Engineer

Content Security Policy

en liste over, hvor en side må hente ting fra

Hvad det er

Content Security Policy er et sæt regler, som et website giver browseren om, hvad dets sider må køre, hente og sende, og som browseren derefter håndhæver på sitets vegne. Navnet forkortes næsten altid til CSP. Sitet sender reglerne i en `Content-Security-Policy` -header i svaret eller skriver dem ind i selve siden i et `meta` -element, og fra det øjeblik bliver hvert script, stilark, billede, hver skrifttype, indlejret ramme og netværksforbindelse, som reglerne ikke tillader, afvist, før det overhovedet indlæses.

En politik er en kort tekststreng af direktiver adskilt af semikolon. Hvert direktiv navngiver én slags ressource eller adfærd og de kilder, det godtager. Her står den med ét direktiv pr. linje, så den er til at læse, selv om den over nettet er én samlet headerværdi:

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

Ordforrådet er lille. `'self'` betyder sidens egen oprindelse. En vært som `https://static.example.com` eller et skema som `https:` lukker flere kilder ind. `'none'` tillader ingenting. En nonce er en tilfældig værdi, som serveren laver til ét bestemt svar og skriver både i politikken og på hvert script-element, den står inde for. En hash er fingeraftrykket af et inline-scripts nøjagtige tekst, så netop det script og intet andet må køre. To nøgleord, `'unsafe-inline'` og `'unsafe-eval'` , slår beskyttelsen fra igen, og navnene er en ærlig advarsel.

`default-src` er reserven for de hentedirektiver, en politik udelader, fx `script-src`, `style-src`, `img-src` og `connect-src`. Den er reserve for dem og kun dem. Direktiver, der styrer adfærd frem for hentning, blandt andre `base-uri`, `form-action` og `frame-ancestors`, er helt åbne, indtil de bliver skrevet ud.

En politik kan også køre i en iagttagende tilstand. Sendt som `Content-Security-Policy-Report-Only` blokerer den ingenting, men rapporterer hver overtrædelse til et endepunkt, der angives med `report-to` og bygger på browserens Reporting API. Det ældre direktiv `report-uri` er udfaset til fordel for det. Den gældende tekst er Content Security Policy Level 3, et W3C Working Draft fra arbejdsgruppen Web Application Security.

Brug begrebet i en beslutning

Brug »Hvad er Content Security Policy?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Konti, adgangskoder og rettigheder sat op, så folk har præcis den adgang, jobbet kræver, og fratrådte mister den samme dag.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har skrevet 1.000 RBAC-regler til Linux-servere — 1,000 adgangsregler skrevet for at lukke et sikkerhedshul

Har skrevet 1.000 RBAC-regler til Linux-servere Kontakt os →

Hvorfor det betyder noget

CSP blev bygget mod cross-site scripting, ofte kaldt XSS. En angriber får tekst efter eget valg ind på en side, gennem et kommentarfelt, et søgeord, som siden gentager i sin HTML, eller en kompromitteret afhængighed, og browseren kører teksten som script med al den myndighed, sitets egen kode har. Den kan læse siden, handle som den indloggede bruger og sende det, den finder, hvorhen den vil. Første forsvar er at escape output korrekt, og det skal sidde rigtigt i hver skabelon, på hver linje, hver gang. CSP er andet forsvar. Når en fejl i escapingen alligevel slipper igennem, skal det indsprøjtede script stadig forbi politikken, og en politik, der kun kører scripts med netop dette svars nonce, afviser det.

Politikken begrænser også, hvad et script, der faktisk kører, kan nå. `connect-src` bestemmer, hvilke værter et script må tale med, og det indsnævrer, hvor en skimmer kan sende det, den samler op. `frame-ancestors` bestemmer, hvilke sites der må vise siden i en ramme, og er standardforsvaret mod clickjacking, hvor én side lægges oven på en anden, så et klik lander et sted, brugeren aldrig sigtede

efter. `form-action` bestemmer, hvorhen formularer må sende, og `base-uri` forhindrer et indsprøjtet `base` -element i stille at flytte hver relativ adresse på siden et andet sted hen.

Det er værd at være præcis om, hvem CSP beskytter. Den kører i browseren og værner altså læseren af siden mod indhold, som en angriber har plantet. Over for den samme læser gør den intet for serveren, for vedkommende kan rette eller fjerne headeren i sin egen browser på få sekunder. En politik er en grund til at stole mindre på siden, aldrig en grund til at stole mere på forespørgslen.

Prisen er først og fremmest overblik. En politik tvinger et team til at kende hver oprindelse, siderne henter fra, hvert inline-script og hver hændelseshandler, der er skrevet som attribut. Et nyt site kan bygges til en stram politik fra første commit for meget lidt. Et ældre site med inline-handlere, en tag manager og en håndfuld widgets skal betale for oprydningen først, og den oprydning er som regel det egentlige projekt.

