# CSP Checklist – WordPress & Joomla

Content Security Policy für klassische CMS-Plattformen

tenbyte

2025 / 2026

WordPress · Joomla · WooCommerce · Elementor

Critical – vor Go-Live · Important · Optional · WordPress · Joomla · Beide

**WordPress-Realität:** Plugins und Themes fügen unkontrolliert Inline-Scripts ein. Immer mit Report-Only starten und mindestens 2 Wochen Violations sammeln bevor blockiert wird.

**Joomla-Hinweis:** Ab Joomla 4.x gibt es native CSP-Unterstützung im Backend unter System → Plugins → System – HTTP Headers. Trotzdem gilt: Plugin-Inventur zuerst.

## 01 Vorbereitung – beide Systeme

WordPress & Joomla

- [ ] **Report-Only als zwingend ersten Schritt setzen** — Critical
  CMS-Systeme mit Plugins produzieren sehr viele Violations. Mindestens 2 Wochen Report-Only auf Staging und Production laufen lassen.
- [ ] **Vollständige Plugin/Extension-Liste erstellen** — Critical
  Jedes Plugin/jede Extension ist eine potenzielle Quelle für Inline-Scripts, externe Fonts oder APIs. Liste vor CSP-Rollout vollständig dokumentieren.
- [ ] **Staging-Umgebung mit Production-Datenbank testen** — Critical
  CSP muss mit echten Inhalten getestet werden – dynamisch eingebettete externe Bilder, iFrames und Scripts sind oft nur auf Live-Daten sichtbar.
- [ ] **Admin-Bereich und Frontend trennen** — Critical
  Backend ( `/wp-admin/` , `/administrator/` ) braucht deutlich lockerere CSP. Separate Policy für Admin-Pfade definieren.
- [ ] **Unnötige Plugins/Extensions vor CSP-Rollout entfernen** — Important
  Jedes entfernte Plugin ist eine Quelle weniger. CSP-Rollout ist ein guter Anlass für Plugin-Bereinigung.

## 02 WordPress – Core & Themes

WordPress

- [ ] **WordPress Core Inline-Scripts inventarisieren** — Critical · WP
  WordPress Core injiziert selbst Inline-Scripts (wp-settings, jQuery-Migrate, etc.). Diese brauchen `'unsafe-inline'` oder Nonce via `wp_scripts_add_data()` .
- [ ] **Theme Inline-Scripts und Styles prüfen** — Critical · WP
  Viele Premium-Themes (Avada, Divi, Astra) fügen Custom-CSS und Inline-JS ein. Enqueued Styles und Scripts im `wp_head` -Hook inspizieren.
- [ ] **Customizer-CSS in style-src berücksichtigen** — Important · WP
  Änderungen im WP Customizer erzeugen dynamisches Inline-CSS. Ohne `'unsafe-inline'` in `style-src` werden diese Styles blockiert.
- [ ] **Gutenberg-Editor CSP-Anforderungen prüfen** — Important · WP
  Gutenberg-Blocks mit eingebetteten Inhalten (YouTube, Twitter, etc.) benötigen entsprechende `frame-src` und `script-src` Einträge.
- [ ] **wp-json REST API in connect-src** — Important · WP
  Wenn Frontend-Scripts `/wp-json/` via fetch() aufrufen (häufig bei Headless oder Gutenberg-Ajax), muss `'self'` in `connect-src` stehen.
- [ ] **Gravatar in img-src freigeben** — Optional · WP
  Kommentar-Avatare kommen von `*.gravatar.com` und `secure.gravatar.com` . Wird bei kommentarfähigen Sites oft vergessen.

