Characterizing and understanding security risks through Security-Aware Mutation Testing of security configuration in RESTFul APIs

Carlos Andres Delgado Saavedra
carlos.andres.delgado@correounivalle.edu.co
Advisors:
Jesus A. Aranda, PhD. Universidad del Valle
Gills Perrouin, PhD. Universite de Namur
James Ortiz. PhD, Universite de Namur

Overview

Problem statement

Problem Statement I

Challenges in API Security

- RESTFul APIs handle sensitive information, such as passwords and personal data.
- Common vulnerabilities include:
  - Authorization issues
  - Data encryption weaknesses
  - Security misconfigurations

Current Limitations

- Existing security tools often fail to uncover configuration-based vulnerabilities.
- High dependency on developers to manually ensure secure configurations.

Mutation Testing

Open problems according to Papadakis et al., 2019

1. Approximately 5% of the mutants are useful
2. Small semantic deviations vs blind syntactical deviations
3. Mutations may be tailored to useful mutants
4. Many redundant mutants
5. Not strong evidence that the mutants are correlated with real faults

Mutation Testing

Open problems according to Papadakis et al., 2019

6. What types of faults are not captured by simple or complex mutants?
7. What percentage of future regression errors can we capture with mutations?
8. When is it appropriate to stop the testing process?
9. How should we integrate mutation testing into our development process?

Mutation Testing

Open problems according to Loise, 2017

1. Collect security patterns in a database to create operators
2. Operators to generate vulnerable versions of the software
3. Creation of the security regression tests

Research Question

How can security-aware mutation operators be designed to improve the coverage of security testing for vulnerabilities in the configuration of security policies in RESTFul APIs?

Hypothesis

Designing security-aware mutation operators for RESTFul APIs can enhance the validation of security configuration policies, reducing the risk of vulnerabilities.

Research Questions

Research Questions I

Overview of Mutation Testing

- Introduced in 1972 as a method to evaluate software reliability by modifying code to create faults.
- Recent advancements include:
  - Mutation operators for specific languages like Java and Python.
  - Use of machine learning techniques to enhance fault detection.

Research Questions II

Security Testing in RESTFul APIs

- Focuses on detecting vulnerabilities in API endpoints.
- Common testing methods:
  - Black-box and white-box testing
  - Penetration testing and property-based testing
- Challenges include managing various data formats (e.g., JSON, XML) and ensuring comprehensive test coverage.

Research Questions III

Q1

What are the existing mutation operators for security testing of RESTFul APIs?

1. Test case generation
2. Source code based
3. Model based mutation testing
4. Mixed strategies

Research Questions IV

Q2

How effective are these mutation operators in detecting security vulnerabilities?

1. Test case generation operators are effective in detecting vulnerabilities and easy to generalize.
2. Model-based mutation depends of the abstraction level of the language and verification strategies.
3. Test case generation mutation operators are specific to the language, it is complex to generalize.

Research Questions V

Q3

What strategies are there for improving security practices in RESTFul APIs?

1. CORS
2. Auth based authentication
3. Encryption of the query parameters and data
4. Sanitize data

Research Questions VI

Q4

What elements define common security misconfigurations in RESTFul APIs?

1. CORS misconfigurations
2. Bad configuration of the authentication, using leak credentials.
3. Not using encryption for the query parameters.
4. Using default configurations.
5. Insecure HTTP methods.
6. Insufficient log and monitoring.
7. Data exposure.

Research Questions VII

Gaps Identified

- Mutation operators are often language-specific, limiting general applicability.
- Insufficient focus on security-aware mutation testing for RESTFul APIs.
- Lack of standardized frameworks for evaluating security tests.

Research Objectives

Research Objectives I

General Objective

- Develop a collection of security-aware mutation operators designed for evaluating the configuration of security policy files within RESTFul APIs.

Research Objectives II

Specific Objectives

1. Identify the key elements of security policies in RESTFul APIs.
2. Design a set of security-aware mutation operators for testing security policies.
3. Develop the security-aware mutation operators and integrate them into testing tools.
4. Evaluate the proposed operators against existing security testing frameworks, focusing on their effectiveness and coverage.

Research Objectives III

Expected Results

- A comprehensive set of mutation operators tailored for RESTFul API security.
- Detailed reports on the performance of these operators compared to current tools.
- Contribution to the development of frameworks for automated security testing.

