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

August, 2024

Overview

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

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

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

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

Universidad del Valle

References II

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., 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

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.

Universidad del Valle