**RESTest: automated black-box testing of RESTful web APIs**

<https://dl.acm.org/doi/10.1145/3460319.3469082>

Brînză Alina-Elena, Apăvăloaei Alexandru-Teodor, Astăluș Adrian-Claudiu

Web APIs have become an essential part of any modern application. Generally, they adhere to the Representational State Transfer (REST) architectural style, thus the name of RESTful APIs. These APIs are described usually using the OpenAPI Specification (OAS), which offers a structured description of a web API in a way that people can understand and discover the capabilities of the API without the need of accessing source code or additional documentation. Even though their frequent use, these APIs can sometimes prove to be faulty. In these situations, they can have a big impact on any application that makes use of them. Therefore, there is a need to constantly test these APIs in order to ensure they do not become more of an impediment for developers, rather than a helpful tool. At the moment, the process of automating API testing is rapidly progressing, but there are still certain elements to clarify and perfect. While current approaches show promising results in the automated detection of bugs, their efectivness is limited for APIs that might require complex input data or data that has to satisfy certain constraints (for example, the YouTube API, or the PayPal API).

The RESTest framework intends to provide a solution for automated black-box testing of RESTful APIs that comply with the OAS format. RESTest can be broken down into more components. Firsly, the framework has a test data generator. Instead of using random values, which provide little significant insight into an API, RESTest automatically generates realistic parameter values using strategies as the reusing of values observed in previous API responses, the extraction of values from knowledge bases (such as BDpedia, a semantic data generator), or the leverage of manually defined generators (strings conforming regular expressions) or data dictionaries. Then, there is the test case generation part of the framework, where test cases are generated (a test case represents a single API call and a set of assertions on the call response) with the use of test case generation strategies, such as fuzzing, adaptive random testing and constraint-based testing. These thest cases can be converted into platform-specific, ready-to-execute test cases for either Postman of REST Assured. The entire testing process can be automated, as the framework allows for test runs both online and offline. For example, test cases can be generated once and then run multiple times as part of regression testing test cases which, in offline mode, can be performed on different machines at different times. In the online mode, the opportunity of continuous API testing appears, as test cases can be generated and executed constantly, in an interleaved way.

In order to validate the framework, two tests have been conducted: one in offline mode, and one online. Offline, three RESTful services with inter-parameter dependencies have been tested. For every service, two thousand test cases have been automatically generated using both constraint-basd and random generators. At the end of these runs, the number of failures uncovered has been used as a metric to determine the efficiency of the framework, failures consisting of 5XX or 4XX status codes in response to valid inputs. The constraint-based generation was proven to be more effective in discovering failures, thus putting into light disconformities with the OAS specification and incorrect handling of valid and invalid inputs in the used APIs. The online testing has the same aim. Over a period of five days, 15 RESTful services of 7 popular APIs have been continuously tested. In the end, out of more than 90.000 test cases, approximately 30% of them have uncovered failures in the tested APIs. The main reasons for these failures were the incorrect specification of inter-parameter dependencies, while discrepancies between the API specification and the actual implementation were found as well.