**Engineer**

# política de seguridad de contenido

reglas que limitan código, orígenes y acciones de una página

## Qué es

Content Security Policy, o política de seguridad de contenido, es un conjunto de reglas que un sitio web entrega al navegador sobre lo que sus páginas pueden ejecutar, obtener y enviar, y que el navegador hace cumplir en nombre del sitio. Casi siempre se abrevia como CSP. El sitio envía las reglas en una cabecera de respuesta `Content-Security-Policy` o las escribe en un elemento `meta` de la página, y desde ese momento se rechaza antes de cargarlo cualquier script, hoja de estilos, imagen, fuente, marco o conexión que las reglas no permitan.

Una política es una cadena breve de directivas separadas por puntos y coma. Cada directiva identifica un tipo de recurso o comportamiento y las fuentes que acepta. Una política estricta tiene este aspecto, con una directiva por línea para facilitar la lectura, aunque en la transmisión sea un único valor de cabecera:

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

El vocabulario es reducido. `'self'` significa el origen de la propia página. Un host como `https://static.example.com` o un esquema como `https:` amplían ese alcance. `'none'` no permite nada. Un nonce es un valor aleatorio que el servidor genera para una respuesta y escribe tanto en la política como en cada elemento de script que avala. Un hash es el resumen criptográfico del texto exacto de un script en línea, para que pueda ejecutarse ese script y ningún otro. Dos palabras clave, `'unsafe-inline'` y `'unsafe-eval'`, vuelven a desactivar protecciones, y sus nombres son una advertencia acertada.

`default-src` es la regla de respaldo para las directivas de obtención que una política omite, como `script-src` , `style-src` , `img-src` y `connect-src` . Solo es un respaldo para ellas. Las directivas que gobiernan el comportamiento en lugar de la obtención de recursos, entre ellas `base-uri` , `form-action` y `frame-ancestors` , quedan sin restricciones salvo que se escriban expresamente.

Una política también puede funcionar en modo de observación. Si se envía como `Content-Security-Policy-Report-Only` , no bloquea nada y, en su lugar, notifica cada infracción a un endpoint identificado con `report-to` , que utiliza la Reporting API del navegador. La directiva anterior, `report-uri` , está obsoleta en favor de esta. El texto actual del estándar es Content Security Policy Level 3, un borrador de trabajo del W3C elaborado por Web Application Security Working Group.

Utilice el concepto en una decisión

Aplique «¿Qué es una política de seguridad de contenido?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Cuentas, contraseñas y permisos organizados para que cada persona tenga exactamente el acceso que necesita y lo pierda el mismo día que deja la empresa.

**Explore nuestros servicios** Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. 1.000 reglas RBAC para servidores Linux — 1,000 reglas de acceso escritas para cerrar una brecha de seguridad

**1.000 reglas RBAC para servidores Linux** Póngase en contacto →

## Por qué importa

El ataque contra el que se diseñó CSP es cross-site scripting. Alguien consigue introducir texto de su elección en una página, mediante un campo de comentarios, un término de búsqueda repetido en el HTML o una dependencia comprometida, y el navegador lo ejecuta como script con toda la autoridad del código del propio sitio: puede leer la página, actuar como el usuario que ha iniciado sesión y enviar lo que encuentre a cualquier lugar. La primera defensa es escapar correctamente la salida, y eso tiene que hacerse bien en cada plantilla, en cada línea y en cada ocasión. CSP es la segunda defensa. Cuando se cuela un error de escape, el script inyectado todavía tiene que superar la política, y una que solo ejecuta scripts con el nonce de esa respuesta lo rechaza.

También limita el alcance de un script que sí se ejecuta. `connect-src` decide con qué hosts puede comunicarse, lo que reduce los destinos a los que un script de robo de datos puede enviar lo que recopila. `frame-ancestors` decide qué sitios pueden mostrar la página dentro de un marco, la defensa estándar contra el

clickjacking, donde una página se superpone a otra para que un clic llegue a un lugar que el usuario nunca pretendió pulsar. `form-action` decide adónde pueden enviar datos los formularios, y `base-uri` impide que un elemento `base` inyectado redirija silenciosamente todas las direcciones relativas de la página.

Conviene precisar a quién protege CSP. Se ejecuta en el navegador, así que protege a la persona que lee la página del contenido que un atacante haya introducido en ella. No protege al servidor frente a esa misma persona, que puede editar o eliminar la cabecera en su propio navegador en segundos. Una política es una razón para confiar menos en la página, nunca para confiar más en la petición.

El coste consiste sobre todo en hacer inventario. Una política obliga al equipo a conocer todos los orígenes desde los que sus páginas cargan recursos, todos los scripts en línea y todos los controladores de eventos escritos como atributos. Un sitio nuevo puede construirse con una política estricta desde el primer commit por muy poco. Uno antiguo con controladores en línea, un gestor de etiquetas y varios widgets paga primero la limpieza, que suele ser el verdadero proyecto.

