Engineer

Content Security Policy

règles sur ce qu'une page peut exécuter, charger et envoyer

Ce que c’est

Content Security Policy est un ensemble de règles qu’un site transmet au navigateur sur ce que ses pages peuvent exécuter, récupérer et soumettre ; le navigateur les applique pour le compte du site. Son nom est presque toujours abrégé en CSP. Le site les envoie dans un en-tête de réponse `Content-Security-Policy` ou les écrit dans la page dans un élément `meta` . Dès lors, tout script, feuille de style, image, police, cadre ou connexion que les règles n’autorisent pas est refusé avant son chargement.

Une politique est une courte chaîne de directives séparées par des points-virgules. Chaque directive nomme une catégorie de ressource ou de comportement et les sources acceptées. Présentée sur plusieurs lignes pour être lisible, même si elle ne forme qu’une valeur d’en-tête sur le réseau, une politique stricte ressemble à ceci :

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

Le vocabulaire est restreint. `'self'` désigne l’origine de la page. Un hôte comme `https://static.example.com` ou un schéma comme `https:` élargit cette portée. `'none'` n’autorise rien. Un nonce est une valeur aléatoire que le serveur génère pour une réponse et inscrit à la fois dans la politique et sur chaque élément script qu’il approuve. Un hachage est l’empreinte du texte exact d’un script en ligne, de sorte que ce script seul puisse s’exécuter. Deux mots-clés, `'unsafe-inline'` et `'unsafe-eval'` , désactivent à nouveau des protections, et leurs noms constituent un avertissement justifié.

`default-src` sert de repli aux directives de récupération omises, comme `script-src`, `style-src`, `img-src` et `connect-src`. Il ne sert de repli qu'aux directives de récupération. Les directives qui régissent un comportement plutôt qu'une récupération, notamment `base-uri`, `form-action` et `frame-ancestors`, restent sans restriction si elles ne sont pas écrites explicitement.

Une politique peut aussi fonctionner en observation. Envoyée comme `Content-Security-Policy-Report-Only`, elle ne bloque rien et signale chaque violation à un point de collecte désigné par `report-to`, via la Reporting API du navigateur. L'ancienne directive `report-uri` est dépréciée à son profit. Le texte actuel de la norme est Content Security Policy Level 3, un document de travail (Working Draft) du W3C émanant du Web Application Security Working Group.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'une Content Security Policy ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des comptes, mots de passe et droits attribués selon les besoins exacts de chaque poste, puis retirés le jour du départ.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 1 000 règles RBAC pour sécuriser Linux — 1,000 règles d'accès écrites pour combler une faille de sécurité

1 000 règles RBAC pour sécuriser Linux Contactez-nous →

Pourquoi c'est important

CSP a été conçue contre le cross-site scripting. Quelqu'un introduit le texte de son choix dans une page, par un commentaire, un terme de recherche répété dans le HTML ou une dépendance compromise ; le navigateur l'exécute comme script avec tous les droits du code du site : lire la page, agir comme l'utilisateur connecté et envoyer ce qu'il trouve n'importe où. La première défense est l'échappement correct de la sortie, qui doit être juste dans chaque template, à chaque ligne et chaque fois. CSP est la seconde défense. Lorsqu'une erreur d'échappement passe tout de même, le script injecté doit encore franchir la politique ; une politique qui n'exécute que les scripts portant le nonce de cette réponse le refuse.

Elle limite aussi ce qu'un script effectivement exécuté peut atteindre. `connect-src` décide avec quels hôtes il peut communiquer et restreint les destinations vers lesquelles un script de vol de données peut envoyer ce qu'il collecte. `frame-ancestors` décide quels sites peuvent afficher la page dans un cadre ; c'est la défense standard contre le détournement de clic, où une page en recouvre une

autre pour que le clic atteigne un endroit non voulu par l'utilisateur. `form-action` décide où les formulaires peuvent envoyer leurs données, et `base-uri` empêche un élément `base` injecté de rediriger discrètement toutes les adresses relatives de la page.

Il faut préciser qui CSP protège. Cette politique s'applique dans le navigateur et protège donc la personne qui lit la page contre le contenu qu'un attaquant y a placé. Elle ne protège en rien le serveur contre cette même personne, qui peut modifier ou enlever l'en-tête dans son navigateur en quelques secondes. Une politique justifie de moins faire confiance à la page, jamais de davantage faire confiance à la requête.

Le coût est surtout celui de l'inventaire. Une politique oblige l'équipe à connaître chaque origine à partir de laquelle ses pages chargent des ressources, chaque script en ligne et chaque gestionnaire d'événement écrit dans un attribut. Un nouveau site peut suivre une politique stricte dès son premier commit pour très peu de travail. Un site ancien avec des gestionnaires en ligne, un gestionnaire de balises et plusieurs widgets doit d'abord faire le nettoyage, qui constitue généralement le véritable projet.

