SecHack365

分散アーキテクチャにおける 統一的なデータバリデーションを実現するミドルウェア Open-VE

開発駆動コース 仲山ゼミ 渋谷和樹（しぶたに かずき）

shibukazu/open-ve @s_k_526

背景: ウェブアプリ開発における分散化の流れ

- マイクロサービスやサーバレスをはじめとしてアプリケーションが複雑化
- 認証認可などセキュリティ上重要なロジックの分散・実装漏れが問題に
- Zanzibar[1]をはじめとした中央集権的なアプローチが一定の成果を上げる
- 本プロジェクトでは重要なロジックとしてバリデーションに着目

バリデーションとは

- ユーザーの入力を検証する処理の総称
- 開発者が意図しない入力を弾くことで、各種インジェクション攻撃などを防ぐ

（例）: フロントエンドにおいて、長すぎる文字列の入力を防ぐバリデーション

```
if (input.name.length > 100) {
        throw new Error(“name is too long”)
} else {
        fetch.post("https://api.example.com", input);
}
```

Open-VEの概要

バリデーションロジックの中央集権管理

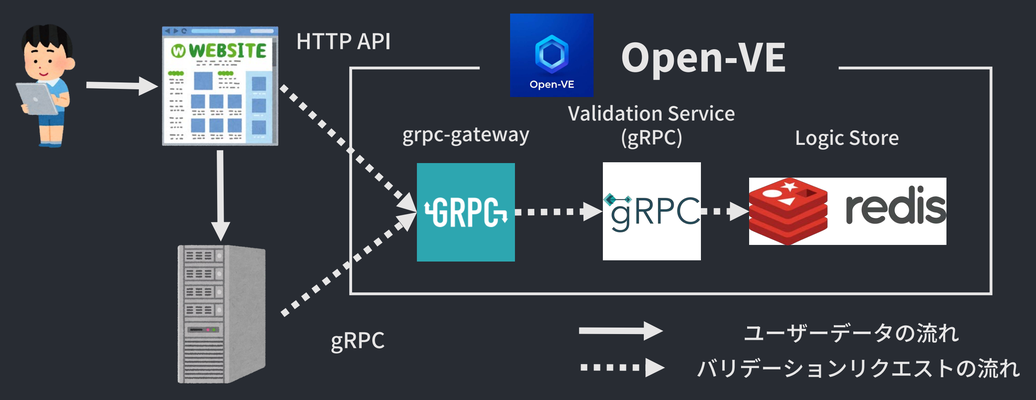
- これまでは分散システムの各レイヤーごとにバリデーションロジックを実装
- 重複実装や実装漏れ、ミスマッチが問題に
- Open-VEでは全レイヤーのロジックを一箇所で管理可能

言語非依存のロジック管理

- CEL[2]を用いたロジック記述により、言語に依存せずロジックを記述可能

API経由でのバリデーション実行

- 言語に依存しないバリデーション呼び出し方法を提供
- 実装のミスマッチを確実に防ぐことが可能に



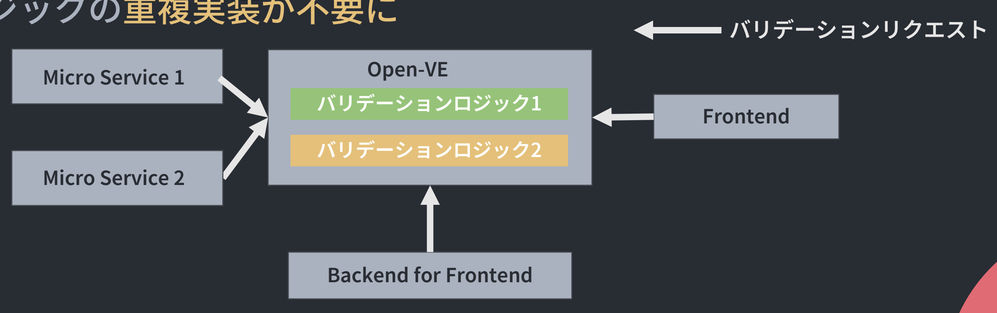
中央集権型 バリデーションロジック管理

これまでのバリデーション

バリデーションロジックをレイヤーごとに管理・実装
同じロジックを複数のレイヤーに記述しなくてはならない

これからのバリデーション

すべてのバリデーションロジックをOpen-VEで一括管理
ロジックの重複実装が不要に



APIでのバリデーション問い合わせ

リクエスト

```
curl --request POST \
  --url http://localhost:8080/v1/check \
  --data '{
  "validations": [
    {
      "id": "item",
      "variables": {
        "price": -100,
        "image": "iVB0Rw0kGgoAAASasaGYGfdGdH"
      }
    }
  ]
}'
```

