Engineer

Validering

at tjekke input mod reglerne, før man stoler på det

Hvad det er

Validering er det trin, hvor et program sammenholder indkommende data med de regler, dataene skal følge, og afviser dem, når de ikke lever op til reglerne, før noget bliver gemt, afviklet eller sendt videre. Det har flere navne afhængigt af laget: inputvalidering for formularfelter og API-kald, skemavalidering, når reglerne er skrevet ned som et JSON Schema eller en OpenAPI-beskrivelse, og markupvalidering, når en webside tjekkes mod HTML's grammatik. Input, der består, kaldes gyldigt.

OWASP's vejledning deler arbejdet op i to spørgsmål. Syntaks handler om, hvorvidt en værdi har den forventede type og form: er det en dato, er det et heltal, kan forespørgslens indhold parses som JSON. Semantik handler om, hvorvidt værdien giver mening for den konkrete handling, så en hotelbooking skal have to gyldige datoer og desuden en slutdato efter startdatoen. At omsætte strengen »500« til et heltal besvarer det første spørgsmål og siger intet om, hvorvidt 500 nætter er et rimeligt ophold.

Markupvalidering er den ældste offentlige udgave af idéen på nettet. W3C's Markup Validation Service beskriver validering som at tjekke et dokument mod den grammatik, det selv hævder at bruge, og sammenligner sin validator med lint til C. Dokumentationen er omhyggelig med, hvad det beviser: en gyldig side er ikke nødvendigvis en god side, kun en side, som et værktøj har fundet i overensstemmelse med et bestemt sæt regler. Den forsigtighed gælder alle de andre slags. Validering fortæller, at dataene passer til de regler, man skrev, og intet om, hvorvidt reglerne var de rigtige.

To nabobegreber bliver tit blandet sammen med det. Sanering ændrer input, så det bliver acceptabelt, ved at fjerne eller omskrive dele af det, hvor validering tager imod eller afviser input, som det er. Encoding og parametrisering gør data

sikre for det sted, de skal hen, en HTML-side eller en SQL-forespørgsel, og de er stadig nødvendige, efter at valideringen er bestået.

Brug begrebet i en beslutning

Brug »Hvad er validering?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Regler og kontroller, der fanger dårlige data på vej ind, og et klart svar på, hvem der ejer hvilket tal, og hvor det kommer fra.

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. Én infrastrukturplan til 14 afdelinger — 14 forskningsafdelinger på én infrastrukturplan

Én infrastrukturplan til 14 afdelinger Kontakt os →

Hvorfor det betyder noget

Beslutningen bag validering er, hvor et system trækker sin tillidsgrænse, og hvad det tjekker der. Hvert kald fra en browser, hver besked fra en kø, hver fil fra en partner og hver konfigurationsændring fra en kollega kommer med antagelser. Bliver antagelserne tjekket ved kanten, kan resten af koden regne med dem; springes tjekket over, må hver funktion længere inde forsvare sig selv, og i praksis gør de fleste det ikke.

Den anden beslutning er, hvordan reglerne skrives. Tjek, der er skrevet i hånden endpoint for endpoint, er billige i starten og glider fra hinanden, efterhånden som systemet vokser, fordi hvert tjek er sit eget lille stykke kode. Regler, der er erklæret ét sted som typer eller som et skema, kan håndhæves af et bibliotek og udgives som dokumentation fra samme kilde, hvilket mindsker risikoen for, at beskrivelsen og adfærden glider fra hinanden. Prisen er et framework at lære og standardindstillinger at forstå, for et skema håndhæver præcis det, det siger, og intet af det, det lader usagt.

Strenghed er det sidste valg. En allowlist beskriver, hvad der er acceptabelt, og afviser alt andet, hvilket OWASP anbefaler frem for at forsøge at genkende enhver ondsindet streng. At afvise felter, som valideringsskemaet ikke kender, fanger slåfejl og forældede klienter tidligt. At afvise hele kaldet ved det første problem, med en tydelig fejl, er sikrere end at fortsætte med data, der kun er delvist tjekket.

Sådan ser det ud i praksis

Et API startede med request-validering skrevet i hånden for hvert endpoint. På det dokumenterede implementeringstrin gjorde flytningen til Huma v2 tjekkene til typer på tværs af 649 paths og 760 operationer. Hvert endpoint havde input- og output-structs, Huma genererede request-valideringen ud fra dem, og en OpenAPI-beskrivelse kom fra de samme typer. To WebSocket-endpoints blev på rå Fiber, hvor request/response-modellen ikke passede. Et tjek i CI sammenlignede de ruter, serveren registrerede, med dem, beskrivelsen annoncerede. Den dokumenterede ruteparitet var 100 %; tjek erklæret i inputtyperne erstattede den tilsvarende håndskrevne validering.

