Engineer

Content Security Policy

die Browserregeln für Quellen, Code und Seitenaktionen

Was es ist

Eine Content Security Policy ist ein Satz von Regeln, den eine Website dem Browser dafür übergibt, was ihre Seiten ausführen, abrufen und absenden dürfen. Der Browser setzt sie im Auftrag der Website durch. Fast immer wird der Begriff zu CSP abgekürzt. Die Website sendet die Regeln in einem `Content-Security-Policy`-Antwortheader oder schreibt sie in ein `meta`-Element der Seite. Ab diesem Moment werden jedes Skript, Stylesheet, Bild, jede Schrift, jeder Frame und jede Verbindung, die die Regeln nicht erlauben, schon vor dem Laden abgelehnt.

Eine Richtlinie ist eine kurze Zeichenfolge von durch Semikolons getrennten Direktiven. Jede benennt eine Ressourcenart oder ein Verhalten und die akzeptierten Quellen. Zur besseren Lesbarkeit steht hier jede Direktive in einer eigenen Zeile. Übertragen wird ein einzelner Headerwert. Eine strenge Richtlinie sieht so aus:

```
default-src 'self';
script-src 'nonce-q8Zt3VwR1mKx7NpL2cYfJA==' 'strict-dynamic';
object-src 'none';
base-uri 'none';
form-action 'self';
frame-ancestors 'none'
```

Das Vokabular ist klein. `'self'` bedeutet die eigene Origin der Seite. Ein Host wie `https://static.example.com` oder ein Schema wie `https:` erweitert das. `'none'` erlaubt nichts. Eine Nonce ist ein zufälliger Wert, den der Server für eine Antwort erzeugt und sowohl in die Richtlinie als auch an jedes Skriptelement schreibt, für das er sich verbürgt. Ein Hash ist der Prüfwert des exakten Textes

eines Inlineskripts. Nur dieses eine Skript darf damit laufen, kein anderes. Zwei Schlüsselwörter, `'unsafe-inline'` und `'unsafe-eval'`, schalten Schutzmaßnahmen wieder ab. Ihre Namen sind eine angemessene Warnung.

`default-src` ist die Rückfallregel für von der Richtlinie ausgelassene Abrufdirektiven wie `script-src`, `style-src`, `img-src` und `connect-src`. Nur für diese ist es eine Rückfallregel. Direktiven für Verhalten statt für Abrufe, darunter `base-uri`, `form-action` und `frame-ancestors`, bleiben uneingeschränkt, solange sie nicht ausdrücklich aufgeführt sind.

Eine Richtlinie kann auch im Beobachtungsmodus laufen. Als `Content-Security-Policy-Report-Only` gesendet blockiert sie nichts, sondern meldet jeden Verstoß an einen mit `report-to` benannten Endpunkt. Dieser nutzt die Reporting API des Browsers. Die ältere Direktive `report-uri` ist veraltet; stattdessen soll dieser Mechanismus verwendet werden. Der aktuelle Standardtext ist Content Security Policy Level 3, ein W3C-Arbeitsentwurf der Web Application Security Working Group.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist eine Content Security Policy?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Konten, Passwörter und Berechtigungen werden so geordnet, dass alle genau den beruflich erforderlichen Zugriff haben und ihn am Tag ihres Ausscheidens verlieren.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 1.000 RBAC-Regeln für sichere Linux-Server — 1,000 Zugriffsregeln zum Schließen einer Sicherheitslücke

1.000 RBAC-Regeln für sichere Linux-Server Kontakt aufnehmen →

Warum es wichtig ist

Der Angriff, gegen den CSP entwickelt wurde, ist Cross-Site-Scripting. Jemand bringt frei gewählten Text in eine Seite, über ein Kommentarfeld, einen ins HTML zurückkopierten Suchbegriff oder eine kompromittierte Abhängigkeit. Der Browser führt ihn als Skript mit allen Rechten des eigenen Websitecodes aus: Das Skript kann die Seite lesen, als angemeldeter Nutzer handeln und Gefundenes an beliebige Ziele senden. Der erste Schutz ist korrektes Escaping der Ausgabe. Das muss in jedem Template, in jeder Zeile und jedes Mal stimmen. CSP ist der zweite Schutz. Schlüpft ein Escapingfehler durch, muss das eingeschleuste Skript immer noch an der Richtlinie vorbei. Eine Richtlinie, die nur Skripte mit der Nonce dieser Antwort ausführt, weist es ab.

