Cross-Site Request Forgery (CSRF) Cross-Site Request Forgery (CSRF) is a security vulnerability where an attacker tricks a user's browser into making unintended requests to a web application, potentially performing actions without the user's knowledge. This can lead to unauthorized actions, such as changing user settings, transferring money, or deleting data.

Real-Life Scenario:

Imagine you're logged into your online banking account and browsing another website in a different tab. If that other website contains malicious code, it could cause your browser to make a request to your bank, transferring money from your account without you realizing it. This happens because your browser sends your bank's authentication cookies with the request, making it seem like it came from you.

What Happens?

In CSRF, an attacker uses a trusted user's authentication to perform actions on their behalf. Since the user's browser sends authentication tokens (like cookies) with every request, the malicious request appears to be legitimate. This can lead to unauthorized changes, such as updating account information, initiating financial transactions, or manipulating sensitive data.

How That Happens:

CSRF typically involves a third-party website that embeds a hidden request to another application where the user is logged in. This can be achieved through:

- **Hidden Forms**: The attacker creates an invisible form that automatically submits with the user's credentials.
- **Scripted Requests**: A script generates a request that is executed without user interaction.
- **Embedded Images/Links**: An attacker can use an image tag or link to initiate a request when the user visits their website.

Example:

An attacker creates a hidden form on their website that sends a request to change a user's email address in their profile. If the user is logged into the target site, their browser will automatically send the authentication cookies with the request, allowing the change to occur without the user's knowledge:

```
<form action="https://bank.com/change_email" method="POST"
style="display: none;">
```

```
    <input type="hidden" name="email" value="attacker@example.com">
    <input type="submit" value="Submit">
</form>
<script>document.querySelector('form').submit();</script>
```

When the user visits this malicious page, the form submits, changing the email on their bank account without their knowledge.

How Can I Prevent That?

To prevent CSRF, you can implement several key measures:

1. **CSRF Tokens**: Include a unique token with each request that requires user authentication. The server checks this token to ensure the request is coming from a legitimate source. This token is typically sent as a hidden field in forms or as a header.
2. **SameSite Cookies**: Use the `SameSite` attribute for cookies to restrict them to same-site requests. This reduces the risk of CSRF by preventing cookies from being sent with cross-site requests.
3. **Use HTTP Headers to Validate Requests**: Require specific HTTP headers (like `Referer` or `Origin`) to ensure requests come from trusted sources. This can help detect and prevent cross-site requests.
4. **Use Double-Submit Cookies**: This method involves sending a cookie and a token with every request. The server checks that they match, providing additional protection against CSRF.
5. **User Education**: Educate users about the risks of CSRF and encourage them to be cautious when clicking on links or visiting untrusted websites while logged into sensitive accounts.

In Summary:

Cross-Site Request Forgery is a significant security risk that allows attackers to perform actions on behalf of authenticated users. To prevent CSRF, use CSRF tokens, implement SameSite cookies, validate HTTP headers, and consider double-submit cookies. By taking these measures, you can safeguard your applications and users from CSRF attacks.