Engineer

Content Security Policy

en liste over hvor en side får laste ting fra

Hva det er

Content Security Policy er et sett med regler et nettsted gir nettleseren om hva sidene får kjøre, hente og sende inn, og som nettleseren deretter håndhever på nettstedets vegne. Navnet forkortes nesten alltid til CSP. Nettstedet sender reglene i svarheaderen `Content-Security-Policy`, eller skriver dem inn i siden i et `meta`-element, og fra det øyeblikket blir alt reglene ikke tillater, avvist før det lastes, enten det er et skript, et stilark, et bilde, en skrift, en ramme eller en tilkobling.

En policy er en kort streng med direktiver skilt med semikolon. Hvert direktiv navngir én type ressurs eller atferd og kildene det godtar. Her er en streng policy, vist med ett direktiv per linje for lesbarhetens skyld, selv om den over nettet er én enkelt headerverdi:

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

Vokabularet er lite. `'self'` betyr sidens eget opphav (*origin*). En vert som `https://static.example.com` eller et URL-skjema som `https:` utvider det. `'none'` tillater ingenting. En nonce er en tilfeldig verdi serveren lager for ett enkelt svar og skriver både inn i policyen og på hvert skriptelement den går god for. En hash er sammendraget (*digest*) av den nøyaktige teksten i et innebygd skript, slik at akkurat det skriptet og ikke noe annet får kjøre. To nøkkelord, `'unsafe-inline'` og `'unsafe-eval'`, slår beskyttelsen av igjen, og navnene deres er en rimelig advarsel.

`default-src` er reserven for de hentedirektivene (*fetch directives*) en policy utelater, som `script-src`, `style-src`, `img-src` og `connect-src`. Den er en reserve for dem og bare dem. Direktiver som styrer atferd og ikke henting, blant dem `base-uri`, `form-action` og `frame-ancestors`, forblir ubegrenset helt til de skrives ut.

En policy kan også kjøre i en observerende modus. Sendt som `Content-Security-Policy-Report-Only` blokkerer den ingenting og rapporterer i stedet hvert brudd til et endepunkt navngitt med `report-to`, som bruker nettleserens Reporting API. Det eldre direktivet `report-uri` er foreldet til fordel for det. Den gjeldende teksten i standarden er Content Security Policy Level 3, et W3C Working Draft fra arbeidsgruppen Web Application Security Working Group.

Bruk begrepet i en beslutning

Bruk «Hva er Content Security Policy?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Kontoer, passord og tilganger ordnet slik at folk har nøyaktig den tilgangen jobben krever, og de som slutter, mister den samme dag.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Skrev 1 000 RBAC-regler for å sikre Linux-servere — 1,000 tilgangsregler skrevet for å tette et sikkerhetshull

Skrev 1 000 RBAC-regler for å sikre Linux-servere Ta kontakt →

Hvorfor det er viktig

Angrepet CSP ble bygd mot, er krysskripting (*cross-site scripting*, XSS). Noen får tekst de selv har valgt, inn i en side, gjennom et kommentarfelt, et søkeord som gjentas i HTML-en, eller en kompromittert avhengighet, og nettleseren kjører den som skript med all den myndigheten nettstedets egen kode har: den kan lese siden, opptre som den innloggede personen og sende det den finner, hvor som helst. Det første forsvaret er riktig koding av utdata (*escaping*), og den må være riktig i hver mal, på hver linje, hver gang. CSP er det andre forsvaret. Når en glipp i kodingen likevel slipper gjennom, må det innsmuglede skriptet fortsatt forbi policyen, og en policy som bare kjører skript med nettopp dette svarets nonce, avviser det.

Den begrenser også hva et skript som faktisk kjører, kan nå. `connect-src` avgjør hvilke verter skript får snakke med, noe som snevrer inn hvor en skimmer kan poste det den samler inn. `frame-ancestors` avgjør hvilke nettsteder som får vise siden inne i en ramme. Det er standardforsvaret mot *clickjacking*, der en side legges over en annen slik at et klikk havner et sted den som klikket, aldri

mente. `form-action` avgjør hvor et skjema får sende inn, og `base-uri` hindrer et innsmuglet `base`-element i å omdirigere hver relative adresse på siden i det stille.

Det er verdt å være presis om hvem CSP beskytter. Den kjører i nettleseren, så den vokter personen som leser siden, mot innhold en angriper har plantet der. Den gjør ingenting for serveren mot den samme personen, som kan endre eller fjerne headeren i sin egen nettleser på sekunder. En policy er en grunn til å stole mindre på siden, aldri en grunn til å stole mer på forespørselen.

Kostnaden er for det meste kartlegging. En policy tvinger et team til å kjenne hvert opphav sidene laster fra, hvert innebygd skript og hver hendelseshåndterer som er skrevet som et attributt. Et nytt nettsted kan bygges etter en streng policy fra første commit for svært lite. Et eldre nettsted med innebygde håndterere, en tag manager og flere widgeter betaler for oppryddingen først, og den oppryddingen er som regel det egentlige prosjektet.

