Engineer

Validation

vérifier les entrées selon des règles avant de s'y fier

Ce que c’est

La validation est l’étape où un programme compare les données entrantes aux règles qu’elles sont censées suivre et les refuse si elles ne les respectent pas, avant de stocker, d’exécuter ou de transmettre quoi que ce soit. Son nom varie selon la couche : validation des entrées pour les champs de formulaire et les appels d’API, validation de schéma lorsque les règles sont écrites dans un JSON Schema ou une description OpenAPI, validation du balisage lorsqu’une page web est vérifiée selon la grammaire de HTML. Une entrée qui passe le contrôle est dite valide.

Les recommandations d’OWASP divisent le travail en deux questions. La syntaxe demande si une valeur a le type et le format attendus : est-ce une date, un entier, ce corps peut-il être analysé comme JSON ? La sémantique demande « si la valeur a du sens pour l’opération » : une réservation d’hôtel a donc besoin de deux dates valides et d’une date de fin postérieure au début. Convertir la chaîne “500” en entier règle la première question et ne dit rien de l’acceptabilité d’un séjour de 500 nuits.

La validation du balisage est la plus ancienne forme publique de cette idée sur le Web. Le W3C Markup Validation Service la décrit comme le contrôle d’un document selon la grammaire qu’il déclare utiliser, et compare son validateur à lint pour C. Sa documentation précise prudemment ce que cela prouve : une page valide n’est pas nécessairement une bonne page, seulement une page qu’un outil a reconnue conforme à un ensemble donné de règles. La même prudence s’applique à toute autre forme. La validation indique que les données respectent les règles écrites ; elle ne dit pas que ces règles étaient les bonnes.

Deux notions voisines sont souvent confondues avec elle. L'assainissement modifie l'entrée pour la rendre acceptable, en supprimant ou réécrivant certaines parties, tandis que la validation l'accepte ou la rejette telle quelle. Le codage et la paramétrisation rendent les données sûres pour leur destination, une page HTML ou une requête SQL, et restent nécessaires après une validation réussie.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que la validation ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des règles et des contrôles qui détectent les données incorrectes dès leur arrivée, avec une réponse claire sur le responsable et l'origine de chaque chiffre.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Un plan d'infrastructure pour 14 départements — 14 départements de recherche sur un même plan d'infrastructure

Un plan d'infrastructure pour 14 départements Contactez-nous →

Pourquoi c'est important

La validation impose de décider où un système trace sa frontière de confiance et ce qu'il y vérifie. Chaque requête du navigateur, chaque message d'une file, chaque fichier d'un partenaire et chaque modification de configuration d'un collègue arrive avec des hypothèses. Les vérifier à la frontière permet au reste du code de s'y fier ; omettre le contrôle impose à chaque fonction en aval de se défendre seule, et en pratique la plupart ne le feront pas.

La deuxième décision est la façon d'écrire les règles. Des contrôles écrits à la main pour chaque point d'accès coûtent peu au début puis divergent lorsque le système grandit, car chacun est un petit morceau de code distinct. Des règles déclarées une fois comme types ou comme schéma peuvent être imposées par une bibliothèque et publiées comme documentation depuis la même source : cela réduit le risque que la description et le comportement divergent. En contrepartie, il faut apprendre à utiliser un framework et comprendre ses réglages prédéfinis, car un schéma impose exactement ce qu'il dit, rien de ce qu'il laisse sous-entendu.

La rigueur est le dernier choix. Une liste d'autorisation décrit ce qui est acceptable et rejette le reste, ce qu'OWASP recommande plutôt que de tenter de reconnaître toutes les chaînes malveillantes. Rejeter les champs inconnus d'un

schéma révèle tôt les fautes de frappe et les clients obsolètes. Rejeter toute la requête dès le premier problème, avec une erreur claire, est plus sûr que de continuer avec des données partiellement vérifiées.

À quoi cela ressemble en pratique

Une API a commencé avec une validation de requête écrite à la main pour chaque point d'accès. Au stade d'implémentation documenté, le passage à Huma v2 a transformé les contrôles en types sur 649 chemins et 760 opérations. Chaque point d'accès avait des structures d'entrée et de sortie, Huma produisait la validation des requêtes à partir d'elles et une description OpenAPI provenait des mêmes types. Deux points d'accès WebSocket sont restés directement sur Fiber, où le modèle requête-réponse ne convenait pas. Un contrôle en CI comparait les routes enregistrées par le serveur à celles annoncées par la description. La parité des routes consignée était de 100 % ; les contrôles déclarés dans les types d'entrée ont remplacé la validation correspondante écrite à la main.

