**Engineer**

# Content Security Policy

перелік джерел, з яких сторінці можна вантажити

## Що це таке

Content Security Policy — це набір правил, які вебсайт передає браузеру: що його сторінкам дозволено виконувати, завантажувати й надсилати. Далі браузер сам стежить за цими правилами від імені сайту. Назву майже завжди скорочують до CSP. Сайт надсилає правила в заголовку відповіді `Content-Security-Policy` або вписує їх просто в сторінку через елемент `meta`, і відтоді кожен скрипт, таблицю стилів, зображення, шрифт, фрейм чи з'єднання, яких правила не дозволяють, браузер відхиляє ще до завантаження.

Політика — це короткий рядок директив, розділених крапкою з комою. Кожна директива називає один тип ресурсу чи поведінки та джерела, які вона приймає. Тут для зручності кожна директива стоїть в окремому рядку, хоча мережею вона йде як одне значення заголовка:

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

Словник невеликий. `'self'` означає власне походження сторінки. Хост на кшталт `https://static.example.com` або схема на кшталт `https:` додають інші джерела. `'none'` не дозволяє нічого. Nonce — це випадкове значення, яке сервер генерує для однієї конкретної відповіді й записує і в політику, і на кожен елемент `script`, за який ручається. Хеш — це дайджест точного тексту вбудованого скрипту, тож виконатися може саме цей скрипт, а інший уже ні. Два ключові слова, `'unsafe-inline'` та `'unsafe-eval'`, знову вимикають захист, і їхні назви — чесне попередження.

`default-src` — запасний варіант для директив завантаження, які політика пропустила: `script-src`, `style-src`, `img-src`, `connect-src` та інших. Тільки для них. Директиви, що керують поведінкою, а не завантаженням, серед них `base-uri`, `form-action` і `frame-ancestors`, лишаються без обмежень, доки їх не прописати явно.

Політика може працювати й у режимі спостереження. Надіслана як `Content-Security-Policy-Report-Only`, вона нічого не блокує, а повідомляє про кожне порушення на кінцеву точку, вказану через `report-to`, яка спирається на Reporting API браузера. Старіша директива `report-uri` застаріла на її користь. Чинний текст стандарту — Content Security Policy Level 3, робочий проєкт W3C від робочої групи Web Application Security.

Застосуйте поняття до рішення

Застосуйте «Що таке Content Security Policy?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Облікові записи, паролі й права налаштовані так, щоб доступ відповідав роботі, а звільнені втрачали його того ж дня.

**Переглянути наші послуги** Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Написала 1 000 правил RBAC для Linux-серверів — 1,000 правил доступу написано, щоб усунути вразливість

**Написала 1 000 правил RBAC для Linux-серверів** Зв'язатися з нами →

## Чому це важливо

CSP створювали проти міжсайтового скриптингу, який зазвичай називають XSS. Зловмисник вставляє на сторінку довільний текст через поле коментаря, пошуковий запит, який сторінка повторює у своєму HTML, або скомпрометовану залежність, і браузер виконує цей текст як скрипт з усіма повноваженнями власного коду сайту. Такий скрипт може читати сторінку, діяти від імені користувача, що увійшов, і надсилати знайдене куди завгодно. Перший захист — правильне екранування виводу, і воно мусить бути правильним у кожному шаблоні, в кожному рядку, щоразу. CSP — другий захист. Коли помилка екранування все ж проскакує, підкинутому скрипту ще треба пройти крізь політику, а політика, яка виконує лише скрипти з nonce саме цієї відповіді, його не пропустить.

Вона також обмежує, куди може дотягнутися скрипт, який таки виконався. `connect-src` визначає, з якими хостами скрипт може спілкуватися, і цим звужує, куди скімер може відправити зібране. `frame-ancestors` визначає, які сайти можуть показувати сторінку у фреймі. Це стандартний захист від

клікджекінгу, коли одну сторінку накладають на іншу, щоб клік потрапив туди, куди користувач і не збирався натискати. `form-action` визначає, куди можуть надсилатися форми, а `base-uri` не дає підкинутому елементу `base` непомітно перенаправити кожну відносну адресу на сторінці.

Варто точно розуміти, кого саме захищає CSP. Вона працює в браузері, отже, оберігає читача сторінки від вмісту, який туди підклав зловмисник. Серверу від цього ж читача вона нічим не допомагає, бо той може змінити або прибрати заголовок у власному браузері за лічені секунди. Політика — привід довіряти сторінці менше, але ніколи не привід більше довіряти запиту.

Головна ціна — інвентаризація. Політика змушує команду знати кожне походження, з якого вантажаться її сторінки, кожен вбудований скрипт і кожен обробник подій, записаний як атрибут. Новий сайт можна з першого коміту будувати під сувору політику майже задарма. Старіший сайт із вбудованими обробниками, менеджером тегів і кількома віджетами спершу платить за прибирання, і саме це прибирання зазвичай і є справжнім проєктом.

