Engineer

Валідація

перевірка вхідних даних за правилами, перш ніж їм довіряти

Що це таке

Валідація — це крок, на якому програма порівнює вхідні дані з правилами, яких ці дані мають дотримуватися, і відхиляє їх, якщо порушено правила, ще до того, як щось буде збережено, виконано чи передано далі. Залежно від рівня її називають по-різному: валідація введення для полів форм і запитів до API, валідація за схемою, коли правила записано як JSON Schema або опис OpenAPI, і валідація розмітки, коли вебсторінку перевіряють за граматикою HTML. Дані, що пройшли перевірку, називають валідними.

Настанови OWASP ділять цю роботу на два питання. Синтаксис визначає, чи має значення очікуваний тип і форму: чи це дата, чи це ціле число, чи розбирається тіло запиту як JSON. Семантика визначає, чи має значення сенс для конкретної операції: бронювання готелю потребує двох коректних дат, а ще й того, щоб дата виїзду була пізнішою за дату заїзду. Перетворення рядка «500» на ціле число відповідає на перше питання і нічого не каже про те, чи прийнятне проживання на 500 ночей.

Валідація розмітки — найстаріша публічна форма цієї ідеї у вебі. Сервіс W3C Markup Validation Service описує валідацію як перевірку документа за граматикою, яку той сам декларує, і порівнює свій валідатор із lint для C. Документація сервісу обережно пояснює, що саме це доводить: валідна сторінка не обов’язково добра, це лише сторінка, яку інструмент визнав відповідною певному набору правил. Та сама обережність стосується всіх інших різновидів. Валідація показує, що дані відповідають написаним правилам, і нічого не каже про те, чи були ці правила слушними.

Із валідацією часто плутають два сусідні поняття. Санітизація змінює вхідні дані, щоб зробити їх прийнятними, вирізаючи чи переписуючи частини, тоді як валідація приймає або відхиляє дані такими, як вони є. Екранування й

параметризація роблять дані безпечними для місця, куди вони потрапляють, — сторінки HTML чи запиту SQL, — і вони потрібні й після того, як валідацію пройдено.

Застосуйте поняття до рішення

Застосуйте «Що таке валідація?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Правила й перевірки, що відсіюють погані дані ще на вході, і чітка відповідь, хто володіє якою цифрою і звідки вона взялася.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Один інфраструктурний план для 14 підрозділів — 14 дослідницьких підрозділів, охоплених спільним інфраструктурним планом

Один інфраструктурний план для 14 підрозділів Зв'язатися з нами →

Чому це важливо

Рішення, що стоїть за валідацією, — де система проводить межу довіри і що на цій межі перевіряє. Кожен запит із браузера, кожне повідомлення з черги, кожен файл від партнера і кожна зміна конфігурації від колеги приходять зі своїми припущеннями. Якщо перевірити їх на вході, решта коду може на них покладатися; якщо перевірку пропустити, кожна функція глибше в системі мусить захищатися сама, а на практиці більшість цього не робить.

Друге рішення — як записувати правила. Перевірки, написані вручну для кожного ендпоінта окремо, спершу дешеві, але з ростом системи розходяться між собою, бо кожна з них — окремий шматок коду. Правила, оголошені один раз як типи чи як схема, може застосовувати бібліотека, а документацію можна публікувати з того самого джерела, що зменшує ризик розходження між описом і поведінкою. Ціна — фреймворк, який треба вивчити, і типові налаштування, які треба розуміти, бо схема застосовує рівно те, що в ній сказано, і жодного правила понад це.

Останній вибір — суворість. Список дозволеного (allowlist) описує прийнятне і відхиляє все інше; OWASP радить саме цей підхід замість спроб розпізнати кожен зловмисний рядок. Відхилення полів, яких схема не знає, рано виявляє одруківки й застарілих клієнтів. Відхилити весь запит на першій же проблемі, з чіткою помилкою, безпечніше, ніж продовжувати з даними, перевіреними лише частково.

Як це виглядає на практиці

Один API починався з валідації запитів, написаної вручну для кожного ендпоінта. На задокументованому етапі реалізації перехід на Huma v2 перетворив ці перевірки на типи для 649 шляхів і 760 операцій. Кожен ендпоінт мав вхідні та вихідні структури, Huma генерувала з них валідацію запитів, а опис OpenAPI виходив із тих самих типів. Два ендпоінти WebSocket лишилися на чистому Fiber, бо модель запиту й відповіді там не підходила. Перевірка в CI порівнювала маршрути, які реєстрував сервер, із тими, що їх декларував опис. Зафіксований паритет маршрутів становив 100%; перевірки, оголошені у вхідних типах, замінили відповідну валідацію, написану вручну.

