Engineer

HTTPS

🔑 de samme forespørgsler over en krypteret forbindelse

Hvad det er

HTTPS er HTTP sendt gennem en TLS-forbindelse. Forespørgsler og svar er de samme beskeder som før, med de samme metoder, statuskoder og headere; det, der ændrer sig, er forbindelsen under dem, som nu er krypteret og autentificeret. Som URI-skema er `https` defineret i RFC 9110, specifikationen af HTTP's semantik fra juni 2022, der afløste RFC 2818, det korte dokument fra maj 2000, som først beskrev HTTP over TLS. Standardporten er 443, hvor ren HTTP bruger 80.

RFC 9110 knytter to regler til dette skema. En klient skal kontrollere, at serverens identitet er et acceptabelt match for værten i adressen, hvilket i praksis betyder, at certifikatet skal dække navnet; en forbindelse til det forkerte navn er mislykket, uanset hvor stærk krypteringen er. Og `http://example.com/` og `https://example.com/` er forskellige origins med hver sit navnerum. For protokollen deler de et værtsnavn og ellers ingenting.

HTTPS er ikke bundet til én version af HTTP. HTTP/1.1 og HTTP/2 kører over TLS på TCP. HTTP/3 kører over QUIC, der bærer håndtrykket fra TLS 1.3 inde i selve transporten, og RFC 9114 udelukker at bruge det til at nå en ren `http://` -server direkte, så den nyeste version kommer med HTTPS eller slet ikke.

Hvad krypteringen dækker, ses lettest med en adresse. For `https://shop.example/orders/1234?ref=mail` er stien, query-strengen, alle headere inklusive cookies og `Authorization` og begge beskedkroppe krypteret. Netværket mellem den besøgende og serveren ser stadig serverens IP-adresse og port, trafikkens timing og størrelse og i de fleste forbindelser også værtsnavnet `shop.example` : browseren slog det op i DNS, og klienten nævner det i den første TLS-besked, så én adresse kan betjene mange sites. HTTPS skjuler, hvad der blev spurgt om, ikke hvem der blev spurgt.

Brug begrebet i en beslutning

Brug »Hvad er HTTPS?« 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

Det åbne spørgsmål for et offentligt site er, om noget overhovedet stadig skal svare på `http://`, og browserne træffer nu det meste af den beslutning selv.

Det første håndtag er den sikre kontekst (secure context), W3C's betegnelse for en side, der er leveret over HTTPS eller fra maskinen selv. Browsere forbeholder en voksende række funktioner til sikre kontekster. Service workers er en af dem: `navigator.serviceWorker` er erklæret til sikre kontekster, så på en side hentet over ren HTTP findes egenskaben simpelthen ikke, og det gør alt, der bygger på den, heller ikke, med offline-caching som det første. Specifikationen Secure Contexts anbefaler også, at forfattere af nye specifikationer beskytter følsomme funktioner på samme måde, så begrænsningen er udgangspunktet for nyt arbejde frem for en undtagelse fra sag til sag.

Det andet håndtag er blandet indhold (mixed content). Når en side selv er HTTPS, sorterer browseren hver af dens hentninger efter W3C's regler for blandet indhold. Billeder, lyd og video, der hentes over `http://`, opgraderes automatisk til `https://` og fejler, hvis den adresse ikke svarer. Scripts, stylesheets og dataforespørgsler blokeres helt med en netværksfejl. Én overset ukrypteret adresse i en skabelon kan derfor slå en hel funktion ud, ikke bare et billede.

Så er der meget lidt tilbage til port 80. En lytter, der besvarer hver forespørgsel med en permanent omdirigering til samme sti på `https://`, gør stadig nytte, fordi folk skriver navne uden skema, og gamle links stadig siger `http://`. Certifikatudstederens HTTP-udfordring ankommer også dér, når et certifikat udstedes eller fornyes. HSTS lader derefter en tilbagevendende browser springe selv omdirigeringen over. Alt ud over de opgaver på port 80 er en side, der serveres uden garantierne ovenfor.

Loopback er den eneste faste undtagelse for almindelige sites. Specifikationen regner `localhost`, `127.0.0.1` og `::1` for potentielt troværdige, så en side fra udviklerens egen maskine får funktionerne fra sikre kontekster uden et certifikat. Den bekvemmelighed for udviklere er kilden til en af fejlene nedenfor.

Sådan ser det ud i praksis

Virksomhedens eget site følger den regel. På produktionsværten besvarer Caddy hver webforespørgsel og fornyer selv HTTPS-certifikatet fra Let's Encrypt, så det er webserverens opgave at holde certifikatet gyldigt og ikke en persons. De samme sider udgives som en Tor onion-tjeneste, og den adresse begynder med `http://` med vilje. En onion-adresse indkoder tjenestens offentlige identitetsnøgle. Tor autentificerer tjenesten og krypterer forbindelsen, før der udveksles HTTP, og derfor bruger dette spejl HTTP over Tor. Samme side linker til Qualys SSL Labs, der bedømmer den aktive HTTPS-opsætning for alle, der spørger.