## 03 WordPress – Häufige Plugins

WordPress

- [ ] **Elementor / Divi / WPBakery: Inline-Scripts** — Critical, WP
  Page-Builder injizieren massiv Inline-Scripts und -Styles. Ohne `'unsafe-inline'` funktionieren sie kaum. Für maximale Sicherheit: Nonce-Patch oder Builder wechseln.
- [ ] **Yoast SEO / RankMath: Schema Inline-JSON** — Important, WP
  SEO-Plugins injizieren JSON-LD als Inline-Script-Tag. Benötigt entweder `'unsafe-inline'` oder einen spezifischen SHA-256-Hash (ändert sich je nach Seite).
- [ ] **WP Rocket / W3 Total Cache: Caching-Konflikte** — Critical, WP
  Caching-Plugins cachen Seiten mit Nonce – alle Besucher bekommen denselben Nonce. Nonce-Seiten vom Cache ausschließen oder auf `'unsafe-inline'` mit starker WAF-Absicherung setzen.
- [ ] **Contact Form 7 / Gravity Forms** — Important, WP
  CF7 und GF injizieren Inline-Scripts für Validierung. `form-action 'self'` prüfen. Bei reCAPTCHA: `www.google.com` und `www.gstatic.com` in `script-src` und `frame-src`.
- [ ] **Google Tag Manager via Plugin** — Important, WP
  GTM-Plugins (z. B. GTM4WP) fügen Inline-Scripts ein. Lösung: Server-Side GTM verwenden, oder Nonce-Implementierung im Plugin-Code, oder `'unsafe-inline'` akzeptieren.
- [ ] **WPML / Polylang: Sprach-Switcher** — Optional, WP
  Sprach-Plugins sind meist CSP-kompatibel, können aber Inline-JS für Dropdown-Switcher enthalten. Im Violations-Report prüfen.
- [ ] **Wordfence / Security-Plugins** — Important, WP
  Sicherheits-Plugins setzen teilweise eigene Security-Header – Konflikte mit eigener CSP prüfen. Manche überschreiben den CSP-Header.

## 04 WooCommerce & Payments

WordPress

- [ ] **Stripe.js vollständig freigeben** — Critical, WP
  `js.stripe.com` in `script-src`, `*.stripe.com` in `frame-src` und `connect-src`, `*.stripe.network` in `connect-src`.
- [ ] **PayPal Domains freigeben** — Critical, WP
  `www.paypal.com`, `www.paypalobjects.com` in `script-src` und `frame-src`. PayPal-Button lädt in einem iframe.
- [ ] **Klarna / Mollie / Afterpay** — Important, WP
  Jeder Payment-Anbieter hat eigene Domain-Anforderungen. Anbieter-CSP-Dokumentation vor Integration prüfen.
- [ ] **WooCommerce Checkout vollständig testen** — Critical, WP
  Alle Checkout-Schritte durchlaufen: Warenkorb → Checkout → Payment → Bestätigung. Jeder Schritt kann andere Scripts laden.
- [ ] **WooCommerce-eigene Inline-Scripts** — Important, WP
  WooCommerce übergibt Konfigurationsdaten als `wp_localize_script()` Inline-JSON an JavaScript. Braucht `'unsafe-inline'` oder Nonce-Integration.

## 05 WordPress – CSP technisch setzen

WordPress

- [ ] **CSP via .htaccess (Apache / Plesk)** — Important, WP
  `Header set Content-Security-Policy "..."` in der Root `.htaccess` eintragen. Saubere zentrale Stelle – kann aber von Plugins/Themes überschrieben werden.
- [ ] **CSP via Plugin (HTTP Security Headers / Headers & Footers)** — Important, WP
  Plugins wie "HTTP Security Headers" oder "GD Security Headers" erlauben Header-Management direkt im WP-Backend. Kein Server-Zugriff nötig.
- [ ] **Separate CSP für wp-admin via .htaccess** — Critical, WP
  In der `.htaccess` mit `<FilesMatch>` oder separater `.htaccess` in `/wp-admin/` eine lockerere Policy für das Backend definieren.
- [ ] **Nonce via wp_scripts_add_data() – Advanced** — Optional, WP
  Für maximale Sicherheit: Nonce per `send_nonce_header()` Hook und `wp_scripts_add_data($handle, 'nonce', $nonce)` implementieren. Hoher Aufwand, setzt PHP-Entwicklung voraus.
- [ ] **Konflikte mit anderen Header-setzenden Plugins prüfen** — Important, WP
  Wordfence, iThemes Security, All In One WP Security können eigene CSP-Header setzen und damit den eigenen überschreiben. Immer Endresultat mit Browser-DevTools verifizieren.