## Cómo se aplica en la práctica

Este sitio se impone una política de este tipo. La página de seguridad enumera una Content Security Policy estricta, declarada en la propia página, con `form-action 'none'` , porque el sitio no envía formularios a ninguna parte. Todas las páginas HTML declaran `script-src 'none'` : no se ejecuta JavaScript ni ningún service worker. El sitio no hace peticiones a terceros; todo lo que necesita una página procede de su propio host, así que hay muy poco que permitir.

Una aplicación con usuarios que han iniciado sesión necesita algo más que una lista breve de elementos permitidos. En una plataforma con datos profesionales y de organizaciones, el middleware proxy de Next.js establece una Content-Security-Policy con un nonce por petición y `strict-dynamic` , junto con HSTS y cookies SameSite. Un nonce nuevo en cada respuesta significa que solo se ejecutarán los elementos de script que el servidor haya marcado en esa respuesta. Con `strict-dynamic` en la política, un script al que el nonce otorga confianza puede cargar por sí mismo más scripts, de modo que la política no necesita nombrar todos los hosts desde los que un paquete obtiene sus fragmentos. Esa plataforma parte del supuesto de que el cliente es hostil. Por eso el servidor vuelve a comprobar los permisos de cada petición, y los controles

del frontend solo cuentan para la experiencia de usuario. CSP es allí una de varias capas, protege a los usuarios del código inyectado y deja la decisión de autorización en otro lugar.

## Dónde surgen los problemas

**La política entra en vigor y medio sitio deja de funcionar.** Aplicar directamente una política estricta bloquea el script en línea que nadie recordaba y la etiqueta de seguimiento que alguien añadió hace dos años. Envíela primero como `Content-Security-Policy-Report-Only`, lea los informes hasta que se estabilicen, permita o elimine cada fuente que señalen y solo entonces aplíquela. La cabecera de solo informe tiene que ser una cabecera real: la especificación no la admite en un elemento `meta`.

**La lista de elementos permitidos confía en un host completo.** `script-src https://cdn.example.com` permite todos los archivos de ese host, incluidas versiones antiguas de bibliotecas y endpoints que devuelven la entrada de quien llama como script. La propia especificación dice que desplegar una política eficaz contra cross-site scripting es un reto y recomienda lo que denomina Strict CSP: `script-src` construido con nonces o hashes y `'strict-dynamic'`, más `base-uri` con `'self'` o `'none'`.

**Se supone que `'unsafe-inline'` permite todos los scripts en línea.** Cuando aparece un nonce o un hash en la misma lista de fuentes, los navegadores que los entienden ignoran `'unsafe-inline'`. La regla permite conservar `'unsafe-inline'` como respaldo para navegadores demasiado antiguos para conocer los nonces, mientras los actuales aplican la lista estricta, y sorprende a un equipo que añade un nonce a un script y ve bloqueados los demás scripts en línea.

**El nonce deja de ser secreto.** La especificación exige un valor nuevo cada vez que se envía una política y recomienda al menos 128 bits procedentes de un generador criptográficamente seguro. Un nonce fijado en una plantilla, o una página completa con su nonce guardada en una caché compartida y servida a todo el mundo, da al atacante el valor que necesita para marcar su propio script.

**Una política en un elemento `meta` hace menos que la misma política en una cabecera.** El navegador ignora `frame-ancestors`, `report-uri` y `sandbox` cuando llegan de esa forma, y la política solo gobierna lo que aparece después del elemento en el documento, así que todo lo que esté antes en la cabecera HTML se carga sin comprobar. Un sitio que no puede decidir sus cabeceras de respuesta

sigue obteniendo protección real de una política `meta`, situada lo antes posible en la cabecera HTML, pero la protección contra la inclusión en marcos necesita la cabecera de respuesta.

**Una segunda política restringe en vez de flexibilizar.** Una respuesta puede llevar varias políticas, procedentes de un framework, un proxy y una CDN, y un recurso tiene que cumplirlas todas. Añadir una política solo puede restringir, así que una cabecera permisiva añadida después para arreglar una petición bloqueada no hace nada, y hay que corregir la política que la bloqueó.

## Lecturas adicionales

- Content Security Policy Level 3 — el propio estándar, con la recomendación Strict CSP y una sección sobre el robo de nonces.
- Estándar HTML: el elemento meta — la directiva pragma que aplica una política declarada en la página, las directivas que descarta y por qué pueden eludirla los recursos obtenidos antes.
- Reporting API — el mecanismo de endpoints que utiliza `report-to` para leer las infracciones antes de aplicar la política.
- Trusted Types — el estándar complementario que activa una directiva `require-trusted-types-for`, dirigido al script inyectado a través del DOM.



Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

**Vea cuánto cuesta** Póngase en contacto →

# Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es una política de seguridad de contenido?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

**Explore nuestros servicios**

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

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

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

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/csp/