Engineer

webhook

📢 en HTTP-forespørgsel, som en tjeneste sender, når noget sker

Hvad det er

En webhook er en HTTP-forespørgsel, som én tjeneste sender til en anden i det øjeblik, noget sker hos den: en betaling går igennem, en faktura bliver betalt, en lang opgave bliver færdig. Den modtagende side registrerer på forhånd en URL, og afsenderen sender en beskrivelse af hver hændelse dertil med POST. Standard Webhooks-specifikationen kalder webhooks »a common name for HTTP callbacks« og beskriver dem som en måde, hvorpå tjenester kan underrette hinanden om hændelser. Ordet skrives af og til i to ord, web hook, men Stripe, GitHub og specifikationen skriver det alle i ét ord.

Specifikationen beskriver også en webhook som en del af en tjenestes API, en slags API, der kører baglæns. Normalt kalder klienten tjenesten, når den vil have noget. Med en webhook kalder tjenesten klienten, når den har nyt. I en typisk API-integration kører begge sider en HTTP-server, den ene til at modtage API-kald og den anden til at modtage webhooks. En tjeneste kan også sende webhooks uden selv at tilbyde et forespørgsels-API. Specifikationen kalder afsenderen producer og modtageren consumer, og den URL, modtageren registrerer, er dens webhook-endpoint.

Specifikationen anbefaler en HTTP POST med en JSON-brødtekst, men kræver ikke et bestemt format eller en bestemt struktur for payloaden. Den foreslåede brødtekst har en `type`, skrevet med punktummer og ordnet hierarkisk, for eksempel `invoice.paid`; et `timestamp` for, hvornår hændelsen fandt sted, i ISO 8601; og hændelsens data enten under `data` eller på øverste niveau. Yderligere metadata kan stå begge steder. Leveringsmetadata ligger i headere: `webhook-id` identificerer hændelsen, `webhook-timestamp` angiver, hvornår netop dette forsøg blev foretaget, i Unix-sekunder, og `webhook-signature` bekræfter dets ægthed. Stripe og GitHub bruger deres egne headernavne, blandt andre `Stripe-Signature`

og `X-Hub-Signature-256`, og deres verificeringsregler er forskellige: Stripe signerer et forsøgstidsstempel sammen med brødteksten, mens GitHubs HMAC kun dækker brødteksten.

En payload kan være fuld og indeholde hele det ændrede objekt, eller tynd og indeholde ikke stort andet end en identifikator, så modtageren selv må hente resten gennem API'et. Specifikationen vejer de to op mod hinanden. En fuld payload sparer modtageren for ekstra kald til API'et. En tynd payload betyder færre data at generere og sende, mere fleksibilitet og mere plads til ændringer senere, fordi en tynd payload altid kan vokse til en fuld, men ikke omvendt. Den gør også dataadgangen sporbar, fordi modtageren udtrykkeligt skal bede om det, den læser. Uanset hvad råder specifikationen til at holde payloads små, normalt under 20 kB. Stripe tilbyder et tyndt format og markerer det som det anbefalede.

Brug begrebet i en beslutning

Brug »Hvad er en webhook?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Den del, brugerne aldrig ser: hvor data ligger, hvilke regler der gælder, og hvordan andre systemer må tale med den.

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 designet og drevet 6 ETL/ELT-pipelines — 6 ETL-pipelines på tre databasemotorer

Har designet og drevet 6 ETL/ELT-pipelines Kontakt os →

Hvorfor det betyder noget

Uden webhooks må et system, der skal vide besked om en ændring et andet sted, polle: spørge igen og igen, for det meste bare for at høre, at intet er sket, og alligevel først opdage ændringen ved næste runde. En webhook vender det om. Den tjeneste, der kender til hændelsen, kalder én gang, i det øjeblik den sker. Sådan melder betalingsudbydere, kodehostingtjenester og e-mailudbydere tilbage til de systemer, der er bygget oven på dem.

Den omvendte retning har en pris, og det meste af den betales af den modtagende ende. Dens endpoint er en offentlig URL, som alle på internettet kan sende en POST til, og intet ved en indgående forespørgsel viser, hvem der har sendt den, så specifikationen gør verificering af ægthed til et krav for enhver sikker implementering. Leveringen prøves igen, indtil modtageren svarer med en 2xx-status, eller leveringspolitikken stopper yderligere forsøg, så den samme hændelse kan ankomme mere end én gang; Stripe advarer om, at endpoints af

og til kan modtage dubletter. Rækkefølgen er heller ikke lovet: Stripe skriver ligeud, at den ikke garanterer, at hændelser ankommer i den rækkefølge, de blev skabt i.