Konfiguration kræver samme behandling, og SSH-hærdningen på virksomhedens vært kører nu tre valideringstrin ved hver konvergering. Kandidatfilen syntakstjekkes, før den installeres, den samlede konfiguration tjekkes efter installationen, og til sidst læses dæmonens egne effektive indstillinger tilbage, og 24 direktiver efterprøves mod dem. Det tredje trin findes, fordi en adgangsrapport i september 2026 viste to indstillinger, som automatiseringen havde skrevet ved hver konvergering, og som dæmonen havde overskrevet, fordi styresystemets egne standarder blev læst først, og for de to vinder den første forekomst. Som siden siger, beviser en søgning efter et direktiv kun, at det blev skrevet.

Markupvalidering var en del af det dokumenterede implementeringstrin i august 2026: HTML-validering var et af tjekkene i en commit-gate med 22 lintere uden advarselsniveau, hvor hver diagnostik var en fejl, og inline-undertrykkelser var forbudt, så en ugyldig side ikke kunne committes ved at markere fejlen som acceptabel.

Hvor det går galt

Serveren stoler på det, browseren allerede har tjekket. En formular, der afviser en forkert e-mailadresse i browseren, ser valideret ud, men kaldet kan sendes helt uden om formularen. OWASP nævner tjek, der kun sker i klienten, som en almindelig faldgrube: valider på serveren, og brug browserens tjek som en hurtig hjælp til den, der skriver.

Et skema tager imod alt, det ikke nævner. I JSON Schema gør det at nævne en egenskab den ikke påkrævet, og det forbyder heller ikke andre felter, og med standardvokabularet i 2020-12 bliver nøgleordet `format` samlet op som

en annotation, mens formattjek som standard er slået fra. Dette skema ser strengt ud:

```
{
    "type": "object",
    "properties": {
        "email": { "type": "string", "format": "email" }
    }
}
```

Det tager imod `{}`, `{"email": "nope"}` og `{"emial": "a@example.com"}`. Tilføj `required` og `"additionalProperties": false`, og slå formatkontrol til i validatoren, eller tjek formatet i koden. Huma afviser for sin del ukendte felter i forespørgslens indhold som standard, dokumenterer `readOnly` og `writeOnly` uden at håndhæve dem og ignorerer ukendte query-parametre, medmindre man beder om andet.

Tjekket består på én udgave af dataene, og programmet bruger en anden. En validator, der læser det rå kald, mens handleren læser en afkodet kopi, eller en test, der kigger på filen på disken, mens dæmonen anvender en anden indstilling, har tjekket noget, systemet aldrig bruger. OWASP's råd er at validere den repræsentation, der faktisk bliver brugt. SSH-arbejdet ovenfor er samme lektie i konfiguration: først dæmonens tilbagelæsning viste, hvad der rent faktisk gjaldt.

En denylist møder den næste variant. At blokere kendte farlige tegn afviser legitimt input, for eksempel en apostrof i et efternavn, og efterlader stadig forespørgslen usikker. Fastlæg, hvad hvert felt accepterer, og afvis resten.

Gyldigt input bliver behandlet som sikkert input. En streng, der har bestået hver eneste regel, kan stadig ødelægge en SQL-sætning eller en side, når den sættes ind i den. Validering indsnævrer, hvad der kommer ind; parametriserede forespørgsler og output-encoding beskytter det sted, hvor dataene ender, og begge dele er nødvendige.

Læs mere

- OWASP Input Validation Cheat Sheet — syntaks og semantik, allowlists, grænser for parsing og typiske faldgruber.

- W3C Markup Validator documentation — validatorens egen beskrivelse af markupvalidering, med hjælpesider om, hvad gyldighed beviser, og hvor grænsen går.
- JSON Schema Validation, draft 2020-12 — valideringsnøgleordene, herunder hvordan `format` opfører sig som standard.
- Huma: Request Validation — validering genereret ud fra Go-struct-tags, med indstillingerne for strenge og løse felter.
- OpenAPI Specification — beskrivelsesformatet, hvis skemaobjekter mange API-validatorer håndhæver.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring et repræsentativt datasæt, de spørgsmål det skal besvare, og den aktuelle flaskehals. Vi kan aftale målinger af kvalitet, gennemløb og gendannelse.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er validering?« 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/validation/