Toward Dependency-Aware API Gateways*

Saman Barakat[1 [0000−0002−7714−3742]], Ana B. Sánchez[1 [0000−0003−1473−0955]], and Sergio Segura[1 [0000−0001−8816−6213]]

SCORE Lab, I3US Institute, Universidad de Sevilla, Seville, Spain
saman.barakat@gmail.com

Abstract. Web APIs often include inter-parameter dependencies that constrain how input parameters can be combined to form valid calls to the service. API requests violating one or more of these dependencies result in an unnecessary exchange of messages, causing a waste of time and user quota. Also, dependencies are often not correctly checked by the servers, resulting in critical failures or uninformative error responses. In this paper, we propose extending API gateways to support the detection and explanation of inter-parameter dependencies violations. To achieve this goal, we leveraged IDL4OAS, an OAS extension for describing the dependencies among input parameters in web APIs, and IDLReasoner, a constraint-based IDL reasoner. Both were integrated into a prototype tool using Spring Cloud Gateway. Preliminary evaluation results on five industrial API operations show that our approach can successfully detect and explain all invalid requests, reducing the response time by around 80.31% and minimizing potential input validation failures.

Keywords: API Gateway · inter-parameter dependencies · web APIs

1 Introduction

Web APIs frequently include dependencies between input parameters that restrict the valid combinations of parameters for making calls to the service. These relationships between input parameters are known as inter-parameter dependencies (or simply dependencies for short). In the Yelp API, for example, the parameter `location` is required if either `latitude` or `longitude` is not provided, and both parameters are required if the `location` is not provided [9]. A recent study revealed that dependencies are extremely common in industrial web APIs: they appear in 4 out of every 5 APIs across all application domains and types of operations [4]. Motivated by the lack of support for defining dependencies in current API specification languages, Martin-Lopez et al. [3] introduced IDL4OAS, an OAS extension for describing the dependencies among input parameters in web APIs as a part of the OAS specification format, and IDLReasoner, a constraint-based Java library that enables the analysis of IDL (Interparameter Dependency

* This work has been partially supported by grants PID2021-126227NB-C22 and TED2021-131023B-C21, funded by MCIN/AEI/10.13039/501100011033 and by European Union "NextGenerationEU"/PRTR».

Language) specifications, e.g., checking whether an API call meets its dependencies. We leverage both of them in this work.

Dependency violations are common in API requests and cause wasted time and quota consumption. Also, dependencies are often not correctly checked by the servers, resulting in critical failures or uninformative error responses.

In this paper, we propose extending API gateways with analysis capabilities making them able to automatically detect and explain inter-parameter dependencies violations. To do this, we leveraged IDL4OAS and IDLReasoner, integrating them into a prototype tool using Spring Cloud Gateway. Preliminary results reveal that our solution can successfully detect and explain dependency violations, reducing the response time of invalid requests by 80%, saving unnecessary communication with the services, and minimizing potential validation failures.

2 Approach

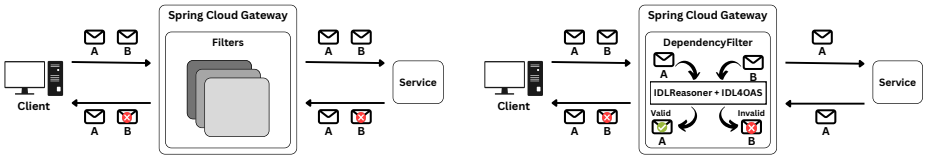
We propose extending API gateways to support detecting and explaining inter-parameter dependencies violations. In a traditional gateway, when a client makes a request to a service through the gateway, the gateway forwards the request to the target service as shown in Figure 1a. However, in our approach, when a client makes a request, the gateway analyzes the request before forwarding it to the server as shown in Figure 1b. If the request is valid, the gateway sends the request to the server. Otherwise, it returns an error explanation message to the client, thereby blocking unnecessary traffic through the services.

As an example, the OMDb API documentation describes the following dependency between parameters i (IMDb ID) and t (Movie title): "while both 'i' and 't' are optional, at least one argument is required" [6]. Using IDL4OAS, this dependency can be expressed as shown in Listing 1. Failing to adhere to this dependency by sending a request to the API without both parameters results in the API gateway forwarding the request to the service, which responds with the message "Incorrect IMDb ID." However, by applying our approach, the API gateway analyzes the request, identifies the dependency violation, and returns an explanation outlining the invalid parameters and the dependencies violated: "InvalidRequestParams=[t=null, i=null], IDLConflicts=[Or(i, t);]". More importantly, our approach blocks the invalid request, saving unnecessary message exchange and mitigating potential failures due to bugs in the validation code of the services.

```
paths:
  /:
    get:
      parameters:
        - name: i [...]
        - name: t [...]
      [...]
      x-dependencies:
        - Or(i, t);
```

