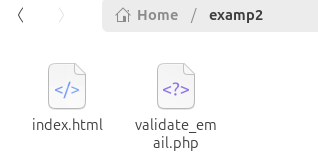
1. Собрали html - страницу, подняли локально php на :5000



//заранее сделали php для валидации почты, ибо визуально внутренняя обработка выглядит простенько. Немного переделал код для эксплуатации, ибо изначальный код не смог развернуть, из-за смешивания php с html (максимально смысл оставил к изначальному). Можно было бы поделить ещё на клиент – сервер, но не стал. Использовал только index.html (пример.2)

1. Кнопка «Subscribe» вызывает JS – функцию Newsletter()
2. Переменная email вставляется в DOM через innerHTML. Отсюда – недостаточная обработка входных данных позволяет произвести XSS – атаку.

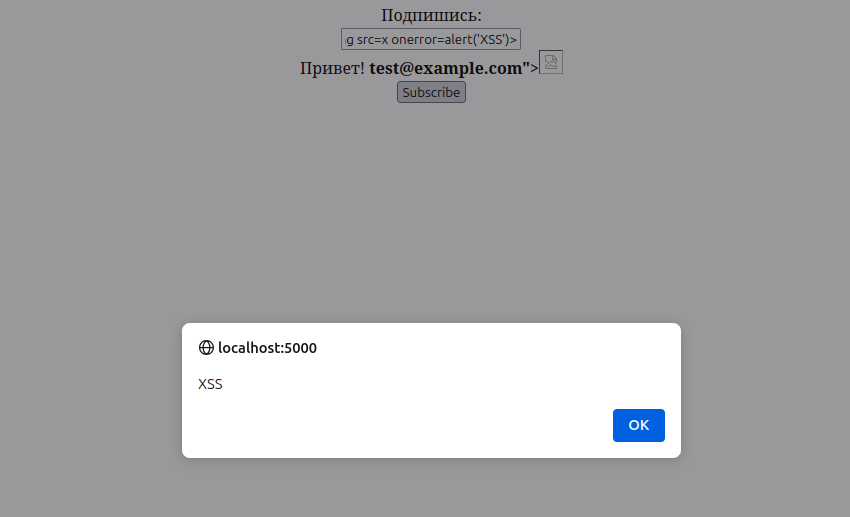
**КАК ВОСПРОИЗВЕСТИ:**

1. Вводим в поле email вредоносный код:

*test@example.com"><img src=x onerror=alert('XSS')>*

*//пытался использовать <script> код, но судя по наблюдениям, сам браузер спасает от такого даже при innerHTML*

1. Получаем ожидаемый результат успешной эксплуатации XSS:



**Как защитить:**

1. Изменим метод вставки данных с innerHTML на textContent:

*if (emailInput.split("@").length == 2) {*

*out.textContent = 'Привет! ' + emailInput;*

*} else {*

*out.textContent = 'Некорректный email!';*

1. Улучшим валидацию почты. А именно, нам нужно ограничить некорректный пользовательский ввод, который может избавить как от «мусорных» данных, так и от возможности реализации XXS. Один из вариантов:

*function isValidInput(input) {*

*const regex = /^[a-zA-Z0-9@.\_-]+$/; // Разрешены только безопасные символы*

*return regex.test(input);*

*}*

1. Добавим экранирование. Даже при вводе "><script>alert('XSS');</script>, код не выполнится из-за экранирования ввода. Пример на JS:

*function escapeHtml(unsafe) {*

*return unsafe.replace(/[&<>"']/g, function (c) {*

*return {*

*'&': '&amp;',*

*'<': '<',*

*'>': '>',*

*'"': '&quot;',*

*"'": '&#039;'*

*}[c];*

*});*

*}*

*const emailInput = document.getElementById('email').value;*

*const safeEmail = escapeHtml(emailInput);*

*document.getElementById('out').textContent = 'Привет! ' + safeEmail;*

В данном случае браузер отобразит пользовательский ввод, как текст, и не выполнит вредоносный JS.