

Enhancing White-Box Search-Based Testing of RESTful APIs

Amid Golmohammadi
Kristiania University College
Oslo, Norway
amid.golmohammadi@kristiania.no

Abstract—Testing RESTful APIs which today are considered as backbone of the Internet is of great importance. The aim of this project is to effectively perform white-box testing of RESTful APIs by using search-based methods. We first conducted a systematic literature review to analyze the current state-of-the-art work on testing of RESTful APIs and summarize research challenges in this area. This could lead us further to the next steps which are enabling white-box heuristics for testing .NET REST APIs, conducting parameter tuning on search-based REST API testing approaches and analyzing different search algorithms for white-box test generation of REST APIs.

Index Terms—REST API, testing, test case generation, fuzzing, white-box testing, SBST

I. INTRODUCTION

RESTful APIs [1] are frequently chosen for developing the backends for web and enterprise applications, especially in microservice architectures [2]. They are not only utilized by numerous businesses (e.g., Twitter¹ and LinkedIn²) for their backends, but also widely available online and offer a wide range of functionalities.

Many approaches for generating test cases for RESTful APIs have emerged in recent years [3]–[8]. These approaches can be divided into black-box and white-box categories. Unlike black-box, white-box strategies make use of the API's source code to produce test cases that aim at maximizing code coverage while attempting to reduce the amount of effort required from the users [9].

In this PhD project, we are going to enhance white-box testing of RESTful APIs by taking advantage of search-based algorithms. The idea behind *Search-Based Software Testing* (SBST) is to cast software testing problems into optimization problems that can be solved by the aid of a search algorithm [10], [11]. By applying this approach, the process of software testing can be automated which is less expensive and more effective than manual testing.

As a first step in our research, we conducted a systematic literature review on 92 research articles in the domain of RESTful API testing [12]. This helped us to review the current state-of-the-art, discover research gaps that we can fill, and

identify challenges to be solved in testing RESTful APIs. For example, we found out that 72% of the studies were focusing on black-box testing. This shows that less attention has been paid to white-box testing in this context. Another finding was that *Having more REST case studies* is mentioned in almost one-seventh of the papers as an open challenge to be addressed in the future. *Improving white-box heuristics* was also mentioned by 4 papers which is significant by considering the few amount of papers focusing on white-box testing (i.e., 15 papers). White-box testing aims at examining all code paths, which could lead to higher code coverage. Additionally, this approach allows for insights into the code structure, facilitating code optimization efforts [13].

II. BACKGROUND

Before explaining the objective of this research, we are going to shed some light on relevant terms in this area.

A. Software Testing

Software testing is a process or a series of processes, designed to make sure computer code does what it was designed to do and that it does not do anything unintended [14]. Not having proper tests for software could not only cost a huge amount of loss, but also could be life threatening. Following Tesla's most recent update on October 2021, Tesla started getting complaints from consumers that their cars had mistakenly detected threats of forward collision, triggering the automated emergency braking (AEB) system and bringing the car to an abrupt stop [15]. The issue was found to be caused by a bug in the Full-Self Driving beta software which made this company to recall about 12,000 vehicles. This shows that applications can fail due to software defects, which can also damage businesses' reputations and incur millions of dollars to rectify.

Black-box testing concentrates on functionality and external behavior, while white-box testing examines the software's internal logic and code. Black-box testing treats the program as a "black box", while white-box testing looks "inside the box". Both techniques can be helpful for assessing the quality of software. There is not any universal strategy and both black-box and white-box and each has its own merits and drawbacks. However, white-box testing makes it possible to thoroughly cover the internal logic, paths, and structure of the software being tested. A higher level of code coverage can be achieved

This project is a part of EAST, which is a larger project funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (EAST project, grant agreement No. 864972).

¹<https://developer.twitter.com/en/docs>

²<https://learn.microsoft.com/en-us/linkedin>

since testers can observe the code and make sure that all branches, conditions, and statements are tested.