Listing 1: OAS document of OMDb API extended with IDL4OAS

Our approach is based on the IDLReasoner, a Java-based library used to perform several analysis operations using OAS documents extended with IDL [3]. We implemented our approach using the Spring Cloud Gateway, a lightweight API gateway built on top of the Spring ecosystem [7]. Spring Cloud Gateway provides several built-in filters and supports the integration of custom ones. In our work, we implemented a custom filter utilizing the IDLReasoner. The filter, referred to as *dependency filter* from now on, uses IDLReasoner for detecting (and optional explaining) violations of the inter-parameter dependencies in the incoming API requests. Violations are detected by checking the API requests against the OAS specification of the service, augmented with a description of its inter-parameter dependencies using the IDL4OAS extension. Figure 1 illustrates two scenarios where a client sends requests to a service through the Spring Cloud gateway without and with our dependency filter.



(a) No dependency filter (b) Dependency filter

Fig. 1: API Gateway with (no) support for dependency detection

3 Preliminary evaluation

We conducted a preliminary evaluation by testing five real-world API operations. We use the testing tool RESTest [5] for generating 2000 random API requests, 1000 valid and 1000 invalid, and then executed them through our prototype API gateway. An API request is considered valid if it satisfies all constraints and dependencies between input parameters described in the web API. Then, we compared the response time with and without our dependency filter. The response time was measured in all cases from the time the gateway receives the request to the moment in which it returns a response to the client.

As anticipated, the API gateway permitted the valid requests to pass through with minimal delay. To be specific, when the dependency filter was not employed, the average response time for the gateway was 243 ms. However, when valid requests were processed using the dependency filter, the average response time increased slightly to 272 ms.

Table 1 shows the result of performing 1000 invalid requests for each operation using the gateway with and without the dependency filter. The results show that the average response time for all operations without the dependency filter, where invalid calls are forwarded to the target services, is 168 ms. However, our dependency filter takes 33 ms to detect invalid API requests (36 ms for detecting and explaining), avoiding forwarding the invalid requests to the services. This means a 80% reduction in response time when used for detection only, and 78% when used for validation and explanation.

Table 1: The average response time of 1000 invalid requests (ms)

Operation	No detection	Detection	Explanation
FSQ-PlaceSearch [1]	159.73	**18.70**	**22.67**
GitHub-UserRepos [2]	196.23	**74.88**	**77.29**
OMDb-ByIdTitle [6]	173.34	**24.91**	**29.26**
Tumblr-BlogLikes [8]	150.66	**26.59**	**29.26**
Yelp-BusinessSearch [9]	160.90	**20.48**	**22.67**
Mean	168.17	**33.11**	**36.23**

4 Conclusions

In this paper, we propose to extend API gateways with capabilities for detecting and explaining dependency violations between parameters in web APIs. This not only allows to reduce response time when processing invalid requests, but also to provide informative messages to users about violated dependencies. In addition, our approach serves as a shield for API servers, stopping invalid requests and preventing input validation failures. Our future plans include a more extensive evaluation and improving explanation messages.

References

1. Foursquare Places API, `https://location.foursquare.com/developer/reference/place-search`, accessed online in March 2023
2. GitHub Repositories API, `https://docs.github.com/en/rest/repos/repos#list-repositories-for-the-authenticated-user`, accessed online in April 2023
3. Martin-Lopez, A., Segura, S., Muller, C., Ruiz-Cortes, A.: Specification and automated analysis of inter-parameter dependencies in web apis. IEEE Transactions on Services Computing pp. 1–14 (2021)
4. Martin-Lopez, A., Segura, S., Ruiz-Cortés, A.: A catalogue of inter-parameter dependencies in restful web apis. In: Yangui, S., Bouassida Rodriguez, I., Drira, K., Tari, Z. (eds.) Service-Oriented Computing. pp. 399–414. Springer International Publishing, Cham (2019)
5. Martin-Lopez, A., Segura, S., Ruiz-Cortés, A.: Restest: Black-box constraint-based testing of restful web apis. In: Kafeza, E., Benatallah, B., Martinelli, F., Hacid, H., Bouguettaya, A., Motahari, H. (eds.) Service-Oriented Computing. pp. 459–475. Springer International Publishing, Cham (2020)
6. OMDb API, `https://www.omdbapi.com/`, accessed online in April 2023
7. Spring Cloud Gateway, `https://cloud.spring.io/spring-cloud-gateway`, accessed online in March 2023
8. Tumblr API - retrieve blogs likes. `https://www.tumblr.com/docs/en/api/v2`, accessed online in April 2023
9. Yelp Search businesses API, `https://docs.developer.yelp.com/reference`, accessed online in March 2023