Engineer

Validering

att kontrollera indata mot reglerna innan man litar på dem

Vad det är

Validering är det steg där ett program ställer inkommande data mot de regler som de är tänkta att följa och avvisar dem som inte håller måttet, innan något lagras, körs eller skickas vidare. Steget har olika namn beroende på lager: indatavalidering för formulärfält och API-anrop, schemavalidering när reglerna är nedskrivna som ett JSON Schema eller en OpenAPI-beskrivning, och validering av uppmärkning när en webbsida kontrolleras mot grammatiken i HTML. Indata som klarar kontrollen kallas giltig.

OWASP delar upp arbetet i två frågor. Syntaxen gäller om ett värde har förväntad typ och form: är det här ett datum, är det ett heltal, går anropets body att tolka som JSON. Semantiken gäller om värdet är rimligt för den operation det ska användas i, så en hotellbokning behöver två giltiga datum och dessutom ett slutdatum som ligger efter startdatumet. Att omvandla strängen ”500” till ett heltal avgör den första frågan och säger ingenting om huruvida 500 nätter är en godtagbar vistelse.

Validering av uppmärkning är den äldsta offentliga formen av idén på webben. W3C:s Markup Validation Service beskriver validering som att kontrollera ett dokument mot den grammatik som det uppger sig använda, och jämför sin validator med lint för C. Dokumentationen är noga med vad det bevisar: en giltig sida är inte nödvändigtvis en bra sida, bara en som ett verktyg har funnit förenlig med en viss uppsättning regler. Samma försiktighet gäller alla andra sorter. Validering säger att data passar de regler du skrev, men inte att reglerna var de rätta.

Två närliggande begrepp blandas ofta ihop med den. Sanering ändrar indata för att göra dem godtagbara, genom att skala bort eller skriva om delar av dem, medan validering godtar eller avvisar dem som de är. Kodning och

parametrisering gör data säkra för den plats dit de ska, en HTML-sida eller en databasfråga i SQL, och de behövs fortfarande efter att valideringen har gått igenom.

Använd begreppet i ett beslut

Använd ”Vad är validering?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Regler och kontroller som fångar dålig data på vägen in, och ett tydligt svar på vem som äger vilken siffra och var den kommer ifrån.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. En infrastrukturplan för 14 avdelningar — 14 forskningsavdelningar under en och samma infrastrukturplan

En infrastrukturplan för 14 avdelningar Hör av er →

Varför det spelar roll

Beslutet bakom validering är var ett system drar sin förtroendegräns och vad det kontrollerar där. Varje anrop från en webbläsare, varje meddelande från en kö, varje fil från en partner och varje ändring i konfigurationen från en kollega kommer med antaganden i bagaget. Att kontrollera dem vid kanten innebär att resten av koden kan lita på dem. Att hoppa över kontrollen innebär att varje funktion längre in måste försvara sig själv, och i praktiken gör de flesta inte det.

Det andra beslutet gäller hur reglerna skrivs. Kontroller som skrivs för hand, slutpunkt för slutpunkt, är billiga i början och glider isär när systemet växer, eftersom var och en är sin egen lilla kodsnutt. Regler som deklareras en gång, som typer eller som ett schema, kan upprätthållas av ett bibliotek och publiceras som dokumentation från samma källa, vilket minskar risken för att beskrivningen och beteendet glider isär. Priset är ett ramverk att lära sig och standardvärden att förstå, eftersom ett schema upprätthåller exakt det som står i det och ingenting av det som det lämnar osagt.

Striktheten är det sista valet. En tillåtelselista beskriver vad som är godtagbart och avvisar allt annat, vilket OWASP rekommenderar framför att försöka känna igen varje skadlig sträng. Att avvisa fält som ett schema inte känner till fångar stavfel och gamla klienter tidigt. Att avvisa hela anropet vid det första problemet, med ett tydligt felmeddelande, är säkrare än att fortsätta med data som bara delvis har kontrollerats.

Så ser det ut i praktiken

Ett API började med validering av anrop som skrevs för hand för varje slutpunkt. Vid det dokumenterade implementeringsskedet gjorde flytten till Huma v2 kontrollerna till typer över 649 sökvägar och 760 operationer. Varje slutpunkt hade structar för indata och utdata, Huma genererade valideringen av anropen utifrån dem, och en OpenAPI-beskrivning kom ur samma typer. Två slutpunkter för WebSocket låg kvar på ren Fiber, där modellen med anrop och svar inte passade. En kontroll i CI jämförde de rutter som servern registrerade med dem som beskrivningen utlovade. Den dokumenterade överensstämmelsen mellan rutterna var 100 %; kontroller deklarerade i indatatyperna ersatte motsvarande handskrivna validering.

