What Exactly Is CSRF?

CSRF (or XSRF) is also known as cross-site request forgery. As the name suggests, CSRF is a kind of attack on sites mostly done by other (malicious) sites, or sometimes by a (malicious) user on the site.

Typically, there are many cases where a site would require a user to fill in data from another website on behalf of that particular user. An example is how a lot of blogs use Disqus to power their commenting systems. To comment in that particular blog, the blog requires that you log in to Disqus first. This is a basic use of a CDN (content delivery network), and this example is a legitimate cross-site request

CSRF attacks usually rely on the user's identity. So what happens when a user visits a malicious website? That site sends hidden forms of some JavaScript XMLHttpRequest. That request uses the credentials of a user (one who visited their malicious site) to do some actions on another website that trusts the user’s browser or identity.

A site usually identifies authenticated users by saving cookies with headers and content that represent that particular user in their browsers. Attackers use this to access the user’s credentials to perform their attacks.

CSRF in Django

Powered by Python, Django is a free and open-source web framework that allows you to develop secure and maintainable websites in no time. Experienced developers built Django with the aim of reducing the unnecessary hassles of web development. This way, you can focus on building without having to reinvent the wheel. It's suitable for back-end and front-end development.

Middleware in Django is a set of functions that run during request and response processes. And in Django, there’s CSRF middleware that helps protect against CSRF attacks in Django apps.

When you start a Django project, you’ll see in your **settings.py** file that the middleware has been activated by default.

How Does the CSRF Token Work?

The CSRF token is like an alphanumeric code or random secret value that's peculiar to that particular site. Hence, no other site has the same code.

In Django, the token is set by **CsrfViewMiddleware i**n the **settings.py** file.

A hidden form field with a **csrfmiddlewaretoken** field is present in all outgoing requests. When you submit a form to the server that it didn't send to you, the server won't accept it—unless it has the CSRF token that matches the one the server recognizes.

The server has its own CSRF token. That's what it sends, along with a form to the client for protection of information.

All incoming requests must have a CSRF cookie, and the **csrfmiddlewaretoken** field must be present and correct. Otherwise, the user will get a 403 error.

Enabling Django CSRF Protection With Django REST and React

What about cases where you're using Django REST and a separate front-end framework, such as React? You'll definitely want to ensure that you have CSRF protection enabled as well. Here's how.

Conclusion

You've now learned what CSRF protection is and how to enable it in Django. This means you're one step ahead of security attacks and malicious attempts against your users.