La configuration a besoin du même traitement, et le durcissement de SSH sur l'hôte de l'entreprise exécute désormais trois étapes de validation à chaque convergence. Le fichier candidat subit un contrôle de syntaxe avant son installation, la configuration assemblée est vérifiée après l'installation, puis les réglages résolus par le démon lui-même sont relus et 24 directives sont vérifiées par rapport à eux. La troisième étape existe parce qu'un rapport d'accès de septembre 2026 a trouvé deux réglages que l'automatisation écrivait à chaque convergence et que le démon remplaçait : les valeurs par défaut du système d'exploitation étaient lues d'abord et, pour ces deux réglages, la première occurrence l'emporte. Comme le dit la page, chercher une directive avec grep prouve seulement qu'elle a été écrite.

Au stade d'implémentation documenté d'août 2026, la validation HTML faisait partie d'un contrôle qualité obligatoire de commit à 22 linters sans niveau d'avertissement, où chaque diagnostic était une erreur et les suppressions directement dans le code étaient interdites. Une page invalide ne pouvait donc pas être intégrée à un commit en déclarant l'erreur acceptable.

Où les problèmes surviennent

Le serveur fait confiance au contrôle du navigateur. Un formulaire qui refuse une mauvaise adresse électronique dans le navigateur semble validé, mais la requête peut être envoyée sans le formulaire. OWASP classe les contrôles

uniquement côté client parmi les pièges courants : il faut valider côté serveur et traiter le contrôle du navigateur comme une indication rapide pour la personne qui saisit.

Un schéma accepte tout ce qu'il ne mentionne pas. En JSON Schema, lister une propriété ne la rend pas obligatoire et n'interdit pas d'autres champs. Avec le vocabulaire 2020-12 utilisé par défaut, le mot-clé `format` est recueilli comme annotation et la vérification du format est désactivée par défaut. Ce schéma semble strict :

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

Il accepte `{}` , `{"email": "nope"}` et `{"emial": "a@example.com"}` . Il faut ajouter `required` et `"additionalProperties": false` , puis activer l'assertion de format dans le validateur ou vérifier le format dans le code. Huma rejette par défaut les champs inconnus du corps, documente `readOnly` et `writeOnly` sans les imposer et ignore les paramètres de requête inconnus sauf instruction contraire.

Le contrôle réussit sur une version des données et le programme en utilise une autre. Un validateur qui lit la requête brute tandis que le gestionnaire lit une copie décodée, ou un test qui examine le fichier sur disque tandis que le démon applique un autre réglage, a vérifié quelque chose que le système n'utilise jamais. Le conseil d'OWASP est : « Valider la représentation qui sera réellement utilisée. » Le travail SSH ci-dessus illustre la même leçon pour la configuration : seule la relecture des réglages du démon a révélé ce qui était en vigueur.

Une liste d'interdiction rencontre la variante suivante. Bloquer les caractères dangereux connus rejette des entrées légitimes, comme une apostrophe dans un nom de famille, et laisse pourtant la requête vulnérable. Il faut définir ce que chaque champ accepte et rejeter le reste.

Des entrées valides sont considérées comme sûres. Une chaîne qui a passé toutes les règles peut encore casser une instruction SQL ou une page lorsqu'elle y est insérée. La validation limite ce qui arrive ; les requêtes paramétrées et le

codage de sortie protègent la destination. Les deux sont nécessaires.

Pour aller plus loin

- OWASP Input Validation Cheat Sheet — syntaxe et sémantique, listes d'autorisation, limites d'analyse et pièges courants.
- Documentation du W3C Markup Validator — la présentation de la validation du balisage par le validateur lui-même et ses pages d'aide sur ce que la validité prouve ou non.
- JSON Schema Validation, projet 2020-12 — les mots-clés de validation, dont le comportement de `format` par défaut.
- Huma : Request Validation — la validation produite à partir des tags de structures Go, avec les réglages stricts et permissifs des champs.
- OpenAPI Specification — le format de description dont de nombreux validateurs d'API imposent les schémas.

Glossaire

Choisissez selon vos contraintes réelles

Présentez un jeu de données représentatif, les questions auxquelles il doit répondre et le goulot d'étranglement actuel. Nous pouvons convenir de la manière de mesurer la qualité, le débit et la reprise.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que la validation ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/validation/