Konfiguration behöver samma behandling, och SSH-härdningen på företagets värd kör nu validering i tre steg vid varje konvergenskörning. Kandidatfilen syntaxkontrolleras innan den installeras, den sammansatta konfigurationen kontrolleras efter installationen, och sedan läses daemonens egna gällande inställningar tillbaka och 24 direktiv kontrolleras mot dem. Det tredje steget finns därför att en åtkomstrapport i september 2026 hittade två inställningar som automatiseringen hade skrivit vid varje konvergenskörning och som daemonen hade åsidosatt. Operativsystemets egna standardvärden lästes först, och för just de två inställningarna vinner den första förekomsten. Som sidan uttrycker det bevisar en sökning efter ett direktiv med grep bara att det skrevs.

Validering av uppmärkning ingick i det dokumenterade implementeringsskedet i augusti 2026: HTML-validering var en av kontrollerna i en kvalitetsgrind med 22 lintverktyg vid commit, utan varningsnivå. Varje diagnostik var ett fel och undertryckningar direkt i koden var förbjudna, så att en ogiltig sida inte kunde committas genom att felet markerades som godtagbart.

Var det går fel

Servern litar på det som webbläsaren redan har kontrollerat. Ett formulär som vägrar ta emot en felaktig e-postadress i webbläsaren ser validerat ut, men anropet kan skickas helt utan formuläret. OWASP räknar kontroller enbart på klientsidan som en vanlig fallgrop: validera på servern, och behandla kontrollen i webbläsaren som en snabb ledtråd till den som skriver.

Ett schema godtar allt som det inte nämner. I JSON Schema gör en listad egenskap varken fältet obligatoriskt eller andra fält förbjudna, och under standardvokabulären i 2020-12 samlas nyckelordet `format` in som en annotering, med kontrollen av formatet avslagen som standard. Det här schemat ser strikt ut:

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

Det godtar `{}` , `{"email": "nope"}` och `{"emial": "a@example.com"}` . Lägg till `required` och `"additionalProperties": false` , och aktivera formatkontroll (format assertion) i validatorn, eller kontrollera formatet i koden. Huma avvisar för sin del okända fält i en body som standard, dokumenterar `readOnly` och `writeOnly` utan att upprätthålla dem och ignorerar okända query-parametrar om inget annat anges.

Kontrollen godkänner en version av data och programmet använder en annan. En validator som läser det råa anropet medan hanteraren läser en avkodad kopia, eller ett test som granskar filen på disken medan daemonen tillämpar en annan inställning, har kontrollerat något som systemet aldrig använder. OWASP råder dig att validera den representation som faktiskt kommer att användas. SSH-arbetet ovan är samma lärdom i konfigurationens form: bara daemonens återläsning visade vad som faktiskt gällde.

En spärrlista möter nästa variant. Att blockera kända farliga tecken avvisar legitim indata, t.ex. en apostrof i ett efternamn, och lämnar ändå databasfrågan osäker. Bestäm vad varje fält godtar och avvisa resten.

Giltig indata behandlas som säker indata. En sträng som klarade varje regel kan ändå bryta sönder en SQL-sats eller en sida när den klistras in i den. Validering begränsar vad som kommer in. Parametriserade databasfrågor och kodning av utdata skyddar platsen där det hamnar, och båda behövs.

Läs mer

- OWASP Input Validation Cheat Sheet — syntax och semantik, tillåtelselistor, gränser för tolkning och vanliga fallgropar.
- W3C Markup Validator documentation — validatorns egen beskrivning av validering av uppmärkning, med hjälpsidor om vad giltighet bevisar och vad den inte bevisar.
- JSON Schema Validation, draft 2020-12 — nyckelorden för validering, bl.a. hur `format` beter sig som standard.
- Huma: Request Validation — validering som genereras från taggar på Go-structar, med inställningarna för strikta och lösa fält.
- OpenAPI Specification — formatet för API-beskrivningar, vars schemadefinitioner många validerare för API:er upprätthåller.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med ett representativt dataset, frågorna det ska besvara och dagens flaskhals. Vi kan avtala mått på kvalitet, genomströmning och återställning.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är validering?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

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/glossary/validation/