負の商品価格

359バイトの画像

レスポンス

```
{
"results": [
  {
    "id": "item",
    "isValid": false,
    "message": "failed validations: price > 0"
  }
]
}
```

商品価格に関するロジックに違反

VS コード生成

protovalidateをはじめとしたバリデーションコードの自動生成アプローチが存在

Pros

- 言語ごとの個別サポートが不要
- 常に最新のバリデーションロジックを提供できる

Cons

- バリデーションごとにネットワークリクエストが発生する

→ パフォーマンスを高めることでこの問題に対処

4 Key Features

Open-VEスキーマの自動生成

Open APIスキーマからOpen-VEスキーマの一部を自動生成

```
{
"paths": {
  "/item": {
     "parameters": [
{
"id": { "type": "integer", "format": "int32" },
"name": { "type": "string" },
"price": { "type": "number", "format": "float"}
}
     ]
  }
}
```

→

```
validations:
- id: SampleObject
  cels: []
  variables:
  - name: SampleObject.id
    type: int
  - name: SampleObject.name
    type: string
  - name: SampleObject.price
    type: double
  testCases: []
```

Open-VEスキーマの単体テスト

Open-VE CLIを用いてOpen-VEスキーマの記述の正しさを検証
テスト駆動的なスキーマ記述が可能に

```
validations:
- id: "price"
  cels:
  - "number > 0"
  variables:
  - name: "number"
    type: "int"
  testCases:
  - name: "invalid price"
    variables:
    - name: "number"
      value: 0
    expected: false
```

→

```
testing open-ve schema filePath=dsl.yml
✅ PASS     : price
Results: 1 passed, 0 failed, 0 not found
```

GoとTypeScript向けの簡易SDKの提供

わずか数行のコードでバリデーション問い合わせが可能に

```
npm install open-ve-typescript-sdk
```

```
go add github.com/shibukazu/open-ve-go-sdk
```

セキュリティ機能

SSL通信のサポート
機密性の高いバリデーションロジックの盗聴を防ぐために
SSL通信をサポート

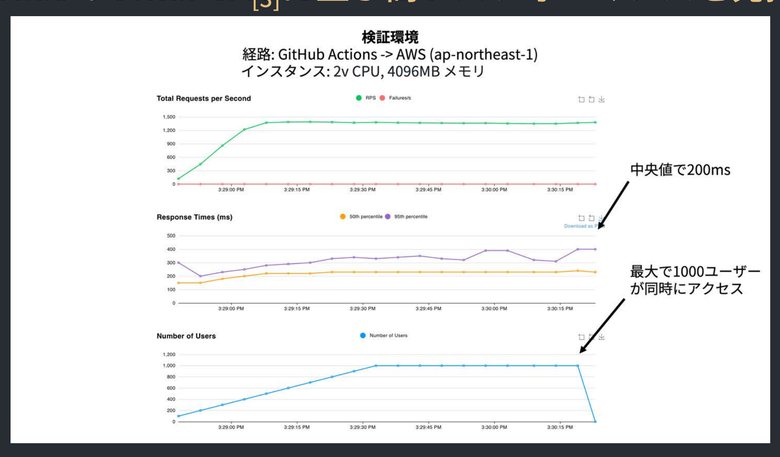
Open-VEへのアクセス認証
バリデーションロジックの不正な書き換えを防ぐために
事前共有キーを用いたアクセス認証をサポート

マスタースレーブモードのサポート

バリデーションリクエストの分散管理を可能に
マイクロサービスなどの分散アーキテクチャとの相性を向上

高いパフォーマンス

実際の利用環境を想定したE2Eパフォーマンステストを実施
類似SaaSのOktaFGA[3]に並ぶ高いパフォーマンスを発揮



高いレベルの開発者サポート

セキュリティとパフォーマンス

[1] https://research.google/pubs/zanzibar-googles-consistent-global-authorization-system/
[2] https://github.com/google/cel-spec
[3] https://www.okta.com/jp/products/fine-grained-authorization/

NICT National Institute of Information and Communications Technology