Kundearbejde viser reglen om sikre kontekster fra den anden side. Til en platform, hvis brugere ofte er søfolk og feltpersonale på telefoner, tit uden forbindelse, leverede virksomheden fuld understøttelse af Progressive Web App med en service worker, der cacher skrifttyper, billeder, lyd, video og CDN-ressourcer efter cache-first-princippet. Fordi service workers kun fungerer ordentligt over HTTPS, blev offline- og installationsadfærden testet over HTTPS på rigtige enheder i stedet for at blive taget for givet. Det samme behov viser sig i værktøjerne til et monorepo på mange sprog med 578 task-mål, hvor det go-task-lag, virksomheden skrev, sætter en HTTPS-variant til test af PWA og mobil ved siden af udviklingstilstandene native og Docker.

Hvor det går galt

Scripts holder op med at køre efter skiftet til HTTPS, og nogle billeder forsvinder. Siderne indeholder stadig `http://`-adresser, hårdkodet i en skabelon, gemt i CMS-indhold eller indsat med et tredjepartsuddrag. Browseren blokerer scripts og stylesheets og opgraderer billederne, som så går i stykker der, hvor den anden vært mangler HTTPS. Browserkonsollen nævner hver blokeret adresse. Søg i skabelonerne og det gemte indhold efter det gamle skema, og skriv hvert fund om til `https://` eller til en sti relativt til sitet:

```
grep -rnE '(src|href)="http://' templates/ content/
```

En `Content-Security-Policy: upgrade-insecure-requests`-header bærer siderne gennem oprydningen, men lader de gemte adresser være, som de var.

Sitet omdirigerer til sig selv, indtil browseren giver op. Et CDN eller en load balancer terminerer TLS og sender forespørgslen videre til applikationen over ren HTTP. Applikationen ser `http`, omdirigerer til `https`, proxyen sender det nye forsøg videre som `http` igen, og løkken ender i en fejl om for mange omdirigeringer. Samme blindhed får applikationen til at bygge absolutte links, der begynder med `http://`. Proxyen skal give det oprindelige skema videre, i standardheaderen `Forwarded: proto=https` fra RFC 7239 eller i den ældre `X-Forwarded-Proto`, og applikationen bør kun tro på headeren, når den kommer fra proxyens egen adresse, for enhver klient kan sende den. Lad derefter ét lag eje omdirigeringen, normalt det, der har certifikatet.

En sessionscookie, der er sat over HTTPS, rejser ukrypteret. RFC 9110 advarer om, at cookies ikke respekterer den grænse, den trækker mellem de to origins. Efter cookiereglerne i RFC 6265 sendes en cookie uden attributten `Secure` også med ukrypterede `http://`-forespørgsler til samme vært. En besøgende, der skriver navnet uden skema, er nok til at sende den, og det samme er en angriber på netværket, der besvarer en hvilken som helst anden ukrypteret forespørgsel med en omdirigering til sitets `http://`-adresse. Sæt `Secure` på hver cookie, sitet udsteder, og tilføj HSTS, så browseren aldrig sender den ukrypterede forespørgsel.

Service workeren kører på den bærbare og mangler på telefonen. Udvikleren åbner `http://localhost:3000`, en sikker kontekst, og alt virker. Telefonen på samme netværk åbner `http://192.168.1.20:3000`, som ikke er en sikker kontekst, så `navigator.serviceWorker` er `undefined`, et bibliotek, der tjekker efter den, springer registreringen over, og offlinetilstanden mangler uden en lyd. Server også udviklingsbuildet over HTTPS. Caddy udsteder for eksempel certifikater til lokale navne og IP-adresser fra sin egen lokale certifikatudsteder og forsøger at lægge roden i maskinens tillidslager. Dens dokumentation advarer om, at det trin ikke er garanteret, og at det er administratorens opgave at få roden ind i alle tillidslagre, og en testtelefon skal have roden installeret i hånden.

Læs mere

- RFC 9110, afsnit 4.2.2: https URI Scheme — https som URI-skema, dets standardport og reglen om, at en klient skal kontrollere serverens identitet.

- Secure Contexts — hvilke sider der regnes for sikre, hvorfor loopback er med, og rådet om at begrænse nye funktioner.
- Mixed Content — hvilke forespørgsler fra en HTTPS-side der opgraderes, og hvilke der blokeres.
- RFC 7239: Forwarded HTTP Extension — standardheaderen, som en proxy bruger til at give det skema videre, den besøgende brugte.
- Automatic HTTPS — Caddys egen beskrivelse af dens certifikater, dens omdirigering fra HTTP og dens lokale certifikatudsteder.

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 HTTPS?« 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/https/