

XSS Prevention Reference

SVG Security: Preventing XSS & Injection Attacks

About This Document

This reference accompanies the article: SVG Security: Preventing XSS & Injection Attacks

Read the full article at: <https://svgeneration.com/docs/svg-security-best-practices>

Article Summary

How to safely handle user-supplied SVG, prevent cross-site scripting via SVG vectors, and sanitize pattern inputs.

Code example: // JavaScript: Always sanitize before injecting inline SVGimport DOMPurify from 'dompurify';const userSvg = getUserInput(); // untrustedconst clean = DOMPurify.sanitize(userSvg, { USE_PROFILES: { svg: true, svgFilters: true }, FORBID_TAGS: ['script', 'use'], FORBID_ATTR: ['href', 'xlink:href', 'onload', 'onclick'],}); // Only then inject:element.innerHTML = clean;

Content Security Policy for SVG

Implement a strict CSP that blocks inline script execution: Content-Security-Policy: script-src 'self'; object-src 'none'; base-uri 'none'. For SVG data URIs in CSS, allow img-src data:. Restrict default-src to prevent SVG images from loading external resources. Test your CSP with the Google CSP Evaluator.

Validating Pattern Inputs

If you allow users to customise pattern parameters (colours, sizes), validate all inputs server-side. Colour values must match `/^#[0-9a-fA-F]{3,6}$/`. Numeric values must be within documented ranges. Never interpolate user input directly into SVG templates without validation a malicious user could inject SVG markup through a colour input field.

Free Resources

Visit SVGeneration.com for 600 free SVG background patterns. No attribution required.

License: Modified MIT License. Use freely in personal and commercial projects.