Engineer

Validierung

Prüfung von Eingaben gegen Regeln vor ihrer Verwendung

Was es ist

Validierung ist der Schritt, in dem ein Programm eingehende Daten mit den dafür vorgesehenen Regeln vergleicht und sie zurückweist, wenn sie diese nicht erfüllen, bevor etwas gespeichert, ausgeführt oder weitergegeben wird. Je nach Schicht hat sie verschiedene Namen: Eingabevalidierung bei Formularfeldern und API-Anfragen, Schemavalidierung bei Regeln in einem JSON Schema oder einer OpenAPI-Beschreibung und Markup-Validierung bei der Prüfung einer Webseite gegen die HTML-Grammatik. Eine Eingabe, die die Prüfung besteht, heißt gültig.

Die OWASP-Empfehlungen teilen die Arbeit in zwei Fragen. Die Syntax betrifft den erwarteten Typ und das Format: Ist das ein Datum, eine ganze Zahl, lässt sich dieser Anfragekörper als JSON parsen? Die Semantik fragt, „ob der Wert für den Vorgang sinnvoll ist“. Eine Hotelbuchung braucht deshalb zwei gültige Datumswerte und außerdem ein Enddatum nach dem Anfangsdatum. Die Zeichenfolge “500” in eine ganze Zahl umzuwandeln beantwortet die erste Frage und sagt nichts darüber, ob 500 Übernachtungen ein akzeptabler Aufenthalt sind.

Markup-Validierung ist die älteste öffentlich verfügbare Form dieser Idee im Web. Der W3C Markup Validation Service beschreibt sie als Prüfung eines Dokuments gegen die Grammatik, die es zu verwenden behauptet, und vergleicht seinen Validator mit lint für C. Seine Dokumentation grenzt sorgfältig ein, was das beweist: Eine gültige Seite ist nicht unbedingt eine gute Seite, sondern nur eine, die ein Werkzeug als mit bestimmten Regeln vereinbar erkannt hat. Dieselbe Vorsicht gilt für jede andere Art. Validierung sagt, dass die Daten zu den geschriebenen Regeln passen; sie sagt nicht, dass es die richtigen Regeln waren.

Zwei benachbarte Verfahren werden oft damit verwechselt. Bereinigung verändert Eingaben, um sie akzeptabel zu machen, indem Teile entfernt oder umgeschrieben werden. Validierung dagegen nimmt die Eingabe unverändert an oder weist sie zurück. Kodierung und Parametrisierung machen Daten für ihr Ziel sicher, etwa eine HTML-Seite oder eine SQL-Abfrage. Sie bleiben auch nach bestandener Validierung erforderlich.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Validierung?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Regeln und Prüfungen fangen fehlerhafte Daten beim Eingang ab und klären, wer für welche Zahl verantwortlich ist und woher sie stammt.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. Ein Infrastrukturplan für 14 Abteilungen — 14 Forschungsabteilungen mit einem gemeinsamen Infrastrukturplan

Ein Infrastrukturplan für 14 Abteilungen Kontakt aufnehmen →

Warum es wichtig ist

Die Entscheidung hinter der Validierung betrifft die Vertrauensgrenze des Systems und die Prüfungen an dieser Grenze. Jede Browseranfrage, jede Nachricht aus einer Warteschlange, jede Datei eines Partners und jede Konfigurationsänderung eines Kollegen bringt Annahmen mit. Werden sie am Rand geprüft, kann sich der restliche Code darauf verlassen. Ohne die Prüfung muss sich jede nachfolgende Funktion selbst verteidigen, was in der Praxis die meisten nicht tun werden.

Die zweite Entscheidung betrifft die Form der Regeln. Für jeden Endpunkt von Hand geschriebene Prüfungen sind anfangs günstig. Mit wachsendem System entwickeln sie sich auseinander, weil jede ein eigenes kleines Stück Code ist. Einmal als Typen oder Schema deklarierte Regeln kann eine Bibliothek durchsetzen und aus derselben Quelle als Dokumentation veröffentlichen. Das verringert das Risiko, dass Beschreibung und Verhalten auseinanderdriften. Der Preis sind ein zu erlernendes Framework und Voreinstellungen, die verstanden werden müssen: Ein Schema erzwingt genau das, was es sagt, und nichts, was es offenlässt.

Die letzte Wahl betrifft die Strenge. Eine Positivliste beschreibt das Zulässige und weist alles andere zurück. OWASP empfiehlt das gegenüber dem Versuch, jede bösartige Zeichenfolge zu erkennen. Unbekannte Schemafelder zurückzuweisen

deckt Tippfehler und veraltete Clients früh auf. Die gesamte Anfrage beim ersten Problem mit einer klaren Meldung zurückzuweisen ist sicherer, als mit nur teilweise geprüften Daten fortzufahren.

Wie es in der Praxis aussieht

Eine API begann mit für jeden Endpunkt von Hand geschriebener Anfragevalidierung. Auf dem dokumentierten Implementierungsstand machte der Wechsel zu Huma v2 die Prüfungen für 649 Pfade und 760 Operationen zu Typen. Jeder Endpunkt hatte Ein- und Ausgabestrukturen. Huma erzeugte daraus die Anfragevalidierung, und aus denselben Typen entstand eine OpenAPI-Beschreibung. Zwei WebSocket-Endpunkte blieben direkt auf Fiber, weil das Modell aus Anfrage und Antwort dort nicht passte. Eine CI-Prüfung verglich die vom Server registrierten Routen mit denen der Beschreibung. Die dokumentierte Routenübereinstimmung betrug 100 %; die in den Eingabetypen deklarierten Prüfungen ersetzten die entsprechende handgeschriebene Validierung.