B. Search-Based Software Testing

Search-Based Software Testing (or SBST) refers to the automation or partial automation of a testing task, such as the automatic production of test data, using a meta-heuristic optimal search technique, such as a *Genetic Algorithm* [16]. In other words, Search-Based Software Testing is where biology meets software testing. The idea is to cast software testing problems into optimization problems that can be solved by the aid of a search algorithm. By applying this approach, the process of software testing can be automated which is less expensive and more effective than manual testing.

C. REST APIs

An API (Application Programming Interface) serves as a way of communication between two or more computer programs [17]. Many modern APIs are following *REST* which stands for Representational State Transfer. REST is an architectural style which was introduced by Fielding in 2000 [1]. It is not a protocol, as it only defines a set of guidelines for designing APIs for accessing and manipulating resources using *HTTP* protocol over the network. HTTP (Hypertext Transfer Protocol) is an application-layer protocol for hypermedia information systems that are distributed and collaborative over a computer network [18].

III. OBJECTIVES OF THE RESEARCH PROJECT

The purpose of this PhD study is to enhance the effectiveness of white-box testing for RESTful APIs using evolutionary techniques. Instead of starting from scratch, to bootstrap our research work, the prototypes for our novel techniques will be integrated into EVOMASTER [4]. To the best of our knowledge, EVOMASTER is the only open-source fuzzer for automatically generating test cases for RESTful APIs that supports both black-box and white-box testing.

To achieve our goal, in this PhD, we aim at answering the following research questions:

- What are the existing challenges in testing REST APIs?
- How to best carry out automated white-box testing for REST APIs?
- What is the impact of parameter tuning on the outcome of automated test generation of REST APIs?
- How do different search algorithms tackle the problem of automated test generation for REST APIs?

IV. METHODS

In this section, the different stages of our research is mapped out. The first stage is to conduct a systematic literature review on RESTful APIs testing. This could help us to understand the start-of-the art work and identify problems to solve. In addition, to better allow white-box testing of REST APIs, one step of our research would be enabling white-box heuristics for APIs developed with .NET framework as to the best of our knowledge, there is no such capability by existing automated

testing tools. Moreover, to improve the performance of SBST techniques, we will investigate the impact of tuning the parameters of an existing SBST tool, namely EVOMASTER, on the effectiveness of automated white-box testing. The study will be performed by replicating the previous studies done by our team with the most recent stable version of the EVOMASTER and re-running experiments with a larger number of case studies and different parameter settings. Last but not least, we intend to conduct an empirical study on different search algorithms which could support for generating test cases for RESTful APIs and identify potential improvements over them.

A. Systematic Literature Review

We conducted a systematic literature review [12] to discover the latest progress in the field of RESTful API testing. There were no related work in this area that could represent the current state of the art, especially by taking the recent dramatic increase of research activities in this field from 2017 onward into consideration. We only found one short similar survey on the testing of RESTful APIs [19], published in 2022. That survey was only based on 16 articles and addressing a limited number of research questions (i.e., just four). We aimed at conducting a survey on a much larger number of papers and tried our best not to miss any relevant paper. In order to find all relevant papers, we opted seven widely used databases in the field of software engineering and performed an initial search along with backward and forward snowballing that resulted in including 92 papers. In addition, we also had a much larger number of research questions. We defined 12 research questions from four perspectives which include *Publication status*, *Existing approaches*, *Available Tools* and *Addressed and Open Challenges*.

B. Automated White-Box Test Generation for .NET APIs

To the best of our knowledge, none of the existing white-box testing tools are supporting APIs developed with .NET Framework which is mostly written in C# language. This issue was also witnessed in the systematic literature review we conducted. In order to enable its white-box testing, we needed to be able to do instrumentation for collecting search-based software testing (SBST) heuristics, such as the *branch distance* and *code coverage* [20]. Thus, the proposed approach [21] is combined with a .NET bytecode instrumentation technique, allowing the gathering of heuristics at runtime while conducting the search process. Furthermore, to be more effective to guide search for generating high coverage test cases, we employed testability transformations [22] by using method replacements for collecting branch distance.

