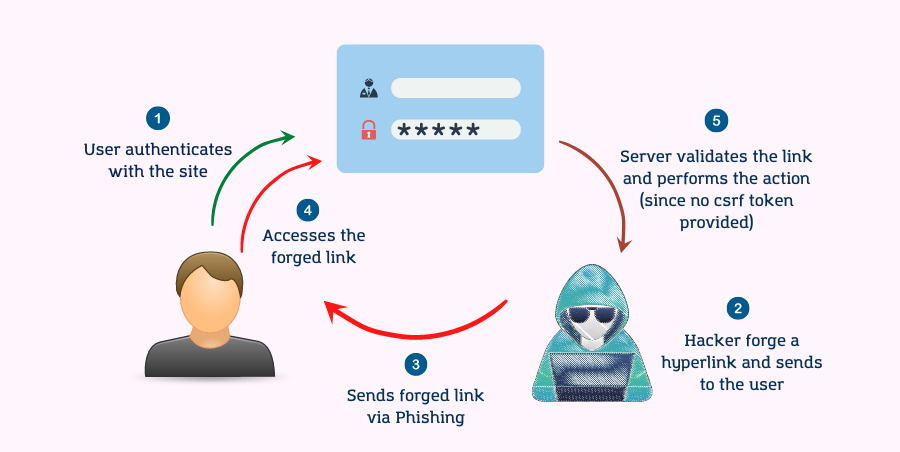
**Cross Site Request Forgery (CSRF)**

Cross-Site Request Forgery (CSRF) is an attack that forces authenticated users to submit a request to a Web application against which they are currently authenticated. For example, this might be to change the email address on their account, to change their password, or to make a funds transfer.

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they’re currently authenticated. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker’s choosing.



**Prevent Cross-Site Request Forgery (XSRF/CSRF) attacks in ASP.NET Core**

To help prevent CSRF attacks, ASP.NET core uses anti-forgery tokens, also called request verification tokens.

1. The client requests an HTML page that contains a form.
2. The server includes two tokens in the response. One token is sent as a cookie. The other is placed in a hidden form field. The tokens are generated randomly so that an adversary cannot guess the values.
3. When the client submits the form, it must send both tokens back to the server. The client sends the cookie token as a cookie, and it sends the form token inside the form data. (A browser client automatically does this when the user submits the form.)
4. If a request does not include both tokens, the server disallows the request.

**Code Implementation in ASP.NET Core API**

1. Create a new .net core API project.
2. Install the Nuget Package for Antiforgery – Microsoft.AspNetCore.Antiforgery
3. Created a custom middleware as AntiforgeryMiddleware, and injected into startup project.
4. Injected the Antiforgery dependency of startup services.

The header name is the request header cookies name which we accept the cookies from header from client side.

builder.Services.AddAntiforgery(option =>

{

option.HeaderName = "X-XSRF-TOKEN";

});

1. Add the Antiforgery globally we need to inject in the dependency of startup services : ValidateAntiForgeryTokenAttribute.

builder.Services.AddControllersWithViews(options =>

{

options.Filters.Add(new ValidateAntiForgeryTokenAttribute());

});

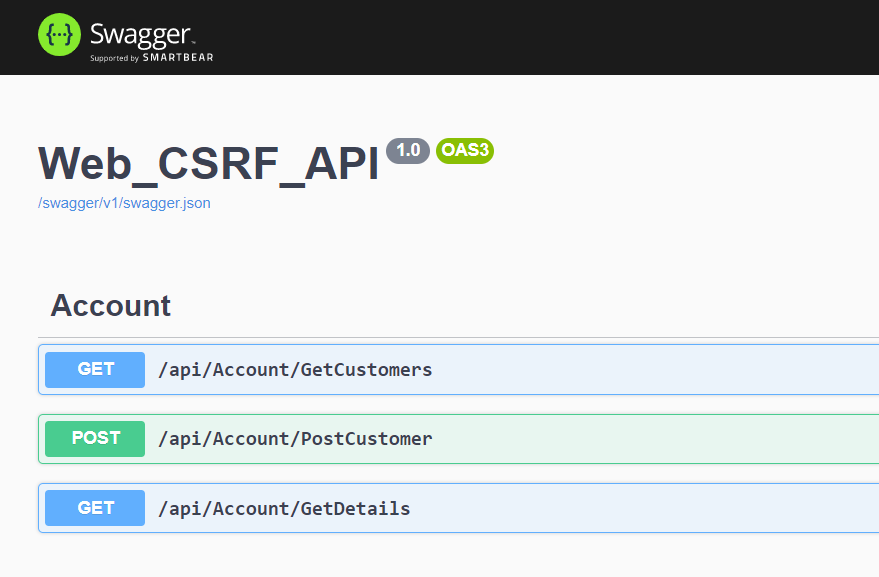
1. Create the controllers and decorate with ValidateAntiForgeryToken attribute.

[ValidateAntiForgeryToken]

1. If any action method needs to without validation use this attribute.

[IgnoreAntiforgeryToken]

1. To run this application, build the project and run the application.



**Code Implementation in Angular**

1. Create a new project in angular.
2. Add new interceptor to read the API cookies and copy/clone the cookies in client side browser.

intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

    const cookieheaderToken = 'X-XSRF-TOKEN';

    let csrfToken = this.httpXsrfTokenExtractor.getToken();

    if (csrfToken !== null && !req.headers.has(cookieheaderToken)) {

        req = req.clone({

          withCredentials: true,

          headers: req.headers.set(cookieheaderToken, csrfToken)

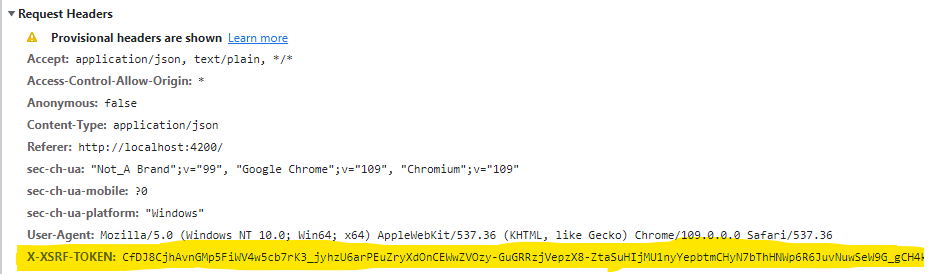
        });

    }

    return next.handle(req);

  }

1. Create GET/POST method in client side to call server side API.
2. In request we can see that in request header this token X-XSRF-TOKEN has been sent.



1. Added a proxy file with proxy.conf.json on root folder. This code will

re-rooted API application path.

{

  "/api": {

    "target": "https://localhost:7194",

    "secure": false

  }

}

1. This proxy file needs to configure in package.josn file under the scripts.

"start": "ng serve --proxy-config proxy.conf.json",

1. To run the application use the following command

npm start

This command will build and run angular project and assign a url to application, open the Url in browser.