Methodology

Proposed Methodology I

The research is divided into four key phases:

1. **Systematic Literature Review:**
    - Identify existing vulnerabilities and security-aware mutation operators.
    - Analyze current tools and techniques used for testing RESTFul APIs.
2. **Design of Mutation Operators:**
    - Define strategies to introduce vulnerabilities using models.
    - Specify and describe security-aware mutation operators.
3. **Development of Mutation Operators:**

Proposed Methodology II

- Specification of mutation testing operators
- Refactor the implementation for efficiency and maintainability.

4. **Evaluation:**
   - Apply operators to case studies.
   - Measure coverage, fault detection, and mutation score.

Proposed Methodology III

Techniques Used

- **Test-Driven Development (TDD):** Ensure mutation operators function as intended.
- **Snowballing Methodology:** Review recent surveys and track relevant studies.
- **Evaluation Metrics:** Analyze coverage, redundancy, and effectiveness of the operators.

Expected Deliverables

- A set of validated mutation operators for RESTFul API security testing.
- A framework for automating security-aware mutation testing.
- Reports detailing the findings and contributions.

Timeline I

The project timeline is structured over 24 months, covering the following phases:

1. **Systematic Literature Review:** Months 1-6
    - Review recent surveys and identify vulnerabilities in RESTFul APIs.
    - Analyze security-aware mutation operators and existing tools.
2. **Design of Mutation Operators:** Months 7-12
    - Define strategies to introduce vulnerabilities.
    - Specify and describe the proposed mutation operators.
3. **Verification of Mutation Operators:** Months 13-18

Timeline II

- Select the strategy to validate the mutation operators.
- Evaluate a set of testing cases to validate the operators.

4. **Evaluation:** Months 19-24
    - Apply the mutation operators to case studies.
    - Evaluate their effectiveness using coverage and redundancy metrics.
    - Generate detailed reports and summarize findings.

Timeline III

Visual Timeline

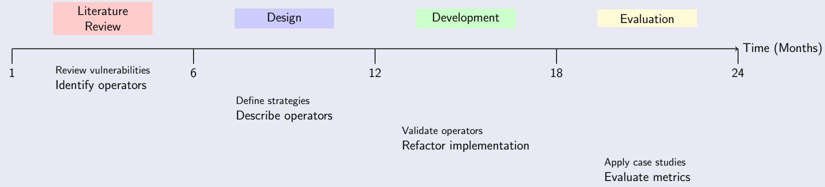

Figure 2: Timeline of Methodology

Expected Results

Expected Results I

Impact on RESTFul API Security

- Provide a systematic approach for evaluating the robustness of RESTFul API security configurations.
- Enhance the ability of developers to detect vulnerabilities during the development lifecycle.

Contributions to the Field

- Specification of a comprehensive set of security-aware mutation operators applicable to RESTFul APIs.
- Introduction of a generic framework for automated security testing tools.
- Empirical evidence showcasing improvements in test coverage and fault detection rates.

Expected Results II

Anticipated Challenges

- Balancing the trade-off between test coverage and execution time.
- Addressing redundancy in mutation operators to avoid excessive equivalent mutants.
- Ensuring scalability and applicability across different frameworks and programming languages.

Questions?

Thank you!

Ahmed, S., & Hamdy, A. (2023). Artificial bee colony for automated black-box testing of restful api. In *Smart innovation, systems and technologies* (pp. 1–17). Springer Nature Singapore. https://doi.org/10.1007/978-981-99-6706-3_1

Ahmed, Z., Zahoor, M., & Younas, I. (2010).Mutation operators for object-oriented systems: A survey. https://doi.org/10.1109/iccae.2010.5451692

Ami, A. S., Kafle, K., Moran, K., Nadkarni, A., & Poshyvanyk, D. (2021).Systematic mutation-based evaluation of the soundness of security-focused android static analysis techniques. *ACM Transactions on Privacy and Security*, *24*(3), 1–37. https://doi.org/10.1145/3439802

Andre, J., & Agnelo, N. (2020). *A robustness testing approach for restful web services*.

