Characterizing and understanding security risks through Fuzzing Secure-Aware Mutation Testing of RESTFul-API services

Carlos Andres Delgado Saavedra
carlos.andres.delgado@correounivalle.edu.co

Universidad del Valle

Université de Namur

AVISPA
Ambientes Visuales de Programación Aplicativa

Colombia

February 23th, 2024

Overview

1. Research Proposal
2. Literature Review
    - Next tasks
    - Challenges
3. Scope
4. Methodology

Research Proposal

Problem

1. API-RESTFul is an architectural style for designing web services
2. RESTFul APIs exchange sensitive information and private date
3. Top 10 vulnerabilities Application Security Project (OWASP) https://owasp.org/www-project-api-security/
4. Coverage of the security tests: penetration and policies
5. Opportunity for mutation testing

Research question

¿How to design fuzzed secure-aware mutation operators in the coverage of the vulnerabilities in the configuration of security policies in RESTFul APis?

Objectives

Develop a collection of security-aware mutation operators designed for safeguarding the configuration of security policies within RESTFul API services.

Specific

Specific objective	Expected result
1. Identification of the elements of the security policies in API-RESTFul services	Characteristics of the security policies in API-Restful services
2. Describe a set of fuzzed security-aware mutation operators for testing of security policies in API-RESTFul services	Description of the mutation operators according to the elements of security policies in API-Restful services
3. Develop the set of security-aware mutation operators for testing in Django Rest and Flask Frameworks in Python	Source code of the secure-aware mutation operators
4. Evaluate the proposed security-aware mutation operators in REST-Ful API services	Report about the performance of the created operators against tools from the literature.

Table 1: Specific objectives and expected results

Literature Review

Universidad del Valle

Strategy

1. Questions about the current state of art in the configuration security policies of RESTFul APIs.
2. Window of time from 2000 to 2024. https://doi.org/10.1515/itit-2013-1035
3. Emphasis in the last 5 years. https://doi.org/10.1145/3617175, https://journal.ijresm.com/index.php/ijresm/article/view/970 the rise of the RESTFul APIs.

Research questions

1. RQ1: What are the elements of the security configuration policies in the RESTFul API Services?
2. RQ2: What are the current challenges about the security policies of RESTFul API Services?
3. RQ3: What are the most common configuration security mistakes of the developers in the building of RESTFul API Services?
4. RQ4: What are the current testing techniques and tools for the testing of configuration policies of RESTFul API Services based on Python?
5. RQ5: What experiences have been reported in the literature about the use of mutation testing for the security testing of RESTFul API Services?

RQ1: Elements of security configuration policies

1. Authentication: Methods for the identification of the user.
2. Authorization: Methods for the access control.
3. Encryption: Protocol SSL/TLS.
4. Data masking: Hide sensitive data in logs and responses.
5. Input validation and sanitization: Prevent injection attacks (SQL, XSS).
6. Thottling: Number of requests per time.
7. API Keys: Each user with their own key.
8. Login level: Detailed and security monitoring.

RQ1: References I

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

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

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

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.

RQ2: Current challenges

1. Keep the data integrity in RESTFul API Services is a challenge that changes every day.
2. Several recent studies have identified security gaps in many of them.
3. One of the most problems about software vulnerabilities is the configuration security policies of RESTFul APIs
4. Testing methods and tools are not enough to cover all the vulnerabilities.

RQ2: References I

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

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.

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

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

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

RQ3: Common configuration mistakes

1. Lack of input validation.
2. Insecure deserialization.
3. Lack of proper authentication and authorization.
4. Insecure direct object references.
5. Lack of proper logging and monitoring.
6. Insecure communication with untrusted components.

RQ3: References I

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

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

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

RQ3: References II

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

RQ4: Testing techniques and tools

Penetration testing, vulnerability assessment, and network scanning.

1. OWASP ZAP: Penetration testing.
2. Postman: API testing.
3. Burp Suite: Penetration testing.
4. Nessus: Vulnerability assessment.
5. Nmap: Network scanning.
6. Metasploit: Penetration testing.

Techniques: Fuzzing, black box, statistical.

RQ4: References I

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

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

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

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

RQ4: References II

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

RQ5: Mutation testing in security of RESTFul API Services

1. Mutation testing has proven to be a strategy for evaluating the security of applications.
2. The literature suggests an emphasis in data integrity.
3. Different strategies for the mutation testing: using artificial intelligence, black box testing, penetration testing, validation of data integrity and statistical methods.

RQ5: References I

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

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

RQ5: References II

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

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

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

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

RQ5: References III

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

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

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

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

Tasks

1. Finish the literature review: Categories and subcategories. Article of the review of the state of the art.
2. Adjust the proposal according to this guidelines.
3. Defense of the proposal.

Challenges

1. RESTFul APIs handle sensitive information that needs to be protected, software testing evaluates how they are handled, but because vulnerabilities are constantly being discovered, there is an opportunity for improvement in this area.
2. Mutation testing has proven to be a strategy for evaluating the security of applications, there has been a lot of work done related to specific applications in languages such as Java and Python, there is an opportunity to contribute to the development of RESTFul API.
3. Security is a challenge for software development today, and several recent studies have identified security gaps in many of them, which could be studied to provide a framework for the development of tools to assess data security and generate recommendations for improvement.

Universidad del Valle

Scope

Scope

1. Security configuration files of applications based on Django and Flask.
2. A selected compatible versions of these frameworks.
3. A selected group of vulnerabilities: security of exchange of data, authentication policies and authorization policies.
4. The mutation operators are modifications of the configuration files
5. The approach is apply penetration testing and black box testing to test the mutation operators
6. The mutation operators are going to design in a open source mutation testing engine
7. The objective measurements are the mutation score of a vulnerabilities selected group and the perfect of redundant and unuseful mutants

Methodology

Methodology I

We identified four big steps:

1. Systematic review of the literature according to Kitchenham et al., 2002.
2. Design of the secure-aware mutation operators for the RESTFul API services.
3. Development of the secure-aware mutation operators for the RESTFul API services.
4. Validation of the secure-aware mutation operators for the RESTFul API services.

Systematic review of literature I

1. Plan review
    1. Define the research questions.
    2. Develop the review protocol: Snowball method.
    3. Validate the review protocol.
2. Conduct review
    1. Identify relevant study
    2. Select primary sources
    3. Extract data
    4. Select data
3. Report review
    1. Summarize the review
    2. Interpret the review
    3. Validate the review

Design of mutation operators I

1. Description of the change of the security configuration properties according to the selected vulnerabilities.
2. Validation of the proposed artificial security vulnerabilities to be introduced: State machine strategy.
3. Changing of source code strategies to introduce an artificial vulnerability: Source code modification based on creation a mutated security configuration file introducing modifications.

Develop of mutation operators I

1. TDD Methodology: Validation a priori of strategies to modify the code
2. Select a python-based mutation testing engine
3. Codification of the mutation operators
4. Evaluation of the unuseful and redundant mutants: according to Papadakis et al., 2019
5. introducing black-box tests and penetration tests to the evaluation

Validation of mutation operators I

1. Selection of a group API-based applications in Django and Flask (compatible version) in open source repositories like Github and Gitlab
2. Application of the mutation testing strategy using black-box tests and pentration tests
3. Analysis of the mutation score and percent of useful mutatans to evalute the perfomance of the strategy

References I

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

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

References II

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

References III

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

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.

References IV

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

References V

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

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.

References VI

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

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

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

References VII

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*.

References VIII

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

References IX

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