Slik ser det ut i praksis

Dette nettstedet holder seg selv til en slik policy. Sikkerhetssiden lister en streng Content Security Policy, deklarert i selve siden, med `form-action 'none'`, fordi nettstedet ikke sender noen skjemaer. Hver HTML-side angir `script-src 'none'`: verken JavaScript eller en service worker kjører. Nettstedet sender ingen forespørsler til tredjeparter; alt en side trenger, kommer fra nettstedets egen vert, så det er svært lite å tillate.

En applikasjon med innloggede personer trenger mer enn en kort tillatelsesliste. På en plattform med faglige og organisatoriske data setter proxy-mellomvaren i Next.js en Content-Security-Policy med en nonce per forespørsel og `strict-dynamic`, i tillegg til HSTS og SameSite-cookies. En ny nonce på hvert svar betyr at bare skriptelementene serveren stemplet på det svaret, får kjøre. Med `strict-dynamic` i policyen får et skript som noncen stoler på, laste flere skript selv, så policyen trenger ikke å navngi hver vert en pakke henter bitene sine fra.

Den samme siden sier tydelig hvor nettleserens del slutter: «Arbeidsantakelsen må være at klienten er fiendtlig». Derfor bestemmer policyen bare hvilken kode nettleseren kjører. Hvem som får gjøre hva, avgjør serveren selv for hver forespørsel, og sperrene i frontenden er UX og ikke mer, eller med sidens egne ord: «Sperrene avgjør hva man ser; serveren avgjør hva man kan gjøre.» CSP er ett lag av flere der, og jobben er å beskytte de innloggede mot innsmuglet kode.

Hvor det går galt

Policyen går i produksjon, og halve nettstedet slutter å virke. En streng policy som sendes rett til håndheving, blokkerer det innebygde skriptet ingen husket, og sporingstaggen noen la til for to år siden. Send den som `Content-Security-Policy-Report-Only` først, les rapportene til de stilner, tillat eller fjern hver kilde de navngir, og håndhev først da. Headeren for report-only må være en ekte header: spesifikasjonen støtter den ikke i et `meta` -element.

Tillatelseslisten stoler på en hel vert. `script-src https://cdn.example.com` tillater hver fil på den verten, også gamle bibliotekversjoner og endepunkter som sender inndata fra den som kaller, tilbake som skript. Spesifikasjonen sier selv at det er en utfordring å rulle ut en policy som virker mot krysskripting, og anbefaler det den kaller en Strict CSP: `script-src` bygd av nonces eller hasher med `'strict-dynamic'` , pluss `base-uri` satt til `'self'` eller `'none'` .

`'unsafe-inline'` **antas å tillate alle innebygde skript.** Så snart en nonce eller en hash står i den samme kildelisten, overser nettlesere som forstår dem, `'unsafe-inline'` . Regelen lar en policy beholde `'unsafe-inline'` som reserve for nettlesere som er for gamle til å kjenne nonces, mens nyere nettlesere bruker den strenge listen. Den overrasker et team som legger en nonce på ett skript og finner de andre innebygde skriptene sine blokkert.

Noncen slutter å være en hemmelighet. Spesifikasjonen krever en ny verdi hver gang en policy sendes, og ber om minst 128 bit fra en kryptografisk sikker generator. En nonce som er hardkodet i en mal, eller en hel side som lagres med noncen sin i en delt cache og serveres til alle, gir en angriper akkurat den verdien som trengs for å stemple sitt eget skript.

En policy i et `meta` -element gjør mindre enn den samme policyen i en header. Nettleseren overser `frame-ancestors` , `report-uri` og `sandbox` når de kommer den veien, og policyen styrer bare det som kommer etter elementet i dokumentet, så alt over det i `head` lastes ukontrollert. Et nettsted som ikke rår over headerne i sine egne svar, får likevel ekte beskyttelse av en `meta` -policy, plassert så tidlig i `head` som den kan stå, men beskyttelse mot innramming krever headeren.

En policy nummer to strammer inn i stedet for å løsne. Et svar kan bære flere policyer, fra et rammeverk, en proxy og et CDN, og en ressurs må bestå hver eneste av dem. Å legge til en policy kan bare begrense, så en romslig header som

legges til lenger ned i kjeden for å rette en blokkert forespørsel, gjør ingenting. Rettelsen må inn i policyen som blokkerte den.

Les mer

- Content Security Policy Level 3 — selve standarden, med anbefalingen om Strict CSP og et avsnitt om hvordan nonces blir stjålet.
- HTML Standard: the meta element — pragmaet som tar i bruk en policy deklarert i siden, direktivene det dropper, og hvorfor ressurser hentet før det kan slippe forbi.
- Reporting API — endepunktmekanismen bak `report-to`, for å lese brudd før policyen håndheves.
- Trusted Types — følgestandarden som direktivet `require-trusted-types-for` slår på, rettet mot skript som smugles inn gjennom DOM-en.

Ordliste

Velg ut fra deres faktiske rammer

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er Content Security Policy?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/csp/