## Описание защиты

1. В качестве защиты от CSRF атак я использовал CSRF токен. При CSRF атаке пользователю навязывается скрытая форма, поля которой идентичны полям легальной формы. При нажатии на форму злоумышленника от имени авторизованного пользователя происходит обращение на целевой сайт, где данные могут быть изменены в угоду автора скрытой формы. Для предотвращения такой ситуации на стороне сервера при очередном запросе формы генерируется ключ, на основании которого с помощью хеш-функции SHA256 вычисляется токен. Использование хеш-функции дает теоретическую невозможность по токену найти ключ. Токен отправляется вместе со страницей, где расположены поля формы. При получении данных от пользователя проверяется, что токен, пришедший от пользователя, не изменился, и данные считаются осознанно введенными легальным пользователем. После этого ключ стирается с сервера. Таким образом, при случайном нажатии пользователем на отправку данных с секретной формы, эти самые данные будут отброшены сервером, так как либо ключ, на основании которого получается токен, не сгенерирован, либо токен с вероятностью, близкой к единице, будет отличаться от того, который может быть навязан злоумышленником.
2. В качестве защиты от подмены данных полей формы я использовал механизм проверки целостности HMAC на основании хеш-функции SHA256. На стороне клиента по введенным данным, токену и ключу, сгенерированным сервером и переданным пользователю, например, по телефону с помощью SMS, вычисляется HMAC код. Он также передается на сервер. Обратить HMAC код теоретически невозможно. При получении данных сервер также вычисляет HMAC код и проверяет его с принятым от клиента. Если они совпали, то данные считаются не измененными. Ключ удаляется с сервера. Предположим, что-то по пути от клиента к серверу данные были изменены злоумышленником. Чтобы сервер принял данные, злоумышленнику нужно также вычислить HMAC код, но он не знает ключа, на основании которого генерируется HMAC код. В таком случае ему необходимо узнать или перехватить этот ключ. Таким образом, неприятель не сможет легким способом навязать серверу измененные пользовательские данные.
3. Также для повышения безопасности, я думаю, стоит использовать защищенное https соединение. Данные с формы, а также токен, будут зашифрованы.
4. В качестве защиты от подменны данных в AJAX запросе я также применил HMAC код на хеш-функции SHA256. Данными выступают конкатенированные значение поля Amount и значение CSRF токена. В качестве ключа HMAC я использовал имитацию хеша пароля пользователя, который пользователь гипотетически ввел при входе в свой личный кабинет. Принцип защиты тот же самый, что и защиты от подмены параметров при отправке формы на сервер (п. 2). В данные при AJAX запросе также добавлен CSRF токен, чтобы убедиться в том, что данные прибыли с ранее отправленной конкретному пользователю формы.