Sådan ser det ud i praksis

Dette site lever selv under en. Sikkerhedssiden nævner en stram Content Security Policy, erklæret i siden selv, med `form-action 'none'` , fordi sitet ikke sender formularer nogen steder. Hver HTML-side angiver `script-src 'none'` : hverken JavaScript eller en service worker kører. Sitet sender ingen tredjepartsforespørgsler; alt, hvad en side skal bruge, kommer fra sitets egen vært, så der er meget lidt at give lov til.

En applikation med indloggede brugere har brug for mere end en kort liste over tilladte kilder. På en platform med professionelle og organisatoriske data sætter Next.js' middleware en Content-Security-Policy med en nonce pr. request og `strict-dynamic` , sammen med HSTS og SameSite-cookies. En frisk nonce i hvert svar betyder, at kun de script-elementer, serveren har stemplet i netop det svar, kommer til at køre. Med `strict-dynamic` i politikken må et script, som noncen stoler på, selv hente flere scripts, så politikken ikke behøver at nævne hver vært, et bundt henter sine dele fra. Siden er også ligefrem om, hvor browserens rolle slutter. Dens arbejdsantagelse er, at klienten er fjendtlig, så rettighederne genkontrolleres på serveren ved hver request, og gates i frontenden behandles kun som UX. CSP er ét lag blandt flere dér. Den beskytter brugerne mod indsprøjtet kode, mens spørgsmålet om, hvem der må hvad, afgøres et andet sted.

Hvor det går galt

Politikken går live, og halvdelen af sitet holder op med at virke. En stram politik, der sendes direkte til håndhævelse, blokerer det inline-script, som alle havde glemt, og det sporingstag, nogen satte ind for to år siden. Send den først som `Content-Security-Policy-Report-Only`, læs rapporterne, til de falder til ro, tillad eller fjern hver kilde, de nævner, og håndhæv først derefter. Report-only-headeren skal være en rigtig header, for specifikationen understøtter den ikke i et `meta`-element.

Listen stoler på en hel vært. `script-src https://cdn.example.com` tillader hver eneste fil på den vært, også gamle biblioteksversioner og endepunkter, der sender en kalders input retur som script. Specifikationen siger selv, at det er en udfordring at udrulle en politik, der virker mod cross-site scripting, og anbefaler det, den kalder en Strict CSP: `script-src` bygget af nonces eller hashes med `'strict-dynamic'`, plus `base-uri` sat til `'self'` eller `'none'`.

Man tror, at `'unsafe-inline'` tillader alle inline-scripts. Så snart en nonce eller en hash står i samme kildeliste, ignorerer browsere, der forstår dem, `'unsafe-inline'`. Reglen lader en politik beholde `'unsafe-inline'` som reserve for browsere, der er for gamle til at kende nonces, mens nyere browsere bruger den stramme liste. Den overrasker til gengæld det team, der giver ét script en nonce og opdager, at alle de andre inline-scripts nu bliver blokeret.

Noncen holder op med at være hemmelig. Specifikationen kræver en ny værdi, hver gang en politik sendes, og beder om mindst 128 bit fra en kryptografisk sikker generator. En nonce, der er hårdkodet i en skabelon, eller en hel side, der ligger i en delt cache med sin nonce og bliver serveret til alle, giver en angriber præcis den værdi, der skal til for at stemple sit eget script.

En politik i et `meta`-element gør mindre end den samme politik i en header. Browseren ignorerer `frame-ancestors`, `report-uri` og `sandbox`, når de kommer den vej, og politikken gælder kun det, der står efter elementet i dokumentet, så alt over det i head indlæses uden kontrol. Et site uden indflydelse på sine svarheadere får stadig reel beskyttelse af en `meta`-politik, placeret så tidligt i head, som den kan komme, men beskyttelse mod at blive vist i fremmede rammer kræver headeren.

En ekstra politik strammer i stedet for at løsne. Et svar kan bære flere politikker, fra et framework, en proxy og et CDN, og en ressource skal igennem dem alle. En ekstra politik kan kun begrænse, så en eftergivende header, der tilføjes længere nede i kæden for at redde en blokeret forespørgsel, gør ingen forskel. Rettelsen skal ind i den politik, der blokerede.

Læs mere

- Content Security Policy Level 3 — selve standarden, med anbefalingen om Strict CSP og afsnit om, hvordan nonces bliver kapret og lækket.
- HTML Standard: meta-elementet — den pragma, der anvender en politik erklæret i siden, de direktiver, den dropper, og hvorfor ressourcer hentet før den kan slippe forbi.
- Reporting API — endepunktsmekanismen bag `report-to`, til at læse overtrædelser, før politikken håndhæves.
- Trusted Types — søsterstandarden, som direktivet `require-trusted-types-for` slår til, rettet mod script, der sprøjtes ind gennem DOM'en.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er Content Security Policy?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/csp/