Доклад

Ошибки проверки вводимых данных: межсайтовый скриптинг в веб-приложениях, межсайтовый скриптинг при наличии SQL-инъекции

Зинченко А.Р

14 мая 2025

Российский университет дружбы народов, Москва, Россия

НБИбд-02-23

Информация

Докладчик

- Зинченко Анастасия Романовна
- НБИбд-02-23
- Российский университет дружбы народов

Введение

Введение

Проверка вводимых данных — это один из ключевых аспектов информационной безопасности веб-приложений. Неправильная или недостаточная проверка пользовательского ввода может привести к серьёзным уязвимостям, включая межсайтовый скриптинг (XSS) и SQL-инъекции.

Межсайтовый скриптинг (XSS)

Межсайтовый скриптинг (XSS)

XSS (Cross-Site Scripting) — это тип уязвимости, при которой злоумышленник внедряет вредоносный скрипт на сайт. Этот скрипт затем выполняется в браузере другого пользователя, обычно без его ведома.

Разновидности XSS:

1. Отражённый (Reflected XSS) — вредоносный код возвращается в ответе сервера немедленно, часто через параметры URL.
2. Сохранённый (Stored XSS) — скрипт сохраняется на сервере и автоматически исполняется при открытии определённой страницы.
3. DOM-based XSS — уязвимость возникает на стороне клиента, когда JavaScript обрабатывает вход без должной фильтрации.

SQL-инъекция и её связь с XSS

SQL-инъекция и её связь с XSS

SQL-инъекция — это тип атаки, при которой злоумышленник внедряет SQL-код в поле ввода, чтобы изменить поведение запроса к базе данных.

Классическая форма: SELECT * FROM users WHERE name = ‘ввод пользователя’;

Если пользователь введёт: ’ OR ‘1’=’1

Запрос станет: SELECT * FROM users WHERE name = ’’ OR ‘1’=‘1’;

XSS через SQL-инъекцию

XSS через SQL-инъекцию

Как это работает: Иногда SQL-инъекция позволяет злоумышленнику внедрить JavaScript-код прямо в базу данных. Если затем эта информация отображается на странице без экранирования — срабатывает XSS. Пример: Хакер вводит

в поле комментария. Комментарий сохраняется в БД. На странице отображается:

Комментарий:

Браузер пользователя исполняет скрипт, возможно, передавая куки злоумышленнику.

Последствия:

1. Кража сессий.
2. Подмена интерфейса сайта.
3. Распространение вредоносного ПО.

Почему это происходит

Почему это происходит

1. Отсутствие валидации данных на сервере.
2. Доверие к пользовательскому вводу. (Пример: сайт получает имя пользователя из cookie: $username = $_COOKIE[“username”]; echo “Привет, $username!”; Хакер меняет cookie на
и скрипт исполняется при заходе на сайт.)

Почему это происходит

3. Отсутствие экранирования HTML/JS при выводе данных.
4. Использование небезопасных функций (например, eval в JS). (Опасный пример: $sql = “SELECT * FROM users WHERE name = ’” . $_GET[‘name’] . “‘“; Если злоумышленник передаст в параметре’ OR ‘1’=‘1, то получится: SELECT * FROM users WHERE name =’’ OR ‘1’=‘1’ в результате возвращаются все пользователи.)

Методы защиты

Методы защиты

##Для предотвращения XSS:

1. Экранирование HTML при выводе.
2. Использование безопасных шаблонизаторов.
3. Запрет выполнения скриптов с помощью Content Security Policy (CSP).
4. Проверка и фильтрация входных данных. (Примеры: использование регулярных выражений ([1]{3,30}$), проверка типов (int, email, url), запрет тегов (strip_tags() в PHP, DOMPurify в JS))

[1] a-zA-Z0-9_

Для предотвращения SQL-инъекций:

1. Использование подготовленных выражений (prepared statements).
2. Использование ORM. (Примеры: Django ORM (Python), Hibernate (Java), Sequelize (Node.js).)
3. Изоляция пользовательского ввода.
4. Минимизация прав учетной записи базы данных.

Общие рекомендации:

1. Никогда не доверяйте данным от пользователя.
2. Проверка данных на клиенте — это удобно, но не надёжно, необходима серверная проверка.

Заключение

Заключение

Межсайтовый скриптинг и SQL-инъекции — это старые, но всё ещё крайне опасные уязвимости, часто возникающие из-за одних и тех же причин: доверия к пользовательскому вводу. Их комбинация может иметь катастрофические последствия — от утечки персональных данных до полного захвата веб-приложения.

Выводы

Выводы

Комплексная защита, основанная на принципах нулевого доверия, экранирования, фильтрации и безопасных API — ключ к устойчивой архитектуре веб-приложений.

Список литературы

Список литературы