# 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

Universidad del Valle

UNIVERSITÉ DE NAMUR

AVISPA
AMBIENTES VISUALES DE PROGRAMACIÓN APLICATIVA

August, 2024

# Overview

1. Research Proposal
2. Methodology
   - Description of the methodology
3. Phases of the Project
4. Systematic Review of the Literature
   - Research Methodology

# Research Proposal

# Context

- RESTful API: An architectural style for designing web services.
  - Uses HTTP requests to access resources.
  - Offers flexibility and scalability for system communication.
- Security challenges in RESTful APIs:
  - Exchange of sensitive data (passwords, credit card numbers, personal information).
  - Vulnerabilities due to lack of authentication and authorization.
- Modern security practices:
  - Encrypting communication.
  - Requiring authentication.
  - Input validation.
  - Restricting resource access.

# The Problem

- RESTful APIs often handle sensitive and private data.
- Critical security mechanisms:
  - Authorization and access policies.
  - Access restrictions and encryption.
- OWASP 2023 reports an increase in API security risks:
  - Authorization lacking.
  - Uncontrolled resource consumption.
  - Security misconfiguration.
  - Unauthorized data access.
- Companies must invest in:
  - Updating applications and security policies.
  - Monitoring data exchange.
  - Implementing encryption protocols (HTTPS/TLS).
  - Authorization mechanisms (OAuth).

# The Importance of Software Testing I

- Growing importance of software testing in detecting vulnerabilities:
    - Early identification and fixing of vulnerabilities.
    - Prevent exploitation by attackers.
- Common vulnerabilities in RESTful APIs:
    - Broken object-level authorization.
    - Broken user authentication.
    - Excessive data exposure.
- Other security risks:
    - Injection attacks (malicious code).
    - Rate limiting attacks (API overload).
    - Denial-of-service attacks.

# Role of Mutation Testing I

- Mutation testing: A tool to evaluate security test capabilities.
    - Creates new scenarios by mutating code.
    - Helps identify potential new vulnerabilities.
- Benefits of mutation testing:
    - Detects unexpected vulnerabilities.
    - Simulates risk situations exploited by attackers.
- Need for security-aware mutation operators:
    - Provides a framework for security tests.
    - Evaluates the quality of security tests performed by developers.

# 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?

# Objectives

Develop a collection of security-aware mutation operators designed for the evaluation of the configuration of security policies files within RESTful APIs.

# Specific

| Specific objective | Expected result |
|---|---|
| 1. Identification of the elements of the security policies in RESTful APIs | Characteristics of the security policies in RESTful API, related to exchanging of data |
| 2. Describe a set of code-based security-aware mutation operators for testing of security policies files in RESTful APIs | Description of the mutation operators, introducing some misconfiguration security policies in the exchanging of data in RESTful APIs |
| 3. Develop the set of security-aware mutation operators for security configuration files | Description of the operators to be applied in security configuration files |
| 4. Evaluate the proposed security-aware mutation operators in the coverage of the security tests | Report about the performance of the created operators against tools from the literature. |

Table 1: Specific objectives and expected results

# Methodology

# Introduction

- Review of vulnerabilities in RESTful APIs
- Description of mutation operators
- Prototype implementation and testing

# Review of Vulnerabilities in RESTful APIs

- Research methodology: Snowballing Noy, 2008
- Initial focus: Recent surveys on mutation testing Papadakis et al., 2019, testing challenges for RESTful APIs Ehsan et al., 2022, and software security testing Golmohammadi et al., 2023.
- Identification of common vulnerabilities and mitigation strategies.
- Focus on OWASP 2023 top 10 vulnerabilities.

# OWASP Top 10 Vulnerabilities for 2023

1. Broken object-level authorization
2. Broken authentication
3. Unrestricted resource consumption
4. Broken authorization at the role level
5. Unrestricted access to sensitive business flows
6. Server-side request forgery (SSRF)
7. Security misconfiguration
8. Inadequate inventory management
9. Insecure API consumption

# Objective of the Vulnerability Review

- Explore characteristics of common vulnerabilities.
- Analyze how these vulnerabilities are handled in the software development process.
- Identify strategies used to mitigate vulnerabilities.
- Determine mutation operators to implement in the prototype.

# Description of Mutation Operators

- Define strategy to introduce vulnerabilities into source code.
- Variations of mutation operators to produce vulnerability effects.
- Analyze possible redundant mutants produced by the mutation operators.

# Mutation Operators: Implementation Details

- Focus on modifying:
  - Configuration files of the RESTful API
  - Source code of the API
  - Test cases
- Goal: Introduce vulnerabilities to analyze and test mitigation strategies.

# Prototype Implementation and Testing

- Approach: Test-Driven Development (TDD) Williams et al., 2003
- Generate test cases from mutation operator descriptions.
- Validate the effect of mutation operators in introducing and identifying vulnerabilities.

# Expected Outcomes

- Successful identification of vulnerabilities through mutation testing.
- Effective mitigation strategies for each identified vulnerability.
- A comprehensive list of mutation operators applicable to RESTful API security testing.

# Phases of the Project

# Phases of the Project

This project defines four phases to approach the objectives:

1. Systematic review of the literature Kitchenham et al., 2002.
2. Design of the security-aware mutation operators for RESTful API services Peffers et al., 2007.
3. Development of the security-aware mutation operators using TDD methodology.
4. Evaluation of the mutation operators using metrics Z. Ahmed et al., 2010.

# Systematic Review of the Literature

# Systematic Review of the Literature

- Conduct a systematic review to identify existing security-aware mutation operators.
- Steps to follow:
  1. Developing a research question.
  2. Identifying relevant databases.
  3. Defining search terms.
  4. Selection criteria.
  5. Data extraction and analysis.

# Research Questions

Key questions guiding the literature review:

1. What are the existing mutation operators for testing the security of RESTful APIs?
2. How effective are these mutation operators in detecting security vulnerabilities?
3. What are the limitations of current mutation operators?
4. What elements define vulnerabilities in RESTful API services?
5. How are these vulnerabilities handled in development?
6. Strategies for mitigating vulnerabilities?
7. Common security misconfigurations?

# Design of the Security-aware Mutation Operators

The design phase focuses on defining and specifying mutation operators based on identified vulnerabilities.

- Identification of vulnerability elements.
- Specification of mutation operators.
- Description of mutation application.
- Determination of testing elements.
- Analysis of coverage and redundancy.
- Evaluation of operator effectiveness.
- Refinement and iteration.

# Development of the Security-aware Mutation Operators

- TDD methodology to ensure desired effects.
- Steps in the development phase:
    1. Selection of case studies using Python frameworks.
    2. Coding mutation operators using tools like MutPy and MutMut.
    3. Analyzing coverage and redundancy metrics.
    4. Evaluating operator effectiveness.
    5. Refactoring code post-test.

# Evaluation of the Security-aware Mutation Operators

The evaluation phase measures the effectiveness of mutation operators using key metrics.

- **Benchmark Selection:** Choosing RESTful APIs with known vulnerabilities.
- **Mutation Operator Application:** Generating mutant APIs using designed operators.
- **Test Execution:** Running test cases against original and mutant APIs.
- **Evaluation and Analysis:** Using metrics like mutation coverage, fault detection rate, and false positive rate.

# 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

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

## 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

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.

# 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

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

# References VII

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

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

# References VIII

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

# References IX

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.

# References X

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