**CSRF - Cross-Site Request Forgery**

CSRF- межсетевая уязвимость POST запросов. Дело все в том, что, когда клиент прошел аутентификацию, в куках сохраняется JSESSIONID и может получится так, что во время исполнения POST запросов, злоумышленник может перехватить эту куку, тем самым перехватит и POST запрос (в качестве post запроса могут быть операции по переводу денег).

**CSRF уязвимость может влиять только на POST, PUT, PATH, DELETE запросы, на GET, TRACE, HEAD, OPTIONS запросы CSRF не влияет. CSRF опасен если мы используем cookies для идентификации пользователя. Если cookie не использовать, то CSRF фильтр нам не нужен**

**Решение проблемы CSRF:**

Synchronized Token pattern – **именно этот способ использует Spring**, когда мы добавляем специальный параметр в качестве hidden параметра либо в header. На деле в нашу форму прикрепляется специальный параметр, который называется \_csrf и этот параметр генерирует токен, который хранится на сервере. Злоумышленник не знает ничего про этот токен (он не может получит значение этого токена)

SameSite Attribute – этот способ предполагает, что мы использует атрибут cookie, который так и называется SameSite.

В Spring по умолчанию CSRF filter включен, но, чтобы его за использовать нам нужно во всех POST запросах использовать обращение к методам контроллера с помощью thymeleaf **через th: action=” @ {/users/update}”** либо писать следующее.

1. <form th:action="@{/login}" method="post"> такой способ автоматически обезопасит наше приложение от CSRF угрозы.
2. Второй способ менее предпочтителен: нам нужно самим указать \_csrf.parameter и csrf.token

