

# Search-based Security Testing of Enterprise Microservices

Susruthan Seran<sup>1</sup>

Kristiania University College

Oslo, Norway

susruthan.seran@kristiania.no

**Abstract**---This article presents Ph.D. research that focuses on the possibilities of adopting search-based software testing methods for microservice-based software systems to improve the software's quality and security. The research is aimed at exploring the possibilities of automatically mocking external web service dependencies of the system under test and detecting security faults through different entry points (e.g., external web services and messaging backbones) using search-based white-box fuzzing. Additionally, the article also discusses the evaluation method, the current status of the research, potential outcomes, and the contribution of the research.

**Index Terms**---Microservices, Automated mock generation, Search-based test generation, Search-based security testing, Search-based software engineering

## I. INTRODUCTION

Enterprise software applications are increasingly becoming more complex in terms of the number of different components and interactions between those components. To address this complexity, microservice-based architectures are used [1]. However, the use of microservices is a double-edged sword, as it makes managing complexity easier, but also makes testing and detecting software faults more challenging [1, 2].

Microservices heavily rely on other services to perform their tasks. During automated test generation, managing such web service dependencies requires effort and time. *Mocking* is a popular strategy to tackle this challenge [3, 4]. Automated mocking could help to test for more dynamic scenarios, compared with handwritten mocks.

Moreover, microservice architecture increases the attack surface significantly. High security coverage means, application should be tested from all the potential entry points. Supply chain attacks [5] and dependency chain abuse [6] are some typical examples of why such testing is important.

The objective of this Ph.D. research is to explore novel approaches to enhance the quality of automated test case generation with high-security coverage for microservices, especially when testing services in isolation. To attain this objective, we will employ search-based software engineering techniques alongside other techniques such as taint analysis and instrumentation. In this paper, we discuss the existing literature on microservices, mocking, and search-based security testing. Alongside that, we discuss the evaluation methodology and the contribution.

## II. RELATED WORK

### A. Microservices

Microservice-based software architecture design helps to tackle large and complex enterprise software systems [1].

However, it adds more complexity to software testing, especially security testing [2]. In a microservice environment, distributed services are interconnected using communication protocols like Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS), Remote Procedure Call (RPC), and Advanced Message Queuing Protocol (AMQP) in a synchronous or asynchronous manner, to execute a common goal [1, 7].

To achieve high code coverage in automated test generation for microservice-based software, it is important that the system under test (SUT) should be able to connect with all the external web services. Existing research shows that *mocking* is an effective way to manage such environmental dependencies [8]. However, if the developer does not have ownership of an external web service dependency, it may not be feasible to construct a mock web service for it. To the best of our knowledge, the automated generation of mock external web service dependencies has not received sufficient attention in the literature, especially when it comes to messaging backbones.

Furthermore, microservices come with their own security challenges, such as ensuring trust between connected services, secret management, etc [9]. Most literature addresses the issue of securing microservices from an external perspective (e.g., exposed web endpoints) [2]. Further research is needed to test the security of microservices from various entry points, to improve security coverage.

### B. Mocking

*Mocking* is a popular strategy used in unit testing to handle dependencies when testing in isolation [10]. In general, *mocking* refers to the replacement of dependencies during testing instead of using the real dependencies to avoid conflicts. *Mock Objects* is a common approach to isolate a dependency from its original ones [3, 4]. This is achieved by replacing the real classes with mock classes. Many industrial frameworks exist for managing mock objects, such as Mockito<sup>1</sup>, EasyMock<sup>2</sup>, and JMock<sup>3</sup> [11].

However, using mock web services is a different form of mocking than using *Mock Objects*. Existing research shows that mocking environmental dependencies improves the quality of the generated test cases [8]. Furthermore, the automatic generation of controllable mock external web service dependencies is a scalable approach that helps improve code coverage during automatic test generation.

<sup>1</sup><https://site.mockito.org/>

<sup>2</sup><https://easymock.org/>

<sup>3</sup><http://jmock.org/>

Simultaneously, it also paves the path to test for security faults from other entry points of the SUT.

In the context of web service mocking, WireMock<sup>4</sup> is one of the popular frameworks that facilitates essential functionalities to manage mock web services during testing [11]. However, to the best of our knowledge, none of these tools are capable of performing mocking in an automated manner. Further, based on existing work, the automatic generation of mock external web service dependencies, including messaging backbones (e.g., Apache Kafka<sup>5</sup>), has been a less explored area.