Konfiguration verlangt dieselbe Behandlung. Die SSH-Härtung auf dem firmeneigenen Host durchläuft heute bei jedem Abgleich drei Validierungsstufen. Die Kandidatendatei wird vor der Installation auf Syntax geprüft, die zusammengesetzte Konfiguration nach der Installation. Danach werden die vom Daemon selbst aufgelösten Einstellungen zurückgelesen und 24 Direktiven dagegen geprüft. Die dritte Stufe entstand, weil ein Zugriffsbericht im September 2026 zwei Einstellungen fand, die die Automatisierung bei jedem Abgleich geschrieben und der Daemon überschrieben hatte: Die eigenen Vorgaben des Betriebssystems wurden zuerst gelesen, und bei diesen beiden Einstellungen gewinnt das erste Vorkommen. Wie die Seite sagt, beweist die Suche nach einer Direktive nur, dass sie geschrieben wurde.

Markup-Validierung gehörte zum dokumentierten Implementierungsstand im August 2026: HTML-Validierung war eine der Prüfungen einer Commit-Schranke mit 22 Lintern. Sie hatte keine Warnstufe: Jede Diagnose war ein Fehler, und Unterdrückungen direkt im Code waren verboten. Eine ungültige Seite konnte deshalb nicht durch die Kennzeichnung des Fehlers als akzeptabel committet werden.

Wo es schiefgeht

Der Server vertraut auf die Prüfung des Browsers. Ein Formular, das im Browser eine fehlerhafte E-Mail-Adresse zurückweist, wirkt validiert. Die Anfrage kann aber ganz ohne das Formular gesendet werden. OWASP nennt rein clientseitige Prüfungen als häufige Falle: Validiert werden sollte auf dem Server. Die Browserprüfung ist ein schneller Hinweis für die eingebende Person.

Ein Schema akzeptiert alles, was es nicht erwähnt. In JSON Schema macht die Auflistung einer Eigenschaft sie weder erforderlich noch verbietet sie andere Felder. Im standardmäßig verwendeten Vokabular 2020-12 wird das Schlüsselwort `format` als Annotation gesammelt; die Formatprüfung ist standardmäßig ausgeschaltet. Dieses Schema wirkt streng:

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

Es akzeptiert `{}` , `{"email": "nope"}` und `{"emial": "a@example.com"}` . Erforderlich sind `required` und `"additionalProperties": false` sowie eine aktivierte Formatprüfung im Validator oder eine Prüfung des Formats im Code. Huma weist unbekannte Felder im Anfragekörper standardmäßig zurück, dokumentiert `readOnly` und `writeOnly` , ohne sie durchzusetzen, und ignoriert unbekannte Abfrageparameter, sofern nichts anderes festgelegt wird.

Eine Datenfassung besteht die Prüfung, verwendet wird eine andere. Ein Validator, der die rohe Anfrage liest, während der Handler eine dekodierte Kopie verwendet, oder ein Test, der die Datei auf der Festplatte prüft, während der Daemon eine andere Einstellung anwendet, hat etwas geprüft, das das System nie verwendet. OWASP rät: „Validieren Sie die Darstellung, die tatsächlich verwendet wird.“ Die obige SSH-Arbeit zeigt dieselbe Lehre für Konfiguration: Erst das Zurücklesen vom Daemon zeigte, was wirksam war.

Eine Negativliste trifft auf die nächste Variante. Bekannte gefährliche Zeichen zu sperren weist legitime Eingaben wie einen Apostroph im Nachnamen zurück und lässt die Abfrage trotzdem unsicher. Für jedes Feld sollte feststehen, was es annimmt; alles andere ist zurückzuweisen.

Gültige Eingaben gelten als sichere Eingaben. Eine Zeichenfolge, die jede Regel bestanden hat, kann immer noch eine SQL-Anweisung oder eine Seite zerstören, wenn sie dort eingefügt wird. Validierung begrenzt das Eingehende; parametrisierte Abfragen und Ausgabekodierung schützen das Ziel. Beides ist erforderlich.

Weiterführende Lektüre

- OWASP Input Validation Cheat Sheet — Syntax und Semantik, Positivlisten, Grenzen beim Parsen und häufige Fallen.
- Dokumentation des W3C Markup Validators — die eigene Darstellung der Markup-Validierung mit Hilfeseiten dazu, was Gültigkeit beweist und was nicht.
- JSON Schema Validation, Entwurf 2020-12 — die Validierungsschlüsselwörter, einschließlich des voreingestellten Verhaltens von `format`.
- Huma: Request Validation — aus Go-Struct-Tags erzeugte Validierung mit strengen und lockeren Feldeinstellungen.
- OpenAPI Specification — das Beschreibungsformat, dessen Schemata viele API-Validatoren durchsetzen.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Bringen Sie einen repräsentativen Datensatz, die Fragen, die er beantworten muss, und den aktuellen Engpass mit. Wir können vereinbaren, wie Qualität, Durchsatz und Wiederherstellung gemessen werden.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Validierung?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/validation/