Notatka

Autor: Oliwier Markiewicz

⟹ Cyberbezpieczeństwo na Frontendzie

Podsumowanie

dangerouslySetInnerHTML – Co to jest i czemu to ryzyko?

- Jest to mechanizm Reacta pozwalający na bezpośrednie wstawienie HTML do komponentu:

```
<div dangerouslySetInnerHTML={{ __html: someHTML }} />
```

- **Dlaczego to ryzyko?**
  - Narażasz się na **XSS (Cross-Site Scripting)** – atakujący może wstrzyknąć złośliwy kod JS, który np. kradnie ciasteczka lub przejmuje sesję.
- **Jak się chronić?**
  ✅ Używaj **DOMPurify** lub innej biblioteki do sanitizacji:

```
import DOMPurify from 'dompurify';

<div dangerouslySetInnerHTML={{ __html: DOMPurify.sanitize(userHTML) }} />
```

- ✅ Zastanów się, czy w ogóle musisz używać dangerouslySetInnerHTML. Często można to zastąpić **Reactowym renderowaniem komponentów**.

2. Gdzie NIE przechowywać tokenów JWT?

- ❌ **LocalStorage** – podatne na ataki XSS, bo JS ma do niego dostęp.
- ❌ **SessionStorage** – ten sam problem.
- ✅ **HttpOnly Secure Cookies** – najlepsza praktyka:
  - Flagi HttpOnly, Secure, SameSite=Strict.
  - Uniemożliwia dostęp do ciasteczek z poziomu JS, co chroni przed XSS.
- **Dodatkowo:**
  - Stosuj **rotację tokenów** (krótkie TTL + refresh tokeny).
  - Rozważ **CSRF Protection** – np. token synchronizer lub SameSite.

3. Jak zabezpieczyć dane wejściowe?

- **Nigdy nie ufaj danym z frontu.**
- **Sanitacja i escapowanie:**
  - Usuń potencjalny kod HTML/JS z inputów.
  - Używaj whitelist (np. akceptuj tylko określone formaty).
- **Sprawdzaj typy danych:**
  - Email → regex i dodatkowa weryfikacja po stronie backendu.
  - Liczby → konwersja na typ numeryczny, walidacja zakresu.
- **Zasada podwójnej walidacji:**
  - Frontend (UX) + Backend (bezpieczeństwo).
- **Uwaga na uploady plików:**
  - Sprawdzaj rozszerzenie i MIME type po stronie serwera.
  - Nigdy nie zapisuj plików w katalogach publicznych bez sprawdzenia.

4. Dlaczego walidacja na froncie NIE wystarczy?

- Frontend jest w pełni kontrolowany przez użytkownika → **łatwe do obejścia** (DevTools, Postman, cURL).
- **Backend = jedyne źródło prawdy**.
- Przykład:
  - Jeśli walidujesz limit 100 znaków w React, ktoś może wysłać 10 000 znaków requestem.
- **Dlatego:**
  - Walidacja na froncie → UX.
  - Walidacja na backendzie → Bezpieczeństwo.

5. Najczęstsze błędy React Developerów w 2025:

- ❗ **Wystawianie kluczy API i danych env do sieci:**
  - Błąd: umieszczanie .env w repo publicznym.
  - Rozwiązanie: używaj **Vercel/Netlify Secrets**, backend proxy.
- ❗ **Brak Content Security Policy (CSP):**
  - CSP pozwala blokować złośliwe skrypty.
- ❗ **Używanie starych paczek bez aktualizacji:**
  - npm audit → raport bezpieczeństwa.
- ❗ **Logowanie danych w konsoli na produkcji:**
  - Może ujawnić tokeny lub poufne dane.
- ❗ **Brak kontroli nad dependencies:**
  - Złośliwe paczki w npm to realny problem → korzystaj z **Snyk** lub **OWASP Dependency-Check**.

BONUSY 🎁

Checklista Bezpiecznego React App:

✔ **Używaj CSP (Content Security Policy).**
✔ **Blokuj XSS (np. przez DOMPurify i helmet w backendzie).**

✔ **Tokeny JWT → HttpOnly Cookies.**
✔ **Weryfikuj dane po stronie serwera.**
✔ **Używaj HTTPS wszędzie.**
✔ **Aktualizuj paczki (np. npm audit).**
✔ **Konfiguruj SameSite, Secure w ciasteczkach.**

Przydatne Narzędzia i Biblioteki:

🔒 **DOMPurify** – sanitizacja HTML.

🛡️ **Helmet** – nagłówki bezpieczeństwa w Node.js.

🔍 **npm audit** / **yarn audit** – analiza podatności paczek.

🧪 **OWASP ZAP** – skanowanie aplikacji pod kątem dziur.

✅ **React Error Boundary** – lepsze zarządzanie błędami.