Sie begrenzt auch, was ein tatsächlich ausgeführtes Skript erreichen kann. `connect-src` entscheidet, mit welchen Hosts Skripte sprechen dürfen, und beschränkt so, wohin ein Skript zum Datendiebstahl die gesammelten Daten senden kann. `frame-ancestors` entscheidet, welche Websites die Seite in einem Frame zeigen dürfen. Das ist der übliche Schutz gegen Clickjacking, bei dem eine Seite über eine andere gelegt wird und ein Klick dadurch an einer nie beabsichtigten Stelle landet. `form-action` bestimmt, wohin Formulare senden dürfen. `base-uri` hindert ein eingeschleustes `base` -Element daran, alle relativen Adressen der Seite unbemerkt umzulenken.

Wen CSP schützt, sollte genau benannt werden. Sie läuft im Browser und schützt deshalb den Leser der Seite vor dort vom Angreifer platzierten Inhalten. Den Server schützt sie nicht vor demselben Leser, der den Header im eigenen Browser in Sekunden ändern oder entfernen kann. Eine Richtlinie ist ein Grund, weniger Teilen der Seite zu vertrauen, nie ein Grund, einer Anfrage mehr zu vertrauen.

Die Kosten liegen hauptsächlich in der Bestandsaufnahme. Eine Richtlinie zwingt das Team, jede Origin zu kennen, von der seine Seiten etwas laden, sowie jedes Inlineskript und jeden als Attribut geschriebenen Eventhandler. Eine neue Website lässt sich mit sehr wenig Aufwand vom ersten Commit an auf eine strenge Richtlinie ausrichten. Eine ältere Website mit Inlinehandlern, Tagmanager und mehreren Widgets bezahlt zunächst das Aufräumen. Dieses Aufräumen ist meist das eigentliche Projekt.

Wie es in der Praxis aussieht

Diese Website hält sich an eine solche Richtlinie. Die Sicherheitsseite nennt eine strenge, in der Seite selbst deklarierte Content Security Policy mit `form-action 'none'` , weil die Website nirgendwo Formulare absendet. Jede HTML-Seite legt `script-src 'none'` fest: Es laufen weder ausführbares JavaScript noch ein Service Worker. Die Website stellt keine Anfragen an Dritte. Alles für eine Seite Nötige kommt vom eigenen Host, deshalb muss sehr wenig erlaubt werden.

Eine Anwendung mit angemeldeten Nutzern braucht mehr als eine kurze Freigabeliste. Auf einer Plattform mit beruflichen und organisatorischen Daten setzt die Next.js-Proxy-Middleware eine Content-Security-Policy mit einer Nonce pro Anfrage und `strict-dynamic` , zusammen mit HSTS und SameSite-Cookies. Eine frische Nonce bei jeder Antwort bedeutet, dass nur die vom Server in dieser Antwort markierten Skriptelemente laufen. Mit `strict-dynamic` in der Richtlinie

darf ein durch die Nonce vertrauenswürdiges Skript selbst weitere Skripte laden. Die Richtlinie muss deshalb nicht jeden Host nennen, von dem ein Bundle seine Chunks abruft. Diese Plattform geht von einem feindlichen Client aus. Die Berechtigungen jeder Anfrage werden daher erneut auf dem Server geprüft. Die Sperren im Frontend dienen allein dem Nutzungserlebnis. CSP ist dort eine von mehreren Schichten und schützt Nutzer vor eingeschleustem Code. Die Autorisierung wird anderswo entschieden.