À quoi cela ressemble en pratique

Ce site s'en impose une. La page de sécurité énumère une Content Security Policy stricte, déclarée dans la page, avec `form-action 'none'`, car le site n'envoie aucun formulaire nulle part. Chaque page HTML déclare `script-src 'none'` : aucun JavaScript exécutable ni service worker ne tourne. Le site ne fait aucune requête tierce ; tout ce dont une page a besoin vient de son propre hôte, ce qui laisse très peu de choses à autoriser.

Une application avec des utilisateurs connectés demande plus qu'une courte liste d'autorisation. Sur une plateforme contenant des données professionnelles et organisationnelles, le middleware proxy Next.js définit une Content-Security-Policy avec un nonce par requête et `strict-dynamic`, aux côtés de HSTS et de cookies SameSite. Un nonce neuf à chaque réponse signifie que seuls les éléments script marqués par le serveur pour cette réponse pourront s'exécuter. Avec `strict-dynamic`, un script approuvé par le nonce peut charger lui-même d'autres scripts ; la politique n'a donc pas à nommer chaque hôte d'où un bundle récupère ses fragments. Cette plateforme suppose le client hostile. Les droits de chaque requête sont donc revérifiés sur le serveur, et les contrôles du

frontend ne comptent que pour l'expérience utilisateur. CSP y est une couche parmi plusieurs, qui protège les utilisateurs du code injecté ; l'autorisation se décide ailleurs.

Où les problèmes surviennent

La politique est activée et la moitié du site cesse de fonctionner. Une politique stricte appliquée directement bloque le script en ligne oublié et la balise de suivi ajoutée deux ans plus tôt. Livrez d'abord `Content-Security-Policy-Report-Only`, lisez les rapports jusqu'à stabilisation, autorisez ou retirez chaque source nommée, puis seulement appliquez les blocages. L'en-tête du mode observation doit être un véritable en-tête HTTP : la spécification ne le prend pas en charge dans un élément `meta`.

La liste d'autorisation fait confiance à un hôte entier. `script-src https://cdn.example.com` autorise tous ses fichiers, y compris les anciennes versions de bibliothèques et les points d'accès qui renvoient une entrée de l'appelant sous forme de script. La spécification dit elle-même qu'une politique efficace contre le cross-site scripting est difficile à déployer et recommande ce qu'elle appelle une Strict CSP : `script-src` fondé sur des nonces ou des hachages avec `'strict-dynamic'`, ainsi que `base-uri` fixé à `'self'` ou `'none'`.

On suppose que `'unsafe-inline'` autorise tous les scripts en ligne. Dès qu'un nonce ou un hachage figure dans la même liste de sources, les navigateurs qui les comprennent ignorent `'unsafe-inline'`. Cela permet de conserver `'unsafe-inline'` en repli pour les navigateurs trop anciens pour connaître les nonces, tandis que les navigateurs actuels appliquent la liste stricte. Une équipe qui ajoute un nonce à un seul script peut donc être surprise de voir les autres scripts en ligne bloqués.

Le nonce cesse d'être secret. La spécification exige une nouvelle valeur à chaque envoi de politique et recommande au moins 128 bits issus d'un générateur cryptographiquement sûr. Un nonce codé en dur dans un template, ou une page entière avec son nonce conservée dans un cache partagé et servie à tous, donne à l'attaquant la valeur nécessaire pour marquer son propre script.

Une politique dans un élément `meta` en fait moins que la même politique en en-tête. Le navigateur ignore `frame-ancestors`, `report-uri` et `sandbox` lorsqu'ils arrivent ainsi, et la politique ne régit que ce qui suit cet élément dans le document ; tout ce qui le précède dans la section head se charge sans contrôle.

Un site qui ne maîtrise pas ses en-têtes bénéficie tout de même d'une véritable protection grâce à une politique `meta`, placée aussi tôt que possible dans la section head, mais la protection contre l'intégration dans des cadres exige l'en-tête.

Une seconde politique durcit au lieu d'assouplir. Une réponse peut porter plusieurs politiques, venant d'un framework, d'un proxy et d'un CDN ; une ressource doit toutes les satisfaire. Ajouter une politique ne peut que restreindre ; un en-tête permissif ajouté en aval pour réparer une requête bloquée ne change donc rien. Il faut corriger la politique à l'origine du blocage.

Pour aller plus loin

- Content Security Policy Level 3 — la norme elle-même, avec la recommandation Strict CSP et une section sur le vol des nonces.
- Norme HTML : l'élément meta — le pragma appliquant une politique déclarée dans la page, les directives qu'il ignore et les raisons pour lesquelles des ressources récupérées avant lui peuvent échapper à la politique.
- Reporting API — le mécanisme de point de collecte derrière `report-to`, pour lire les violations avant d'appliquer les blocages.
- Trusted Types — la norme complémentaire activée par `require-trusted-types-for`, visant les scripts injectés par le DOM.



Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce qu'une Content Security Policy ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/csp/