XSS-атаки. Устранение уязвимостей.

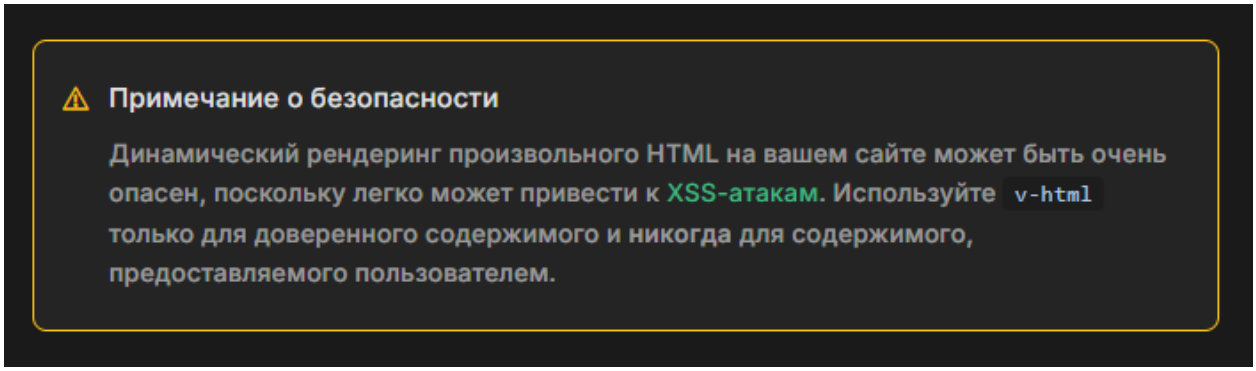
Данная тема является заключительной в текущем обучении. Здесь Вы узнаете про XSS-атаки и методы противодействия им. Это очень важно знать каждому фронтенд-разработчику, поскольку за незнание этого увольняют с работы. Также Вы завершите работу над проектом notes-app, освоив за всё обучение базовый функционал фреймворка Vue.

Эта тема полностью посвящена XSS-уязвимостям. XSS (Cross-Site Scripting) – это подтип атаки на веб-системы, заключающийся во внедрении в выдаваемую веб-системой страницу вредоносного кода (который будет выполнен на компьютере пользователя при открытии им этой страницы) и взаимодействии этого кода с веб-сервером злоумышленника.

Данная уязвимость возникает, когда используются необработанные данные, которые могут быть введены пользователем, например, в поля HTML-формы, в query-параметры запроса или иными способами.

Например, если Вы используете генерацию DOM-дерева с использованием JS, например, element.innerHTML [4], и берёте содержимое для вставки из поля <textarea> или <input>, то злоумышленники могут вставить в текст код в виде HTML-тэгов с обработчиками событий, где будет исполняться вредоносный код.

Фреймворк Vue не защищает от таких уязвимостей. Одна из таких опасных директив это v-html [5]. Разработчики Vue даже вынесли предупреждение в своей документации:



Обязательно ознакомьтесь с материалом в статье [7], посвящённой введению в XSS-уязвимости. Там также представлена демо-страница с уязвимостью, где Вы можете самостоятельно произвести атаку. Настоятельно рекомендуется это проделать.

Вообще, сами по себе element.innerHTML и директива v-html могут быть безопасно использованы в проекте, если код генерируется в JS-скрипте и не зависит ни каким образом от пользовательского ввода. Но если необходимо использовать значения, вводимые пользователем, то необходимо произвести их обработку. Один из вариантов такой обработки – экранирование символами-мнемониками [6]. Это особенно необходимо в случае, если требуется вывести фрагмент кода, фрагмент HTML или XML разметки. Другие варианты решения специфичны для каждого конкретного случая и должны быть рассмотрены индивидуально. Один из таких вариантов Вам предстоит придумать и реализовать самостоятельно в одной из задач к данной теме.

Итак, главное, что Вы должны понять из прочитанного – искать XSS-уязвимости и устранять их нужно всегда в тех случаях, когда данные могут быть введены пользователем (это может быть любое из полей ввода, либо адресная строка).

Вопросы и задания

1. Добавьте функционал удаления заметки из списка в приложении notes-app. Для этого создайте компонент модального окна, необходимого для подтверждения пользователем удаления заметки. Макеты доступны по прежней ссылке [1]. Допишите функцию запроса на удаление, для которой эндпоинт указан в файле README.md [2]. После подтверждения удаления в модальном окне должно произойти обновление списка выводимых заметок.

2. Вернитесь к работе над карточкой заметки в директории /src/views/notes/note.vue. Доработайте css-стили так, чтобы текст заметки имел форматирование – абзацные отступы (табуляцию) и кликабельные ссылки (гипертекст).

   Если в тексте заметки встречаются символы переноса строки «\n» или каретки «\r» или их комбинация «\r\n», а также символы «\t», то

текст должен иметь соответствующий вид. Эти символы возникают при вводе текста в элемент <textarea>.

Если в тексте встречается ссылка, то она должна выделяться другим цветом на фоне текста и быть кликабельной. Для этого можете воспользоваться регулярными выражениями.

Самостоятельно придумайте, как реализовать данный функционал.

3. Сможете ли найти XSS-уязвимость в недавно реализованном функционале? Если Вы все сделали правильно, то такой уязвимости нет. В этом случае приведите рабочий пример кода, когда достигается такая уязвимость. Также приведите пример успешной атаки на приложение (можно и безобидную, главное, чтобы работало на практике). Если же у Вас уже реализована уязвимость, то также выполните предыдущие действия, но дополнительно постарайтесь исправить критическую ситуацию.

4. Создайте локально SFC-компонент модального окна с выводом полноэкранного изображения, прикрепленного к заметке, согласно макетам [1].

5. *Самостоятельно реализуйте маршрутизацию с применением функционала npm-пакета Vue-Router. Он уже установлен в проекте notes-app, можете убедиться в этом, заглянув в файл package.json. Создайте страницу со списком удалённых заметок (архив), переход к которой осуществляется из основной страницы через кнопку «Архив» (см. макеты в [1]). Реализуйте пагинацию на этой странице.

   Для этого самостоятельно ознакомьтесь с документацией Vue-Router [3].

Ссылки

[1]. https://www.figma.com/board/UK7yR8nqWpGJlkJGl28chk/%D0%A8%D0%BA%D0%BE%D0%BB%D0%B0_frontend_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%87%D0%B8%D0%BA%D0%B0?node-id=0-1&p=f&t=bOSL2DflJq0mNLAH-0

[2]. https://github.com/GreysMouse/notes-app-server

[3]. https://router.vuejs.org/guide/

[4]. https://developer.mozilla.org/ru/docs/Web/API/Element/innerHTML

[5]. https://ru.vuejs.org/api/built-in-directives#v-html

[6]. https://htmlacademy.ru/courses/301/run/8

[7]. https://habr.com/ru/articles/511318/