**Engineer** Rådgivning

# Har skrevet en parser, der tjekker hvert FFI-kald

Har skrevet en parser, der læser den rigtige C-header på 7.308 linjer og verificerer hvert kaldssted, hver enum-konstant, og at hver klasse, der ejer en pointer, er final, efter at en håndskrevet pladsholder-header lod kald til tre fjernede funktioner kompilere, linke og crashe.

september 2026

## 7,308

linjers C-header, læst i stedet for antaget

**Situation.** Tidligt i projektet blev C-grænsefladen repræsenteret af en håndskrevet header, der beskrev de funktioner, programmet forventede. Den header kunne kompileres, programmet blev linket, og kald til tre funktioner, der ikke længere fandtes i kernen, nåede frem til at blive kaldt og brød ned. Både oversætteren og linkeren havde ladet sig nøje med en beskrivelse af biblioteket og ikke selve biblioteket, og kløften viste sig først under kørsel.

**Opgave.** Den rigtige header — 7.308 linjer og 268 erklæringer — skulle være den gældende autoritet, og hver anvendelse af den i Swift-koden skulle efterprøves mod den automatisk og ikke ved gennemgang.

**Handling.** Tjekket er en parser, der læser den faktiske upstream-header og opbygger mængden af funktioner, enum-konstanter og typer, den erklærer, og derefter læser Swift-kildekoden og slår hvert kaldested og hver konstanthenvisning op i den mængde. Et kald til en funktion, headeren ikke erklærer, får buildet til at fejle. En henvisning til en enum-konstant, der er blevet omdøbt, får buildet til at fejle. Lige nu henvises der til 132 af de 268 erklæringer, og at vide, hvilke 136 der ikke bruges, er i sig selv nyttigt, for det siger præcis, hvor meget af kernen klienten ikke har nået. Parseren håndhæver også en regel, oversætteren ikke kan: hver Swift-klasse, der ejer en pointer ind i kernen, skal være final. En klasse, der ejer en pointer og ikke er final, kan nedarves, og en

underklasse, der overstyrer deinitialiseringen eller tilføjer sin egen levetid, ændrer, hvornår pointeren frigives — en use-after-free uden noget unsafe-nøgleord i nærheden. Reglen tjekkes ud fra navn i hele træet.

**Resultat.** Den defektklasse, der frembragte det oprindelige nedbrud, kan ikke gentage sig, for en forældet henvisning er nu en buildfejl og ikke en kørselsfejl. Begrænsningen er, at parseren forstår headerens erklæringer og ikke dens betydning: den beviser, at en funktion findes med et matchende navn, og en funktion, hvis betydning eller ejerskabsregel ændrede sig upstream, mens signaturen blev bevaret, slipper igennem uden bemærkning.

### Brug erfaringen i jeres projekt

Brug »Har skrevet en parser, der tjekker hvert FFI-kald« som udgangspunkt for en lignende udfordring i jeres organisation. Ydelserne bag arbejdet kan også bruges i jeres projekt. Overdragelser, driftsvejledninger og manualer, der lader jeres egne folk drive systemet uden at ringe til den, der byggede det.

**Se vores ydelser** Kontakt os →

### Se bag resultatet

Læs om situationen, beslutningerne og resultatet i eksemplet. Vi kan drøfte, hvad der passer til jeres vilkår, og hvad der kræver en anden tilgang. Har skrevet en parser, der tjekker hvert FFI-kald — 7,308 linjers C-header, læst i stedet for antaget

**Se vores ydelser** Kontakt os →

## Ville dit build fange et kald til en funktion, der er væk?

Tal om jeres projekt med den ingeniør, der har ansvaret for leverancen. Vi aftaler omfang, prisgrundlag og overdragelse skriftligt, før arbejdet begynder.

Rolle i opgaven: macOS-applikationsudvikler. Det trækker på Backend-udvikling, API'er & integration, Automatisering & CI/CD, Sikkerhed, Test & QA og Dokumentation. Og når I alligevel er her, er vi ret dygtige til Backend- & API-udvikling, DevOps & CI/CD-automatisering og Teknisk dokumentation.

**Kontakt os** Se vores ydelser

## Ét stykke af **en længere fortegnelse** — og det hele ligger her på sitet.

Hver post er skrevet på samme måde: situationen, opgaven, hvad vi gjorde, og hvad der ændrede sig.

**Se hele fortegnelsen**

### Definér den ønskede forbedring

Medbring beslutningen, de berørte mennesker og de overvejede muligheder. Vi kan gøre afvejningerne tydelige og efterlade en dokumenteret anbefaling.

**Se, hvad det koster** Kontakt os →

# Ville dit build fange et kald til en funktion, der er væk?

Spørg til beslutningerne bag »Har skrevet en parser, der tjekker hvert FFI-kald«. I taler med mennesker, der kan forklare arbejdet og vurdere jeres eget projekt.

**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://advisory.engineer.company/da/portfolio/wrote-a-parser-verifying-every-ffi-call/**