### C. Search-based Security Testing

There is a significant number of proposals available from academia about how to build a secure software system [12]. Simultaneously, there are numerous tools and methods available to detect vulnerabilities in advance [13]. However, security flaws continue to exist [14].

Existing research indicates that search-based algorithms are successful at solving software engineering problems, particularly in software testing [15]. Similar to software testing, detecting software vulnerabilities using search-based algorithms has been a crucial research area [16, 17]. In the context of search-based security testing, previous work from various researchers has focused mostly on cross-site scripting (XSS) [18], SQL injection (SQLi) [17, 19] and Extensible Markup Language (XML) injection [20]. Even though there are papers tackling different types of web application vulnerabilities, some vulnerability types (e.g., insecure object deserialization, and broken access control) [21] did not receive enough attention based on the limited number of papers covering those [16].

In addition, increasing the attack surface is an innate feature of microservice architectures [2]. Existing work in search-based security testing focuses on the main entry point of the SUT (e.g., exposed web endpoints). The SUT must be tested thoroughly from all possible entry points to achieve high-security coverage. Based on the literature, detecting security faults from different entry points of a SUT has not received sufficient attention.

## III. RESEARCH OBJECTIVES

In summary, the main three objectives of this Ph.D. are the following:

**Objective-A:** As mentioned in the related work section, automatic generation of dynamically controllable mock external web service dependencies will improve the quality of the generated tests [4]. Moreover, controllable mock web services will help to increase the attack surface of the SUT during automatic test generation. At this stage, we will focus on automatically generating external web service dependencies for HTTP and HTTPS based web services.

**Objective-B:** It is important to test all possible entry points of a system to detect potential faults. Similar to the previous objective, we will explore novel approaches to automatically generate controllable mock messaging backbone dependencies while addressing the obstacles associated with handling asynchronous communications.

**Objective-C:** As the final objective, we aim to develop novel methods to detect security defects through various entry points of the SUT.

## IV. RESEARCH METHODOLOGY

The research will focus on designing novel techniques to be implemented as an extension to the open-source AI-driven automatic test case generation tool EVOMASTER [22]. As part of each objective, experiments will be conducted on selected open-source case studies [23], and case studies from industrial partners [7, 24]. Experiments will measure multiple metrics, including code coverage and fault-finding rate. The results will be analysed using statistical methods, Mann–Whitney U test (i.e.,  $p$ -value) and Vargha-Delaney effect size (i.e.,  $\hat{A}_{12}$ ) to perform comparison analyses between the base and our approach [25, 26].

## V. CURRENT STATUS

We have already published three papers related to the research. The initial paper is an experience report on building EVOMASTER since 2017, which was published in the Software Quality Journal [27]. It discusses the simplification of experiment execution, as well as the collection and automated analysis of results. The second paper, which was published in the 16th IEEE International Conference on Software Testing, Verification and Validation (ICST 2023), covers the case study repository used for experiments [23].

Additionally, a first-author conference paper entitled *Search-Based Mock Generation of External Web Service Interactions* [28] was published as part of *Objective-A* at the 15th Symposium on Search-Based Software Engineering (SSBSE) 2023. In addition, we are currently working on its journal extension.

Besides, we are expanding the case study repository [23] and conducting more experiments with new open-source case studies from various projects including, payment service from GOV.UK<sup>6</sup>, backend server of Signal messaging application<sup>7</sup>, and a few from The Norwegian Labour and Welfare Directorate<sup>8</sup>.

## VI. CONTRIBUTION

This novel research will make a significant contribution to the field of search-based software engineering. It will provide critical insights into automated mocking of external web service dependencies during the process of automatic test generation, especially improving the quality of generated tests. It will also contribute to search-based security testing, specifically focusing on software security of microservice-based enterprise systems.

## VII. ACKNOWLEDGEMENT

This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research, innovation program (grant agreement No. 864972).

I would like to express my sincere gratitude to my advisors, Prof. Andrea Arcuri, Dr. Man Zhang and Dr. Onur Duman, for their continuous support of my Ph.D. research.