## Як це виглядає на практиці

Цей сайт і сам живе за такою політикою. Сторінка безпеки називає сувору Content Security Policy, оголошену в самій сторінці, з `form-action 'none'`, бо сайт нікуди не надсилає форм. Кожна HTML-сторінка вказує `script-src 'none'`: ні JavaScript, ні service worker не виконуються. Сайт не робить запитів до третіх сторін; усе, що потрібно сторінці, надходить із власного хоста сайту, тож дозволяти майже нічого.

Застосунку, де користувачі входять в обліковий запис, короткого переліку дозволених джерел замало. На платформі з професійними та організаційними даними проксі-мідлвар Next.js встановлює Content-Security-Policy з nonce для кожного запиту та `strict-dynamic`, а також HSTS і SameSite cookies. Свіжий nonce у кожній відповіді означає, що виконаються лише ті елементи `script`, які сервер позначив саме в цій відповіді. Із `strict-dynamic` у політиці скрипт, якому довіряє nonce, може сам підвантажувати інші скрипти, тож політиці не треба перелічувати кожен хост, з якого бандл тягне свої частини. Та сама сторінка прямо каже, де закінчується роль браузера. Її робоче припущення таке: клієнт вороже налаштований, тому права доступу сервер перевіряє повторно на кожен запит, а фронтендні обмеження

розглядаються лише як UX. CSP там — один із кількох рівнів. Вона захищає користувачів від підкинутого коду, а те, кому що дозволено, вирішується деінде.

## Де виникають проблеми

**Політику вмикають, і пів сайту перестає працювати.** Сувора політика, одразу відправлена в режим примусу, блокує вбудований скрипт, про який усі забули, і тег відстеження, який хтось додав два роки тому. Спершу надсилайте її як `Content-Security-Policy-Report-Only`, читайте звіти, доки вони не вщухнуть, дозвольте або приберіть кожне джерело, яке вони називають, і лише тоді вмикайте примус. Заголовок report-only має бути справжнім заголовком, бо специфікація не підтримує його в елементі `meta`.

**Перелік довіряє цілому хосту.** `script-src https://cdn.example.com` дозволяє кожен файл на цьому хості, зокрема старі версії бібліотек і кінцеві точки, які повертають введене користувачем як скрипт. Специфікація сама визнає, що розгорнути політику, яка справді працює проти міжсайтового скриптингу, складно, і радить те, що називає Strict CSP: `script-src` з nonce або хешів разом із `'strict-dynamic'`, плюс `base-uri` зі значенням `'self'` або `'none'`.

**Вважають, що `'unsafe-inline'` дозволяє будь-який вбудований скрипт.** Щойно в тому самому переліку джерел з'являється nonce чи хеш, браузери, які їх розуміють, ігнорують `'unsafe-inline'`. Це правило дає політиці змогу лишити `'unsafe-inline'` як запасний варіант для браузерів, надто старих для nonce, тоді як сучасні застосовують суворий перелік. І воно ж дивує команду, яка додала nonce одному скрипту й побачила, що решта вбудованих скриптів тепер заблокована.

**Nonce перестає бути секретом.** Специфікація вимагає нового значення щоразу, коли надсилається політика, і просить щонайменше 128 біт від криптографічно стійкого генератора. Nonce, жорстко вписаний у шаблон, або ціла сторінка, яка разом зі своїм nonce лежить у спільному кеші й віддається всім, дає зловмиснику саме те значення, яке потрібне, щоб позначити власний скрипт.

**Політика в елементі `meta` робить менше, ніж та сама політика в заголовку.** Браузер ігнорує `frame-ancestors`, `report-uri` і `sandbox`, коли вони приходять таким шляхом, а політика діє лише на те, що стоїть після елемента в документі, тож усе вище нього в head завантажується без перевірки. Сайт,

який не має впливу на заголовки відповіді, все одно отримує реальний захист від політики в `meta`, розміщеної в head якомога раніше, але захист від показу сторінки в чужих фреймах потребує заголовка.

**Друга політика затягує, а не послаблює.** Відповідь може нести кілька політик, від фреймворку, проксі та CDN, і ресурс мусить пройти кожну з них. Додана політика здатна лише обмежувати, тож поблажливий заголовок, доданий далі в ланцюжку, щоб розблокувати запит, нічого не змінить. Виправлення треба вносити в ту політику, яка заблокувала.

## Що почитати далі

- Content Security Policy Level 3 — сам стандарт, із рекомендацією Strict CSP і розділами про те, як nonce перехоплюють і викрадають.
- HTML Standard: елемент meta — прагма, що застосовує політику, оголошену в сторінці, директиви, які вона відкидає, і чому ресурси, завантажені до неї, можуть проскочити.
- Reporting API — механізм кінцевих точок, на який спирається `report-to`, щоб читати порушення ще до ввімкнення примусу.
- Trusted Types — супутній стандарт, який вмикає директива `require-trusted-types-for`, спрямований проти скриптів, підкинутих через DOM.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

**Дивитися, скільки це коштує** Зв'язатися з нами →

# Зробіть термінологію практичною

Запитайте, як «Що таке Content Security Policy?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

**Переглянути наші послуги**

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

**Дбайливо зроблено з любов'ю — D.**

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

**Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.**

**Надруковано з https://engineer.company/uk/glossary/csp/**