options.HeaderName = "X-XSRF-TOKEN"; }); Then in the frontend (Angular/React), include the token in requests.

■ Summary:

- CSRF = attacker tricks a logged-in user into making unintended requests. - Defenses = Antiforgery tokens, SameSite cookies, and sometimes double-submit cookie strategy.