<sup>4</sup><https://wiremock.org/>

<sup>5</sup><https://kafka.apache.org/>

<sup>6</sup><https://github.com/alphagov/pay-publicapi>

<sup>7</sup><https://github.com/signalapp/Signal-Server>

<sup>8</sup><https://github.com/navikt>

## REFERENCES

- [1] S. Newman, *Building microservices*. "O'Reilly Media, Inc.", 2021.
- [2] N. Mateus-Coelho, M. Cruz-Cunha, and L. G. Ferreira, "Security in microservices architectures," *Procedia Computer Science*, vol. 181, pp. 1225--1236, 2021.
- [3] T. Mackinnon, S. Freeman, and P. Craig, "Endo-testing: unit testing with mock objects," *Extreme programming examined*, pp. 287--301, 2000.
- [4] A. Arcuri, G. Fraser, and J. P. Galeotti, "Generating tcp/udp network data for automated unit test generation," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 155--165.
- [5] T. O. Foundation. M2: Inadequate supply chain security. [Online]. Available: <https://owasp.org/www-project-mobile-top-10/2023-risks/m2-inadequate-supply-chain-security>
- [6] -----, Cidc-sec-3: Dependency chain abuse. [Online]. Available: <https://owasp.org/www-project-top-10-ci-cd-security-risks/CICD-SEC-03-Dependency-Chain-Abuse>
- [7] M. Zhang, A. Arcuri, Y. Li, Y. Liu, and K. Xue, "White-box fuzzing rpc-based apis with evomaster: An industrial case study," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 5, pp. 1--38, 2023.
- [8] A. Arcuri, G. Fraser, and J. P. Galeotti, "Automated unit test generation for classes with environment dependencies," in *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, 2014, pp. 79--90.
- [9] OWASP. Owasp top 10 - 2017. [Online]. Available: [https://owasp.org/www-pdf-archive/OWASP\\_Top\\_10-2017\\_%28en%29.pdf.pdf](https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_%28en%29.pdf.pdf)
- [10] D. Spadini, M. Aniche, M. Bruntink, and A. Bacchelli, "To mock or not to mock? an empirical study on mocking practices," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 2017, pp. 402--412.
- [11] D. Lin, F. Liping, H. Jiajia, T. Qingzhao, S. Changhua, and Z. Xiaohui, "Research on microservice application testing based on mock technology," in *2020 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*. IEEE, 2020, pp. 815--819.
- [12] M. Felderer, M. Büchler, M. Johns, A. D. Brucker, R. Breu, and A. Pretschner, "Security testing: A survey," in *Advances in Computers*. Elsevier, 2016, vol. 101, pp. 1--51.
- [13] R. Leszczyna, "Review of cybersecurity assessment methods: Applicability perspective," *Computers & Security*, vol. 108, p. 102376, 2021.
- [14] A. Golmohammadi, M. Zhang, and A. Arcuri, "Testing restful apis: A survey," *ACM Trans. Softw. Eng. Methodol.*, aug 2023, just Accepted. [Online]. Available: <https://doi.org/10.1145/3617175>
- [15] S. Ali, L. C. Briand, H. Hemmati, and R. K. Panesar-Walawege, "A systematic review of the application and empirical investigation of search-based test case generation," *IEEE Transactions on Software Engineering*, vol. 36, no. 6, pp. 742--762, 2009.
- [16] F. Ahsan and F. Anwer, "A critical review on search-based security testing of programs," *Computational Intelligence: Select Proceedings of InCITE 2022*, pp. 207--225, 2023.
- [17] J. Thomé, A. Gorla, and A. Zeller, "Search-based security testing of web applications," in *Proceedings of the 7th International workshop on search-based software testing*, 2014, pp. 5--14.
- [18] A. Avancini and M. Ceccato, "Towards security testing with taint analysis and genetic algorithms," in *Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems*, 2010, pp. 65--71.
- [19] B. Aziz, M. Bader, and C. Hippolyte, "Search-based sql injection attacks testing using genetic programming," in *Genetic Programming: 19th European Conference, EuroGP 2016, Porto, Portugal, March 30-April 1, 2016, Proceedings 19*. Springer, 2016, pp. 183--198.
- [20] S. Jan, A. Panichella, A. Arcuri, and L. Briand, "Search-based multi-vulnerability testing of xml injections in web applications," *Empirical Software Engineering*, vol. 24, pp. 3696--3729, 2019.
- [21] OWASP. Owasp top 10 - 2021. [Online]. Available: <https://owasp.org/www-project-top-ten/>
- [22] A. Arcuri, "EvoMaster: Evolutionary Multi-context Automated System Test Generation," in *IEEE International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2018.
- [23] 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.
- [24] 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>
- [25] 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.
- [26] -----, "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.
- [27] 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.
- [28] S. Seran, M. Zhang, and A. Arcuri, "Search-based mock generation of external web service interactions," in *International Symposium on Search Based Software Engineering (SSBSE)*. Springer, 2023.