The instrumentation is achieved using the Mono.Cecil³ library, enabling the analysis and modification of CIL code. This approach specifically works with DLL-based .NET libraries, requiring instrumentation to be done after source code compilation rather than at runtime. The implementation involves a .NET Core console application, in which the main

³<https://www.mono-project.com/docs/tools+libraries/libraries/Mono.Cecil/>

method takes the SUT path as input, performs instrumentation, and saves the modified file at the designated location.

The proposed approach was implemented on top of EVOMASTER as a driver. The amount of modification to the core of EVOMASTER was insignificant as this tool is designed in a way that the core functionalities of the test generation performed by the search algorithm is carried out in a separate module.

Through experiments conducted on three .NET RESTful APIs, we observed that our approach outperforms a grey-box random testing technique in two of the experiments, demonstrating significantly better performance. However, in one case study involving a database, our approach did not surpass random testing. While EVOMASTER has the capability to handle SQL and calculate heuristics for SQL queries [23], we did not implement this technique in our bytecode instrumentation as it would require significant engineering effort due to its complexity. However, this could be a potential area for future improvement and development. Additionally, analyzing the code coverage achieved by the generated tests, we found that our approach effectively handles numerical and string-related branches, achieving line coverage ranging from 67% to 98% across multiple repetitions in two of the case studies.

C. Replicating Studies and Parameter Tuning

This part of the research aims at investigating the impacts of parameter tuning on improving existing SBST of REST APIs and studying the validity and replicability of existing published studies.

As discovered in the findings of our systematic literature review, having more REST case studies was one of the most common open challenges mentioned by the research papers. In this replication study, we will replicate a selection of five previous white-box testing studies conducted by our team with the latest version of EVOMASTER on a larger number of case studies (i.e., 15 APIs) from *EMB*⁴ [24] which is a repository containing a collection of open source REST APIs for relevant scientific research. The selected studies were conducted against older versions of the EMB which means they were evaluated with lower number of case-studies (e.g., 5). But we are in the process of replicating all of them with a much higher number and exactly the same case-studies to get a better understanding of their performance.

In addition, we will experiment more parameter settings of the search algorithm (e.g., *Probability of Random Sampling*). The findings of [25] substantiate the significant influence of parameter tuning on the performance of algorithms, and caution against the serious risk of overfitting in parameter tuning, which can undermine the generalizability of empirical analyses in Search-Based Software Engineering (SBSE). The aim of the planned parameter tuning is to find out if the performance of the test generation for the replicated studies can be improved by modifying the parameter settings of the algorithms and provide more insights to both researchers and

practitioners on selecting the parameters in the context of SBST of REST APIs.

D. Examining Different Search-Based Algorithms

EVOMASTER takes advantage of Many Independent Objective (MIO) [26] which is a population-based evolutionary algorithm designed to deal with white-box system testing problem for automating test suite generation. However, according to the “No Free Lunch” theorem [27], there is no universally superior optimization algorithm for all types of problems. In essence, there is no single best algorithm that can outperform others across all problem domains, emphasizing the need to carefully choose or design algorithms based on the specific problem at hand.

In this stage of our research, we plan to implement other search algorithms (e.g., Imperialistic Competitive Algorithm [28] and Particle Swarm Optimization [29]) and integrate them into EVOMASTER’s core. The idea is to study how the test generation is performing when different algorithms are utilized and to discover if any improvement is achieved.

A related work in this area is an empirical evaluation of 13 different search algorithms and 2 random approaches for unit testing using EvoSuite⁵ [30]. Results of the evaluation demonstrate that the choice of algorithm can have a considerable impact on the performance of whole test suite optimization. However, in this stage of our work, we are going to investigate the impact of different search algorithms on the performance of system-level test generation for REST APIs using EVOMASTER.

