МЕТОДИ ТА ЗАСОБИ САНИТИЗАЦІЇ ДАНИХ MARKDOWN ТА HTML ДЛЯ ЗАХИСТУ ВЕБ-ДОДАТКІВ ВІД МАНІПУЛЮВАННЯ ВХІДНИМИ ДАНИМИ

Пантелей Д.О., Гріненко Т.О.
Харківський національний університет радіоелектроніки, Харків, Україна

Широке використання веб-додатків із можливістю створення користувацького контенту супроводжується ризиком маніпулювання вхідними даними. Формати розмітки Markdown і HTML, які підтримують вбудований код, можуть використовуватись зловмисниками для Cross-Site Scripting (XSS)-атак та підміни контенту. Неконтрольована обробка користувацьких даних є одним з основних джерел компрометації веб-додатків [1]. Існуючі засоби фільтрації переважно орієнтовані на кінцевий HTML-код, що не виключає збереження шкідливих фрагментів у вихідному Markdown.

Метою доповіді є порівняльний аналіз методів та засобів забезпечення захисту веб-додатків від маніпулювання вхідними даними, що надходять у форматах Markdown та HTML. **В доповіді** наводяться результати дослідження та порівняльного аналізу існуючих рішень з HTML-санитизації, оцінка ефективності їх роботи; побудовані модель порушника та модель загроз.

Для ідентифікації загроз використано методологію STRIDE [2, 3]. Відповідно до класів загроз, найбільш значними є:

– підробка ідентифікаційної інформації – вставка підробленої форми для виконання певної дії (наприклад, входу);

– втручання в дані – зміна даних в межах сторінки на стороні клієнта;

– розголошення інформації – зчитування даних в межах сторінки на стороні клієнта;

– відмова в обслуговуванні – ініціювання ресурсозатратного процесу для значного погіршення або унеможливлення відображення сторінки в браузері;

– підвищення привілеїв – отримання локально збережених параметрів доступу (сесійні параметри, токени) з document.cookie, localStorage, sessionStorage.

Для систематизації векторів атак, зокрема Cross-Site Scripting (XSS), застосовано матрицю MITRE ATT&CK. Встановлено, що Markdown–HTML-ін'єкції зазвичай відповідають наступним технікам [4]:

1. T1190 «Exploit Public-Facing Application»;
2. T1027 «Obfuscated Files or Information (обхід фільтрів)»;
3. T1059.007 «JavaScript Execution»;
4. T1539 «Steal Web Session Cookie»;
5. T1565 «Data Manipulation».

Результати дослідження показали, що Markdown дозволяє вбудовувати довільні HTML-елементи (<script>, <iframe>, <svg>) та обробники подій (onload, onerror, onclick), які можуть використовуватись для ін'єкції

шкідливого коду. Встановлено, що основними техніками обходу санитайзерів є обфускація тегів, кодовані символи та маніпуляції з протоколами (javascript:, data:).

Неконтрольоване перетворення Markdown у HTML може призвести до:

– викрадення облікових даних користувача через XSS;
– несанкціонованого доступу до сесій;
– підміни вмісту сторінок або фішингових вставок;
– використання ресурсів клієнта для виконання шкідливих дій (наприклад, криптомайнінгу).

Сучасні системи здебільшого реалізують фільтрацію лише на етапі виведення HTML. Основними архітектурними підходами є [5]:

1. Server-Side Rendering (SSR) – коли конвертація Markdown у HTML та санитизація виконуються на сервері перед відправленням користувачу. Типові рішення: OWASP HTML Sanitizer (Java), Bleach (Python).

2. Client-Side Rendering (CSR) – коли Markdown рендериться безпосередньо у браузері користувача, а санитизація виконується на стороні клієнта. Основний інструмент: DOMPurify (JavaScript).

Обидва підходи мають спільні недоліки: зберігання у сховищі необроблених даних із потенційними XSS-векторами; надлишкові витрати ресурсів через повторну санитизацію при кожному рендерингу; дублювання логіки безпеки між клієнтською і серверною сторонами.

Метою подальших досліджень є розробка та тестування модуля для санитизації Markdown та HTML, здатного ефективно захищати Web-додатки від ін'єкційних атак. Модуль дозволить вирішити зазначені вище проблеми за рахунок того, що запропонований підхід орієнтований на попередню обробку Markdown до моменту зберігання чи відображення. Це дозволить усунути ризики збереження небезпечного контенту та зменшить навантаження під час відтворення. Пропонується реалізація модуля як ізольованого компоненту, що буде легко інтегруватися в існуючі системи обробки контенту.

Список літератури

1. Д'якова, Н. Є., & Сєвєрінов, О. В. (2022). Тестування вразливостей сучасних веб-ресурсів.

2.The STRIDE Threat Model [Електронний Ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)?redirectedfrom=MSDN.

3. Моргуль Д.М., Нарєжній О.П., Гріненко Т.О. Модель порушника та модель загроз для веб-сервісу QRNG. *Radiotekhnika* No. 221. С. 31-38. DOI:10.30837/rt.2025.2.221.04.

4. Techinques – Enterprice | MITRE ATT&CK [Електронний Ресурс]. – Режим доступу: https://attack.mitre.org/techniques/enterprise/

5. SSR vs. CSR: Server-Side vs. Client-Side Rendering Explained [Електронний Ресурс]. – Режим доступу: www.shopify.com/blog/ssr-vs-csr.