

SVG Security Checklist

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.

Why SVG Is a Security Risk

SVG is XML and XML can contain scripts, event handlers, external references, and XSLT. A malicious SVG file can embed `<script>` tags that execute when the SVG is rendered as an HTML image, trigger HTTP requests to external servers, load external resources via `<image href>`, execute JavaScript via event handlers (onload, onclick), and exploit XSLT processing instructions.

Safe vs Unsafe SVG Usage

Safe: SVG as CSS background-image data URI the SVG is rendered in a sandboxed context; scripts and external references do not execute. SVG referenced via `<img src>` also sandboxed. Unsafe: SVG rendered as inline HTML (direct DOM injection) scripts execute in the page context. SVG in `<iframe src>` without sandboxing may execute scripts depending on CSP.

Sanitizing User-Supplied SVG

Never render user-supplied SVG as inline HTML without sanitization. Use DOMPurify (JavaScript) or SVG Sanitizer (PHP/Python). These libraries whitelist safe elements and attributes, stripping anything potentially dangerous. Configure DOMPurify with the SVG profile: `DOMPurify.sanitize(svg, { USE_PROFILES: { svg: true, svgFilters: true } })`.

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.