Arcuri, A., & Galeotti, J. P. (2020). Testability transformations for existing apis. *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, 153–163. https://doi.org/10.1109/ICST46399.2020.00025

Atlidakis, V., Geambasu, R., Godefroid, P., Polishchuk, M., & Ray, B. (2020). Pythia: Grammar-based fuzzing of rest apis with coverage-guided feedback and learning-based mutations. https://doi.org/10.48550/ARXIV.2005.11498

Bakhtin, A., Al Maruf, A., Cerny, T., & Taibi, D. (2022). Survey on tools and techniques detecting microservice api patterns. *2022 IEEE International Conference on Services Computing (SCC)*. https://doi.org/10.1109/scc55611.2022.00018

Belhadi, A., Zhang, M., & Arcuri, A. (2024). Random testing and evolutionary testing for fuzzing graphql apis. *ACM*

Transactions on the Web, *18*(1), 1–41. https://doi.org/10.1145/3609427

Corradini, D., Zampieri, A., Pasqua, M., Viglianisi, E., Dallago, M., & Ceccato, M. (2022).Automated black-box testing of nominal and error scenarios in restful apis. *Software Testing, Verification and Reliability*, *32*(5). https://doi.org/10.1002/stvr.1808

Ehsan, A., Abuhaliqa, M. A. M. E., Catal, C., & Mishra, D. (2022).Restful api testing methodologies: Rationale, challenges, and solution directions. *Applied Sciences*, *12*(9), 4369. https://doi.org/10.3390/app12094369

Felício, D., Simão, J., & Datia, N. (2023).Rapitest: Continuous black-box testing of restful web apis. *Procedia Computer Science*, *219*, 537–545. https://doi.org/10.1016/j.procs.2023.01.322

Golmohammadi, A., Zhang, M., & Arcuri, A. (2023). Testing restful apis: A survey. *ACM Trans. Softw. Eng. Methodol.* https://doi.org/10.1145/3617175

Hussain, F., Hussain, R., Noye, B., & Sharieh, S. (2020). Enterprise api security and gdpr compliance: Design and implementation perspective. *IT Professional*, *22*(5), 81–89. https://doi.org/10.1109/mitp.2020.2973852

Idris, M., Syarif, I., & Winarno, I. (2022). Web application security education platform based on OWASP API security project. *EMIT. Int. J. Eng. Technol.*, 246–261.

Jin, Z., Xing, L., Fang, Y., Jia, Y., Yuan, B., & Liu, Q. (2022). P-verifier: Understanding and mitigating security risks in cloud-based iot access policies. *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security.* https://doi.org/10.1145/3548606.3560680

Kellezi, D., Boegelund, C., & Meng, W. (2019). Towards secure open banking architecture: An evaluation with owasp. In *Lecture notes in computer science* (pp. 185–198). Springer International Publishing. https://doi.org/10.1007/978-3-030-36938-5_11

Khoda Parast, F., Sindhav, C., Nikam, S., Izadi Yekta, H., Kent, K. B., & Hakak, S. (2022).Cloud computing security: A survey of service-based models. *Computers and Security*, *114*, 102580. https://doi.org/10.1016/j.cose.2021.102580

Kim, M., Xin, Q., Sinha, S., & Orso, A. (2022).Automated test generation for rest apis: No time to rest yet. *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis.* https://doi.org/10.1145/3533767.3534401

Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., Emam, K. E., &

Rosenberg, J. (2002).Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, *28*, 721–734. https://doi.org/10.1109/TSE.2002.1027796

Leotta, M., Paparella, D., & Ricca, F. (2023).Mutta: A novel tool for e2e web mutation testing. *Software Quality Journal*. https://doi.org/10.1007/s11219-023-09616-6

Loise, T. (2017). *Towards security aware mutation testing* [Master's thesis] [Mouehli et al. [39Therefore, for the purposes of this study we will use the following definition: a security bug is a piece of code that can lead to one or several vulnerabilities in an application.].

Luo, Y., Puyang, T., Luo, W., Shen, Q., Ruan, A., & Wu, Z. (2016). Multipol: Towards a multi-policy authorization framework for restful interfaces in the cloud. In *Lecture notes in computer science* (pp. 214–226). Springer

International Publishing.
https://doi.org/10.1007/978-3-319-50011-9_17

Lyu, C., Xu, J., Ji, S., Zhang, X., Wang, Q., Zhao, B., Pan, G., Cao, W., & Beyah, R. (2023). Miner: A hybrid data-driven approach for rest api fuzzing.
https://doi.org/10.48550/ARXIV.2303.02545

Madden, N. (2021, February). *API security in action*. Manning Publications.