Конфігурація потребує того самого, і посилення захисту SSH на хості компанії тепер проходить три етапи перевірки під час кожного застосування конфігурації. Файл-кандидат перевіряють на синтаксис перед встановленням, зібрану конфігурацію перевіряють після встановлення, а потім зчитують фактичні параметри, які застосовує демон, і звіряють із ними 24 директиви. Третій етап з’явився тому, що звіт про доступ у вересні 2026 року показав два параметри, які автоматизація записувала під час кожного застосування конфігурації, а демон перекривав: власні усталені значення операційної системи читалися першими, а для цих двох параметрів перемагає перше входження. Як сказано на сторінці, пошук директиви доводить лише те, що її було записано.

Валідація розмітки входила до задокументованого етапу реалізації в серпні 2026 року: перевірка HTML входила до набору з 22 лінтерів, що виконувалися перед комітом без рівня попереджень, де кожна діагностика була помилкою, а локальні вимкнення перевірок у коді були заборонені, тож невалідну сторінку не можна було закомітити, просто позначивши помилку як прийнятну.

Де виникають проблеми

Сервер довіряє тому, що вже перевірив браузер. Форма, яка відхиляє неправильну адресу електронної пошти в браузері, створює враження валідації, але запит можна надіслати й оминаючи форму. OWASP називає перевірки лише на боці клієнта поширеною пасткою: валідувати слід на сервері, а перевірку в браузері вважати швидкою підказкою для людини, що заповнює форму.

JSON-схема приймає все, про що в ній не сказано. У JSON Schema перелічена властивість ще не стає обов'язковою і не забороняє інших полів, а за типовим словником 2020-12 ключове слово `format` збирається як анотація, і перевірку формату типово вимкнено. Ця схема виглядає суворою:

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

Вона приймає `{}` , `{"email": "nope"}` і `{"emial": "a@example.com"}` . Варто додати `required` і `"additionalProperties": false` та ввімкнути перевірку формату у валідаторі або перевіряти формат у коді. Huma зі свого боку типово відхиляє невідомі поля в тілі запиту, документує `readOnly` і `writeOnly` , але їх не застосовує, і пропускає невідомі параметри рядка запиту, доки це окремо не налаштовано.

Перевірка проходить на одній версії даних, а програма використовує іншу. Валідатор, що читає сирий запит, тоді як обробник читає декодовану копію, або тест, що дивиться на файл на диску, тоді як демон застосовує інший параметр, перевірили те, чим система не користується. OWASP радить валідувати те подання, яке справді буде використано. Робота із SSH вище — той самий урок для конфігурації: лише зчитування з демона показало, що діє насправді.

Список заборон зустрічає наступний варіант. Блокування відомих небезпечних символів відкидає цілком законні дані, скажімо апостроф у прізвищі, і водночас залишає запит до бази даних небезпечним. Слід визначити, що приймає кожне поле, і відхиляти решту.

Валідні дані вважають безпечними. Рядок, що пройшов кожне правило, однаково може зламати інструкцію SQL чи сторінку, коли його туди вставлять. Валідація звужує те, що надходить; параметризовані запити й екранування виводу захищають місце, куди дані потрапляють, і потрібні обидва заходи.

Що почитати далі

- OWASP Input Validation Cheat Sheet — синтаксис і семантика, списки дозволеного, межі розбору й типові пастки.
- W3C Markup Validator documentation — власний опис валідації розмітки від валідатора, зі сторінками довідки про те, що валідність доводить і де її межі.
- JSON Schema Validation, draft 2020-12 — ключові слова валідації, зокрема типова поведінка `format` .
- Huma: Request Validation — валідація з тегів структур Go, із суворим і м’яким режимами полів.
- OpenAPI Specification — формат опису, схеми якого застосовує багато валідаторів API.

Словник

Обирайте за фактичними умовами

Надайте репрезентативний набір даних, запитання до нього та поточне вузьке місце. Узгодимо вимірювання якості, пропускної здатності та відновлення.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке валідація?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/validation/