Afsenderen må på sin side blive ved med at prøve. Specifikationen anbefaler en tidsplan for nye forsøg, der strækker sig over flere dage, med eksponentiel backoff og lidt tilfældig jitter. I dens eksempel gøres der ti forsøg, det sidste 75 timer, 35 minutter og 5 sekunder efter det første. Stripe prøver igen i op til tre dage i live mode. En modtager, der kommer tilbage inden for det vindue, indhenter selv det forsømte; derefter kan man hos Stripe sende en hændelse igen manuelt i op til 15 dage efter, at den blev oprettet, eller i op til 30 dage med Stripes CLI. Når leveringen bliver ved med at fejle i lang tid, anbefaler specifikationen, at afsenderen giver modtageren besked ad en anden kanal, for eksempel e-mail, og deaktiverer endpointet.

Sådan ser det ud i praksis

Her er headerne på en levering i specifikationens format. Identifikatoren, tidsstemplet og signaturen er de eksempelværdier, specifikationen selv bruger; værdien af `Host` og stien er opdigtede:

```
POST /webhooks/billing HTTP/1.1
Host: example.com
Content-Type: application/json
webhook-id: msg_2KWPBgLlAfxdpx2AI54pPJ85f4W
webhook-timestamp: 1674087231
webhook-signature: v1,K5oZfzN95Z9UVu1EsfQmfVNQhnkZ2pj9o9NDN/H/pI4=
```

Og her er brødteksten, en tynd payload fra samme specifikation, her vist med indrykning. Når den faktisk sendes, anbefaler specifikationen, at den er minificeret:

```
{
  "type": "contact.created",
  "timestamp": "2022-11-03T20:26:10.344522Z",
  "data": {
    "id": "1f81eb52-5198-4599-803e-771906343485"
  }
}
```

Præfikset `v1` markerer en symmetrisk signatur: HMAC-SHA256 over hændelsens identifikator, tidsstemplet og brødteksten, sat sammen med punktummer og beregnet med en hemmelig nøgle, der er unik for endpointet. Fordi identifikatoren og tidsstemplet signeres sammen med brødteksten, kan ingen af dem ændres, uden at signaturen bliver ugyldig. Specifikationen tillader også asymmetriske ed25519-signaturer, markeret `v1a`, og anbefaler dem, når man ikke selv kontrollerer sikkerheden på både afsender- og modtagersiden.

GitHub signerer også sine leveringer med HMAC-SHA256, lægger hex-digesten i `X-Hub-Signature-256` med præfikset `sha256=` og offentliggør en hemmelighed, en payload og en signatur, som man kan afprøve en verifikator mod. Her er en verifikator i Python, afprøvet mod netop det eksempel:

```
import hashlib
import hmac


def verify(body: bytes, secret: str, header: str) -> bool:
    mac = hmac.new(secret.encode("utf-8"), body, hashlib.sha256)
    expected = "sha256=" + mac.hexdigest()
    return hmac.compare_digest(expected, header)


secret = "It's a Secret to Everybody"
header = (
    "sha256=757107ea0eb2509fc211221cce984b8a"
    "37570b6d7586c22c46f4379c8b043e17"
)
print(verify(b"Hello, World!", secret, header))  # True
print(verify(b"Hello, World! ", secret, header))  # False
```

Ét enkelt mellemrum til sidst i brødteksten er nok til at få verificeringen til at fejle. `hmac.compare_digest` bruger den samme tid, uanset om strengene afviger tidligt eller sent; GitHubs dokumentation siger, at man aldrig må sammenligne signaturer med et almindeligt `==`. En solid handler arbejder derefter i en fast rækkefølge: læs de rå bytes, verificér signaturen over præcis de bytes, kontrollér, at forsøgstidsstemplet er tilstrækkeligt nyt, hvis afsenderen signerer et, kontrollér, om hændelsens identifikator er set før, læg hændelsen i en kø, og svar med en 2xx. Det egentlige arbejde sker bagefter. Stripe nævner som eksempel at svare med 200, før en kundes faktura markeres som betalt i regnskabssystemet.

To projekter i sitets portefølje beskæftiger sig med webhooks fra den modtagende ende. I projektet om adgangsret via Stripe og in-app-køb skal et abonnement blandt andet kunne overleve en webhook, der ankommer to gange, og en webhook, der ankommer i forkert rækkefølge. I e-mailsystemet med failover og leveringslog melder indgående webhooks tilbage, hvad hver udbyder gjorde med en besked, og projektet registrerer både det, der blev sendt, og det, der kom tilbage, i en afsendelseslog og en tabel over leveringshændelser.

Hvor det går galt

Signaturen passer aldrig. En almindelig årsag er, at modtagerens framework parser JSON-brødteksten, og at koden serialiserer den igen, før HMAC'en beregnes. Specifikationen kalder det en meget almindelig fejl: JSON-serialisering er forskellig fra implementering til implementering, og et enkelt overflødigt mellemrum gør signaturen ugyldig. Stripe skal have den rå brødtekst af samme grund og advarer om, at enhver ændring af den får verificeringen til at fejle. GitHub tilføjer, at en proxy eller load balancer foran endpointet ikke må ændre payloaden eller headerne, og at payloaden bør håndteres som UTF-8. Beregn signaturen over de bytes, præcis som de ankom, før noget parser dem.