Marculescu, B., Zhang, M., & Arcuri, A. (2022).On the faults found in rest apis by automated test generation. *ACM Transactions on Software Engineering and Methodology*, *31*(3), 1–43.
https://doi.org/10.1145/3491038

Martin-Lopez, A., Segura, S., & Ruiz-Cortés, A. (2020). Restest: Black-box constraint-based testing of restful web apis. In *Lecture notes in computer science* (pp. 459–475). Springer International Publishing. https://doi.org/10.1007/978-3-030-65310-1_33

Martin-Lopez, A., Segura, S., & Ruiz-Cortés, A. (2022). Online testing of restful apis: Promises and challenges. https://doi.org/10.5281/ZENODO.6941292

Noy, C. (2008).Sampling knowledge: The hermeneutics of snowball sampling in qualitative research. *International Journal of Social Research Methodology*, *11*(4), 327–344. https://doi.org/10.1080/13645570701401305

Papadakis, M., Kintis, M., Zhang, J., Jia, Y., Traon, Y. L., & Harman, M. (2019).Mutation testing advances: An analysis and survey (1st ed.). *Advances in Computers*, *112*, 275–378. https://doi.org/10.1016/bs.adcom.2018.03.015

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007).A design science research methodology for information systems research. *The Missouri Review, 24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Petrović, G., Ivanković, M., Fraser, G., & Just, R. (2021). Practical mutation testing at scale. https://doi.org/10.48550/ARXIV.2102.11378

Roth, S., Barron, T., Calzavara, S., Nikiforakis, N., & Stock, B. (2020).Complex security policy? a longitudinal analysis of deployed content security policies. *Proceedings 2020 Network and Distributed System Security Symposium.* https://doi.org/10.14722/ndss.2020.23046

Sánchez, A. B., Delgado-Pérez, P., Medina-Bulo, I., & Segura, S. (2022).Mutation testing in the wild: Findings from github. *Empirical Software Engineering, 27*(6). https://doi.org/10.1007/s10664-022-10177-8

Siriwardena, P. (2020). *Advanced api security: Oauth 2.0 and beyond*. Apress. https://doi.org/10.1007/978-1-4842-2050-4

Subramanian, H., & Raj, P. (2019, January). *Hands-On RESTful API design patterns and best practices*. Packt Publishing.

Tokos, A. (2023). *Evaluating fuzzing tools for automated testing of rest apis using openapi specification*.

Tsai, C.-H., Tsai, S.-C., & Huang, S.-K. (2021).Rest api fuzzing by coverage level guided blackbox testing. *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*, 291–300. https://doi.org/10.1109/QRS54544.2021.00040

Viglianisi, E., Dallago, M., & Ceccato, M. (2020).Resttestgen: Automated black-box testing of restful apis. *2020 IEEE 13th International Conference on Software Testing,*

Validation and Verification (ICST), 142–152.
https://doi.org/10.1109/ICST46399.2020.00024

Votipka, D., Fulton, K. R., Parker, J., Hou, M., Mazurek, M. L., & Hicks, M. (2020). Understanding security mistakes developers make: Qualitative analysis from build it, break it, fix it. *29th USENIX Security Symposium (USENIX Security 20)*, 109–126.
https://www.usenix.org/conference/usenixsecurity20/presentation/votipka-understanding

Williams, L., Maximilien, E. M., & Vouk, M. (2003). Test-driven development as a defect-reduction practice. *14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003.*, 34–45.

Wu, H., Xu, L., Niu, X., & Nie, C. (2022). Combinatorial testing of restful apis. *Proceedings of the 44th International Conference on Software Engineering*.
https://doi.org/10.1145/3510003.3510151

Yandrapally, R., & Mesbah, A. (2021). Mutation analysis for assessing end-to-end web tests. *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 183–194. https://doi.org/10.1109/ICSME52107.2021.00023

Zhang, M., & Arcuri, A. (2023). Open problems in fuzzing restful apis: A comparison of tools. *ACM Transactions on Software Engineering and Methodology*, *32*(6), 1–45. https://doi.org/10.1145/3597205