Engineer

Skrev en parser som verifierar varje FFI-anrop

Skrev en parser som läser den verkliga C-headern på 7 308 rader och verifierar varje anropsställe, varje enum-konstant och att varje klass som äger en pekare är final, efter att en handskriven platshållarheader hade låtit anrop till tre borttagna funktioner kompilera, länka och krascha.

september 2026

7,308

rader C-header, lästa i stället för antagna

Situation. Tidigt i projektet representerades C-gränssnittet av en handskriven headerfil som beskrev de funktioner som applikationen förväntade sig. Den headerfilen kompilerade, applikationen länkades, och anrop till tre funktioner som inte längre fanns i kärnan kom ända fram till att anropas och kraschade. Både kompilatorn och länkaren hade nöjt sig med en beskrivning av biblioteket i stället för biblioteket självt, och glappet visade sig först vid körning.

Uppgift. Den riktiga headerfilen — 7 308 rader och 268 deklarationer — måste bli auktoriteten, och varje användning av den i Swift-koden måste verifieras mot den automatiskt i stället för genom granskning.

Handling. Kontrollen är en parser som läser den faktiska headerfilen från uppströms och bygger upp mängden av de funktioner, enum-konstanter och typer den deklarerar, och sedan läser Swift-källkoden och slår upp varje anropsställe och varje konstantreferens mot den mängden. Ett anrop till en funktion som headerfilen inte deklarerar får bygget att fallera. En referens till en enum-konstant som har bytt namn får bygget att fallera. Det aktuella antalet är 132 av de 268 deklarationerna som refereras, och att veta vilka 136 som inte används är i sig användbart, eftersom det säger exakt hur stor del av kärnan som klienten inte har nått. Parsern upprätthåller också en regel som kompilatorn inte kan: varje Swift-klass som äger en pekare in i kärnan måste vara final. En klass som äger en pekare och inte är final kan ärvas, och en subklass som åsidosätter

deinitieringen eller lägger till en egen livslängd ändrar när pekaren frigörs — en use-after-free utan ett enda unsafe-nyckelord i närheten. Regeln kontrolleras med namn över hela trädet.

Resultat. Den feltyp som orsakade den ursprungliga kraschen kan inte återkomma, eftersom en inaktuell referens nu är ett byggfel i stället för ett körtidsfel. Begränsningen är att parsern förstår headerfilens deklarationer men inte dess semantik: den bevisar att en funktion med ett matchande namn finns, och en funktion vars betydelse eller ägarskapsregel har ändrats uppströms medan signaturen behållits passerar utan anmärkning.

Använd erfarenheten i ert projekt

Använd ”Skrev en parser som verifierar varje FFI-anrop” som utgångspunkt för en liknande utmaning i er organisation. Tjänsterna bakom arbetet finns tillgängliga för ert projekt. Den del användarna aldrig ser: där data lagras, vilka reglerna är och hur andra system får kommunicera med den.

Se våra tjänster Hör av er →

Se bakom resultatet

Läs om situationen, besluten och resultatet i exemplet. Vi kan diskutera vad som passar era förutsättningar och vad som kräver ett annat angreppssätt. Skrev en parser som verifierar varje FFI-anrop — 7,308 rader C-header, lästa i stället för antagna

Se våra tjänster Hör av er →

Skulle ditt bygge fånga ett anrop till en funktion som inte längre finns?

Prata om ert projekt med ingenjören som ansvarar för leveransen. Vi avtalar omfattning, hur priset beräknas och överlämning skriftligt innan arbetet börjar.

Roll i uppdraget: Applikationsingenjör för macOS. Det bygger på Backendutveckling, API:er och integration, Automatisering och CI/CD, Säkerhet, Test och kvalitetssäkring och Dokumentation. Och nu när ni är här: vi är ganska bra på Backend- och API-utveckling, DevOps och CI/CD-automatisering och Teknisk dokumentation.

Hör av er Se våra tjänster

En del av **en längre historik** — och den finns i sin helhet här.

Varje post är skriven på samma sätt: situationen, uppgiften, vad vi gjorde och vad som förändrades.

Se hela historiken

Definiera förbättringen ni behöver

Beskriv arbetsflödet, befintliga gränssnitt och användarna som behöver förändringen. Vi kan definiera en testbar leverans och en överlämning som gör att teamet kan underhålla lösningen.

Se vad det kostar Hör av er →

Skulle ditt bygge fånga ett anrop till en funktion som inte längre finns?

Fråga om besluten bakom ”Skrev en parser som verifierar varje FFI-anrop”. Ni talar med personer som kan förklara arbetet och bedöma ert eget projekt.

Se våra tjänster

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

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/portfolio/wrote-a-parser-verifying-every-ffi-call/