Documentation from GraphQL schema

Author:
Alessandro Buonerba

Supervisors:
Dr Konstantin Kapinchev

A proposal submitted in fulfillment
of the requirements for the degree of
BSc (Hons) Computer Science (Cybersecurity)

Department of Computing & Mathematical Sciences
Liberal Arts & Sciences

University of Greenwich
London, United Kingdom

May 2022

PROJECT PROPOSAL

INTRODUCTION

API documentation is a very important part of the software development process, it improves the dev experience and makes it easier to integrate, improves maintability and easily enables versioning with indication on deprecated fields (Fan et al., 2021). Working with API documentation is not always straightforward, especially when the working with gateway and federations, as it would require much effort from all the developers involved in the process. Swagger makes documentation for REST-APIs very straightforward, as it automatically generates HTML based visualisation and interaction out of the box for any consumer of the API (Koren and Klamma, 2018). Since Facebook released GraphQL to the mass in 2015, many individual developers and companies are switching and converting to it to build their own APIs as it enables them to have a much more flexible and efficient way of building their APIs (Brito and Valente, 2020). GraphQL gives to the consumer only the data that he needs, solving the overfetching and underfetching problem related to the traditional REST-APIs (Wittern, Cha, and Laredo, 2018). Querying the API the way that user want, enables for so much flexibility but can also be a threat to security if not handled properly. The objective of this project is to enable API producers to build a secure GraphQL APIs without recheable introspection on endpoint while still count on a tool that can generate a framework agnostic structured documentation.

PROBLEM DOMAIN

When working on such a problem, there are many different complications as having updated documentation for GraphQL APIs, one of them - and probably the most important - being security. Introspection enables users to query a GraphQL API and discover its schema structure, giving bad actors a chance to discover potentially malicious operations (Khalil, 2021) easily and disrupt the availability of the API, but it is also a requirement for tools such as *GraphiQL* and *Playground*. This makes this a severe dilemma for producers who want to keep their APIs as secure as possible, away from indiscreet eyes, and open to potential threats but still have documentation tooling.

METHODOLOGY

EVALUATION

BIBLIOGRAPHY

Brito, Gleison and Marco Tulio Valente (Mar. 2020). "REST vs GraphQL: A Controlled Experiment". In: *2020 IEEE International Conference on Software Architecture (ICSA)*, pp. 81–91. DOI: 10.1109/ICSA47634.2020.00016.

Fan, Qiang, Yue Yu, Tao Wang, Gang Yin, and Huaimin Wang (Jan. 2021). "Why API Documentation Is Insufficient for Developers: An Empirical Study". In: *Science China Information Sciences* 64.1, p. 119102. ISSN: 1674-733X, 1869-1919. DOI: 10.1007/s11432-019-9880-8. URL: http://link.springer.com/10.1007/s11432-019-9880-8 (visited on 10/09/2021).

Khalil, Stemmler (2021). *Why You Should Disable GraphQL Introspection In Production – GraphQL Security*. URL: https://www.apollographql.com/blog/graphql/security/why-you-should-disable-graphql-introspection-in-production/ (visited on 10/09/2021).

Koren, István and Ralf Klamma (2018). "The Exploitation of OpenAPI Documentation for the Generation of Web Frontends". In: *Companion Proceedings of the the Web Conference 2018*. International World Wide Web Conferences Steering Committee. ISBN: 978-1-4503-5640-4. DOI: 10.1145/3184558.3188740. URL: https://doi.org/10.1145/3184558.3188740.

Wittern, Erik, Alan Cha, and Jim A. Laredo (2018). "Generating GraphQL-Wrappers for REST(-like) APIs". In: *Web Engineering*. Ed. by Tommi Mikkonen, Ralf Klamma, and Juan Hernández. Cham: Springer International Publishing, pp. 65–83. ISBN: 978-3-319-91662-0.