Den samme hændelse ankommer to gange. Nye forsøg og netværksproblemer giver begge dubletter. Ifølge specifikationen er `webhook-id` den samme ved hvert nyt forsøg med en hændelse og fungerer som idempotensnøgle; dens eksempel er at gemme identifikatorerne i Redis i fem minutter. Stripe beder dig registrere de hændelses-id'er, du har behandlet, og springe dem over, du allerede har set. Stripe bemærker også, at der i nogle tilfælde oprettes og sendes to separate Event-objekter; dem kan man genkende på objektets id i `data.object` sammen med hændelsestypen. GitHubs `X-GitHub-Delivery` -header forbliver den samme, når en levering bliver sendt igen. Behandlingen bør være idempotent, så en anden levering ikke ændrer noget.

En opsnappet levering bliver afspillet igen. En gyldig signatur forbliver gyldig, så en angriber, der opsnapper én levering, kan sende den igen. Derfor signeres tidsstemplet: specifikationen siger, at et `webhook-timestamp` uden for en tilladt tolerance skal afvises. Stripes biblioteker bruger som standard fem minutter, Stripe advarer om, at en tolerance på 0 slår kontrollen helt fra, og anbefaler NTP til at holde serverens ur præcist. Både Stripe og specifikationen sætter et nyt tidsstempel på hvert nyt forsøg, så alderen måles fra det forsøg og ikke fra den oprindelige hændelse; et forsinket forsøg kan stadig falde uden for tolerancen. Gem de sete identifikatorer mindst lige så længe som tolerancen, så bliver alt, der afspilles igen inden for vinduet, fanget som en dublet.

Hændelserne ankommer i forkert rækkefølge. Stripes eksempel: når et abonnement oprettes, kan det give `customer.subscription.created`, `invoice.created`, `invoice.paid` og, hvis der er en debitering, `charge.created`, og intet garanterer, at de ankommer i den rækkefølge. Stripes snapshot-hændelser angiver `created` i hele sekunder, så forskellige hændelser kan have samme tidsstempel, og Stripe siger, at man ikke skal bruge det til at sortere hændelser eller finde dubletter. Den robuste fremgangsmåde behandler en hændelse som en besked om, at noget har ændret sig, og henter objektets aktuelle tilstand fra API'et, før den handler på den.

Handleren er for langsom. GitHub forventer en 2xx inden for 10 sekunder; derefter afbryder GitHub forbindelsen og regner leveringen som mislykket. Specifikationen anbefaler en timeout for forespørgslen på et sted mellem 15 og 30 sekunder. En handler, der udfører det egentlige arbejde, inden den svarer, løber ind i timeouten, afsenderen prøver igen, og arbejdet bliver udført to gange. Stripe og GitHub anbefaler begge en kø. Stripe peger på spidsbelastningen i starten af måneden, når alle abonnementer fornyes.

Kun en 2xx tæller. Alt andet er en fejl, også en 3xx, for både specifikationen og Stripe, så et endpoint, der er flyttet bag en omdirigering, fejler ved hver levering, indtil den registrerede URL bliver opdateret. Specifikationen giver nogle få koder en betydning for afsenderen: `410 Gone` betyder, at endpointet skal deaktiveres, en 429, 502 eller 504 betyder, at afsenderen skal sætte tempoet ned, og en `retry-after`-header bør styre det næste forsøg.

For afsenderen er URL'en fjendtligt input. Kunder kan angive en hvilken som helst URL, og afsenderens servere vil kalde den. Specifikationen kalder webhook-systemer særligt sårbare over for server-side request forgery (SSRF): en URL, der peger på en intern adresse, får afsenderen til at kalde ind i sit eget netværk. Dens råd er at sende alle webhook-forespørgsler gennem en proxy, der filtrerer interne IP-adresser fra, for eksempel Stripes smokescreen, og at køre webhook-workerne i et privat subnet, der ikke kan nå interne tjenester.

Læs mere

- Standard Webhooks specification — version 1.0.0: payloadens opbygning, signaturmetoden, headere, tidsplanen for nye forsøg, håndtering af statuskoder og SSRF.

- Receive Stripe events in your webhook endpoint — tynde payloads og snapshot-payloads, nye forsøg, dubletter, hændelser i forkert rækkefølge og beskyttelse mod replay-angreb.
- Best practices for using webhooks — GitHubs liste: få hændelser, en webhook-hemmelighed, HTTPS, et svar inden for 10 sekunder, ny levering og `X-GitHub-Delivery`.
- Validating webhook deliveries — `X-Hub-Signature-256`, sammenligning i konstant tid og et testeksempel at kontrollere en verifikator mod.

Ordliste

Vælg ud fra jeres faktiske vilkår

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er en webhook?« 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/webhook/