E. Evaluation

To evaluate the effectiveness of our research (Subsections IV-B, IV-C and IV-D), we conduct experiments on REST APIs from *EMB* repository and also industrial APIs as case studies [31], [32]. To assess the performance of our techniques, we use two different metrics. The first one is *Line Coverage* which indicates the amount of lines in the system-under-test being covered by the testing tool. The second one is *Found Faults* which shows the number of faults detected in the system-under-test during the automated testing. The experiment results will be analyzed by taking advantage of statistical methods [33], [34].

V. CURRENT STATUS

We have already written and submitted two research papers to disseminate the results of the accomplished steps of the project so far (i.e., Subsection IV-A and Subsection IV-B). The systematic literature review covering Subsection IV-A is currently under major revision in an international journal. However its initial version is also published on arXiv [12]. The second paper which is titled as “*.NET/C# Instrumentation for Search-Based Software Testing*” [21] is accepted to be published in Software Quality Journal⁶.

⁴<http://github.com/EMResearch/EMB>

⁵<https://www.evosuite.org/>

⁶<https://www.springer.com/journal/11219>

Moreover, I have co-authored two additional papers. The first paper [35], published in the Software Quality Journal, reports on the experiences of building EVOMASTER since 2017. It discusses the simplification of experiment execution, as well as the collection and automated analysis of results. The second paper [24], presented at ICST 2023, introduces the EMB repository which I have also contributed to its development.

At the moment, the third step in the project which is replicating studies and parameter tuning (i.e., Subsection IV-C) is being conducted. Coming up, more research papers will be published to spread the achievements of the remaining steps of the project. The tentative date of thesis defense is expected to be mid 2025.

ACKNOWLEDGMENT

I would like to thank my supervisors Prof. Andrea Arcuri and Dr. Man Zhang for all their continuous support, help and invaluable advice with this PhD.