## 06 Joomla – Core & Native CSP

Joomla

- [ ] **System – HTTP Headers Plugin aktivieren (Joomla 4+)** — Critical, Joomla
  Das native HTTP-Headers-Plugin unter System → Plugins ermöglicht CSP direkt im Backend ohne Server-Konfiguration. Zuerst im Report-Only Modus testen.
- [ ] **Joomla 3.x: CSP nur via .htaccess oder Nginx** — Critical, Joomla
  Joomla 3 hat kein natives CSP-Plugin. Header müssen am Webserver gesetzt werden. Plugin-Alternative: "HTTP Headers Manager".
- [ ] **Joomla Core Mootools / Bootstrap Inline-Scripts** — Important, Joomla
  Joomla lädt Mootools und Bootstrap teilweise mit Inline-Initialisierungsscripts. Im Report-Only Violations-Log identifizieren welche Direktiven betroffen sind.
- [ ] **Joomla Nonce-Unterstützung (ab 4.1)** — Optional, Joomla
  Joomla 4.1+ unterstützt Nonces über das HTTP Headers Plugin. In Kombination mit dem Plugin konfigurierbar – deutlich sicherer als `'unsafe-inline'`.
- [ ] **/administrator separat absichern** — Critical, Joomla
  Joomla-Backend unter `/administrator/` braucht lockerere CSP. Separate Regel im HTTP-Headers-Plugin oder in der Server-Konfiguration definieren.
- [ ] **Joomla Update-Server in connect-src** — Optional, Joomla
  Wenn Joomla-Updates direkt über das Backend eingespielt werden, muss `update.joomla.org` in `connect-src` des Admin-Bereichs stehen.

## 07 Joomla – Extensions & Templates — Joomla

- [ ] **Template-Framework Inline-Scripts (YOOtheme, T4)** — Critical, Joomla
  Populäre Template-Frameworks wie YOOtheme Pro oder T4 Framework injizieren Konfigurationsdaten als Inline-JSON. In Report-Only prüfen welche Scripts betroffen sind.

- [ ] **VirtueMart / HikaShop Payment-Domains** — Critical, Joomla
  Shop-Extensions laden Payment-Scripts (Stripe, PayPal). Identische Anforderungen wie bei WooCommerce – alle Payment-Domains in `script-src`, `frame-src` und `connect-src`.

- [ ] **JCE Editor / TinyMCE CSP-Anforderungen** — Important, Joomla
  Rich-Text-Editoren nutzen eval() für Plugin-Mechanismen. Im Frontend-Editor-Bereich ggf. lockerere CSP nötig.

- [ ] **RSForm / Chronoforms: reCAPTCHA** — Important, Joomla
  Formular-Extensions mit reCAPTCHA v2/v3 brauchen `www.google.com` und `www.gstatic.com` in `script-src` und `frame-src`.

- [ ] **JoomShopping / andere Shop-Extensions** — Important, Joomla
  Jede Shop-Extension bringt eigene Script-Anforderungen. Extension-Dokumentation auf CSP-Anforderungen prüfen.

## 08 Häufige Drittdienste – beide Systeme — WordPress & Joomla

- [ ] **Google Analytics / GA4** — Important, Beide
  `www.googletagmanager.com` und `www.google-analytics.com` in `script-src`, `connect-src` und `img-src`.

- [ ] **Google Maps Embed** — Important, Beide
  `maps.googleapis.com` in `script-src` und `img-src`. Bei iframe-Embed: `www.google.com` in `frame-src`.