Wo es schiefgeht

Die Richtlinie wird aktiviert, und die halbe Website funktioniert nicht mehr. Eine sofort durchgesetzte strenge Richtlinie blockiert das vergessene Inlineskript und den vor zwei Jahren hinzugefügten Trackingtag. Liefern Sie sie zunächst als `Content-Security-Policy-Report-Only` aus. Lesen Sie die Berichte, bis sie sich stabilisieren, erlauben oder entfernen Sie jede darin genannte Quelle und setzen Sie die Richtlinie erst dann durch. Der Report-Only-Header muss ein echter Header sein. Die Spezifikation unterstützt ihn nicht in einem `meta`-Element.

Die Freigabeliste vertraut einem ganzen Host. `script-src https://cdn.example.com` erlaubt jede Datei dieses Hosts, auch alte Bibliotheksversionen und Endpunkte, die Eingaben des Aufrufers als Skript zurückgeben. Die Spezifikation selbst bezeichnet es als Herausforderung, eine gegen Cross-Site-Scripting wirksame Richtlinie einzuführen. Sie empfiehlt eine sogenannte Strict CSP: `script-src` aus Nonces oder Hashes mit `'strict-dynamic'`, dazu `base-uri` mit dem Wert `'self'` oder `'none'`.

`'unsafe-inline'` **soll jedes Inlineskript erlauben.** Sobald eine Nonce oder ein Hash in derselben Quellenliste steht, ignorieren Browser, die diese verstehen, `'unsafe-inline'`. So kann eine Richtlinie `'unsafe-inline'` als Rückfallregel für Browser behalten, die für Nonces zu alt sind, während aktuelle Browser die strenge Liste anwenden. Das überrascht Teams, die einem Skript eine Nonce geben und dann feststellen, dass die übrigen Inlineskripte blockiert werden.

Die Nonce bleibt nicht geheim. Die Spezifikation verlangt bei jeder gesendeten Richtlinie einen neuen Wert und empfiehlt mindestens 128 Bit aus einem kryptografisch sicheren Generator. Eine im Template fest eingetragene Nonce oder eine ganze Seite, die mit ihrer Nonce aus einem gemeinsamen Cache an alle ausgeliefert wird, gibt dem Angreifer den Wert, mit dem er sein eigenes Skript markieren kann.

Eine Richtlinie im `meta`-Element leistet weniger als dieselbe im Header. Der Browser ignoriert auf diesem Weg `frame-ancestors`, `report-uri` und `sandbox`. Die Richtlinie gilt nur für das, was im Dokument nach dem Element kommt. Alles davor im Dokumentkopf wird ungeprüft geladen. Eine Website ohne Einfluss auf ihre Antwortheader erhält dennoch echten Schutz durch eine möglichst früh im Dokumentkopf platzierte `meta`-Richtlinie. Schutz vor Einbettung in Frames braucht aber den Header.

Eine zweite Richtlinie verschärft statt zu lockern. Eine Antwort kann mehrere Richtlinien enthalten, von Framework, Proxy und CDN. Eine Ressource muss alle Richtlinien erfüllen. Eine zusätzliche Richtlinie kann nur weiter einschränken. Ein nachgelagert ergänzter großzügiger Header zur Reparatur einer blockierten Anfrage hilft deshalb nicht. Die Korrektur muss in die blockierende Richtlinie.

Weiterführende Lektüre

- Content Security Policy Level 3 — der Standard selbst mit Strict-CSP-Empfehlung und einem Abschnitt über den Diebstahl von Nonces.
- HTML-Standard: das meta-Element — die Anweisung zur Anwendung einer in der Seite deklarierten Richtlinie, ausgelassene Direktiven und der Grund, warum zuvor abgerufene Ressourcen ihr entgehen können.
- Reporting API — der Endpunktmechanismus hinter `report-to`, um Verstöße vor der Durchsetzung auszuwerten.
- Trusted Types — der durch eine Direktive `require-trusted-types-for` eingeschaltete Begleitstandard gegen durch das DOM eingeschleuste Skripte.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist eine Content Security Policy?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/csp/