REFERENCES

- [1] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.
- [2] S. Newman, *Building Microservices*. "O'Reilly Media, Inc.", 2015.
- [3] S. Segura, J. A. Parejo, J. Troya, and A. Ruiz-Cortés, "Metamorphic testing of RESTful web APIs," *IEEE Transactions on Software Engineering (TSE)*, 2017.
- [4] A. Arcuri, "Restful api automated test case generation with evomaster," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 28, no. 1, p. 3, 2019.
- [5] A. Martin-Lopez, S. Segura, and A. Ruiz-Cortés, "REStest: Automated Black-Box Testing of RESTful Web APIs," in *ACM Int. Symposium on Software Testing and Analysis (ISSTA)*. ACM, 2021, pp. 682–685.
- [6] E. Viglianisi, M. Dallago, and M. Ceccato, "Resttestgen: Automated black-box testing of restful apis," in *IEEE International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2020.
- [7] V. Atlidakis, P. Godefroid, and M. Polishchuk, "Restler: Stateful REST API fuzzing," in *ACM/IEEE International Conference on Software Engineering (ICSE)*, 2019, p. 748–758.
- [8] H. Ed-douibi, J. L. Cánovas Izquierdo, and J. Cabot, "Automatic generation of test cases for rest apis: A specification-based approach," in *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*, 2018, pp. 181–190.
- [9] A. Martin-Lopez, A. Arcuri, S. Segura, and A. Ruiz-Cortés, "Black-box and white-box test case generation for restful apis: Enemies or allies?" in *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2021, pp. 231–241.
- [10] M. Harman, Y. Jia, and Y. Zhang, "Achievements, open problems and challenges for search based software testing," in *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2015, pp. 1–12.
- [11] M. Harman and B. F. Jones, "Search-based software engineering," *Journal of Information & Software Technology*, vol. 43, no. 14, pp. 833–839, 2001.
- [12] A. Golmohammadi, M. Zhang, and A. Arcuri, "Testing restful apis: A survey," *ACM Transactions on Software Engineering and Methodology*, 2022.
- [13] M. Kumar, S. K. Singh, R. Dwivedi *et al.*, "A comparative study of black box testing and white box testing techniques," *International Journal of Advance Research in Computer Science and Management Studies*, vol. 3, no. 10, 2015.
- [14] S. K. Singh and A. Singh, *Software testing*. Vandana Publications, 2012.
- [15] "'10 Biggest Software Bugs of 2021,'" <https://www.testdevlab.com/blog/10-biggest-software-bugs-and-tech-fails-of-2021>, [Accessed: 08 Jan 2023].
- [16] P. McMinn, "Search-based software testing: Past, present and future," in *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*. IEEE, 2011, pp. 153–163.
- [17] M. Reddy, *API Design for C++*. Elsevier, 2011.
- [18] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol-http/1.1," 1999.
- [19] A. Ehsan, M. A. M. Abuhaliqa, C. Catal, and D. Mishra, "Restful api testing methodologies: Rationale, challenges, and solution directions," *Applied Sciences*, vol. 12, no. 9, p. 4369, 2022.
- [20] P. McMinn, "Search-based software test data generation: A survey," *Software Testing, Verification and Reliability*, vol. 14, no. 2, pp. 105–156, 2004.
- [21] A. Golmohammadi, M. Zhang, and A. Arcuri, ".NET/C# instrumentation for search-based software testing," *Software Quality Journal*, p. Forthcoming, 2023.
- [22] M. Harman, L. Hu, R. Hierons, J. Wegener, H. Sthamer, A. Baresel, and M. Roper, "Testability transformation," *IEEE Transactions on Software Engineering (TSE)*, vol. 30, no. 1, pp. 3–16, 2004.
- [23] A. Arcuri and J. P. Galeotti, "Handling sql databases in automated system test generation," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 29, no. 4, pp. 1–31, 2020.
- [24] A. Arcuri, M. Zhang, A. Golmohammadi, A. Belhadi, J. P. Galeotti, B. Marculescu, and S. Susruthan, "Emb: A curated corpus of web/enterprise applications and library support for software testing research," in *IEEE International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2023.
- [25] A. Arcuri and G. Fraser, "On parameter tuning in search based software engineering," in *International Symposium on Search Based Software Engineering (SSBSE)*, 2011, pp. 33–47.
- [26] A. Arcuri, "Many Independent Objective (MIO) Algorithm for Test Suite Generation," in *International Symposium on Search Based Software Engineering (SSBSE)*, 2017, pp. 3–17.
- [27] Y.-C. Ho and D. L. Pepyne, "Simple explanation of the no-free-lunch theorem and its implications," *Journal of optimization theory and applications*, vol. 115, pp. 549–570, 2002.
- [28] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition," in *2007 IEEE congress on evolutionary computation*. Ieee, 2007, pp. 4661–4667.
- [29] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. IEEE, 1995, pp. 1942–1948.
- [30] J. Campos, Y. Ge, N. Albulian, G. Fraser, M. Eler, and A. Arcuri, "An empirical evaluation of evolutionary algorithms for unit test suite generation," *Information and Software Technology (IST)*, vol. 104, pp. 207–235, 2018.
- [31] M. Zhang, A. Arcuri, Y. Li, K. Xue, Z. Wang, J. Huo, and W. Huang, "Fuzzing microservices in industry: Experience of applying evomaster at meituan," 2022. [Online]. Available: <https://arxiv.org/abs/2208.03988>
- [32] A. Arcuri and J. P. Galeotti, "Enhancing search-based testing with testability transformations for existing apis," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 1, pp. 1–34, 2021.
- [33] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *ACM/IEEE International Conference on Software Engineering (ICSE)*, 2011, pp. 1–10.
- [34] —, "A Hitchhiker's Guide to Statistical Tests for Assessing Randomized Algorithms in Software Engineering," *Software Testing, Verification and Reliability (STVR)*, vol. 24, no. 3, pp. 219–250, 2014.
- [35] A. Arcuri, M. Zhang, A. Belhadi, B. Marculescu, A. Golmohammadi, J. P. Galeotti, and S. Seran, "Building an open-source system test generation tool: lessons learned and empirical analyses with evomaster," *Software Quality Journal*, pp. 1–44, 2023.