- [ ] **YouTube / Vimeo Embeds** — Important, Beide
  `www.youtube-nocookie.com` in `frame-src`. Bei Vimeo: `player.vimeo.com`.

- [ ] **Cloudflare Rocket Loader** — Important, Beide
  Rocket Loader injiziert Inline-Scripts und lädt Scripts asynchron um. Kann CSP brechen oder umgehen. Im Zweifel deaktivieren wenn CSP aktiviert ist.

- [ ] **Cookie-Consent Tools (Cookiebot, Usercentrics)** — Important, Beide
  Cookie-Banner-Skripte blockieren und entsperren andere Scripts – Interaktion mit CSP komplex. Anbieter-Dokumentation zu CSP-Kompatibilität prüfen.

- [ ] **CDN (BunnyCDN, Cloudflare, KeyCDN)** — Important, Beide
  CDN-Domain für statische Assets in `script-src`, `style-src`, `img-src` und `font-src` freigeben.

- [ ] **Mailchimp / HubSpot Formulare** — Optional, Beide
  Marketing-Formulare als iframe eingebettet: Anbieter-Domain in `frame-src`. Eingebettete JS-Variante: `script-src` und `connect-src` freigeben.

## 09 Testing & Validation — WordPress & Joomla

- [ ] **Alle Seitentypen testen (Blog, Shop, Kontakt, Login)** — Critical
  Jeder Seitentyp kann andere Plugins/Widgets laden. Nicht nur die Homepage testen.
- [ ] **Admin-Bereich vollständig durchtesten** — Critical
  Plugin-Einstellungsseiten, Media Library, Editor – alle häufig genutzten Admin-Bereiche auf CSP-Violations prüfen.
- [ ] **Checkout- und Formular-Flows vollständig** — Critical
  Warenkorb, Checkout, Formular-Submit, Bestätigungs-Seite – jeder Schritt durchklicken. Payment-Scripts oft erst im letzten Schritt geladen.
- [ ] **CSP Evaluator (Google) ausführen** — Critical
  csp-evaluator.withgoogle.com – Policy auf Bypass-Muster prüfen bevor sie live geht.
- [ ] **securityheaders.com Scan** — Important
  Gesamtbewertung aller Security-Header. Ziel: A-Rating. HSTS, X-Content-Type-Options und Referrer-Policy parallel setzen.
- [ ] **Nach jedem Plugin-Update erneut testen** — Important
  Plugin-Updates können neue Inline-Scripts oder externe Domains einführen. Besonders bei Major-Updates von Page-Buildern und SEO-Plugins.

## 10 Wartung & Lifecycle — WordPress & Joomla

- [ ] **CSP-Policy dokumentieren und versionieren** — Important
  Policy in einer separaten Datei oder im Repo dokumentieren – mit Kommentaren warum jede Domain freigegeben ist.
- [ ] **Violations-Monitoring nach Go-Live** — Important
  report-uri aktiv lassen. Neue Violations nach Deployments zeitnah auswerten.
- [ ] **Bei CMS-Major-Update CSP erneut durchgehen** — Critical
  WordPress 6.x → 7.x oder Joomla 4 → 5: Core-Updates können neue Script-Anforderungen mitbringen. Immer auf Staging testen.
- [ ] **Quartalsweise Plugin-Inventur** — Important
  Aktive Plugins regelmäßig überprüfen – deaktivierte/ungenutzte Plugins entfernen reduziert die CSP-Komplexität dauerhaft.
- [ ] **Staging-Umgebung dauerhaft mit Report-Only betreiben** — Optional
  Staging im Report-Only Modus halten – so werden Violations durch neue Plugins oder Updates frühzeitig sichtbar bevor sie Production erreichen.



Alle Angaben ohne Gewähr. Diese Checkliste dient als Orientierungshilfe und ersetzt keine individuelle Sicherheitsberatung. tenbyte.de übernimmt keine Haftung für Schäden, die durch die Verwendung dieser Informationen entstehen. Die Vollständigkeit und Aktualität der Inhalte kann nicht garantiert werden.