Characterizing and understanding security risks through Secure-Aware Mutation Testing of RESTful APIs

Carlos Andres Delgado Saavedra. Msc
Student Code: 202004419
carlos.andres.delgado@correounivalle.edu.co

Advisers
Jesús Alexander Aranda. PhD. Universidad del Valle,
Gilles Perrouin. PhD. Namur University, Belgium,
James Ortiz. PhD. Namur University, Belgium,

Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación
Doctorado en Ingeniería: Enfasis en ciencias de la computación
Cali, Valle del Cauca
2023

Abstract

Mutation testing changes the code to see if the tests can find the alterations. Then, we can know if the tests can detect security issues and unexpected flaws.

RESTFul APIs is a mechanism for exchanging information between applications, some of this sensitive information. Research suggests that current security tools are not enough for privacy problems and vulnerabilities.

Mutation testing is a way to test for RESTFul API vulnerabilities by evaluating different data vulnerability and access scenarios. Our proposal aims to identify and create specific vulnerabilities in Flask and Django Rest frameworks using Python language, based on the set of vulnerabilities specified by OSWAP for 2023. We will then evaluate the effectiveness of the tests typically performed during the development of these applications.

Security-aware mutation operators can boost RESTful API software testing coverage to ensure secure access to sensitive data.

Introduction

Mutation testing is a set of practices that consists in the application of mutation operators, which are the introduction of changes into the applications code and see if the security tests can able to detect it. These are useful in the security analysis of policies[1], fail models[2] and quality of software updates[3]. RESTful APIs use HTTP requests to exchange data, with this technology is possible to perform operations of reading, updating, creating and deleting in different sources of data, hence, the security of this data is an important concern in software development. There are some risks according OWASP foundation related to authorization, authentication, resource consumption and encryption of the data.

Security in data in web contexts is an enormous challenge because some applications handle sensitive information and it is fundamental secure them from a potential intruder. Developers apply different security tests in development cycles according to different policies in security.

There are challenges related to the security of the information that is exchanged in the RESTFul APIs, because unknown vulnerabilities are revealed constantly because of situations that arise with unauthorized access to sensitive information. Software testing evaluates vulnerabilities, but there are limitations regarding certain situations that cannot be evidenced immediately, hence security-aware mutation operators can be designed that can provide an improvement for the detection of these through mutation testing.

The goal of this research proposal is to extend the work of Loise Thomas, *Towards secure-aware mutation Testing* [4], which designed mutation operators to test security policies in Java, in RESTFul API information exchange applications, because it is a similar and current problem in the field of software development.

In this way, the state of the art shows that exists a limitation in the mutation operators because these only works in limited backgrounds because their design is only for a specific language or library. Thus, it reinforced the idea about a potential contribution of a set of secure-aware mutation operators for the application of new and better strategies to test the security of the RESTFul API.

The structure of this document is as follows: Chapter 2 presents the research problem and the objectives. Chapter 3 discusses the challenges in mutation testing and security in RESTful APIs. Chapter 4 explains the methodology used in this work. Finally, the expected results are described in Chapter 5.

Contents

1 Problem and objectives **2**
- 1.1 Context . . . 2
- 1.2 The problem . . . 2
- 1.3 Research question . . . 3
- 1.4 Hypothesis . . . 4
- 1.5 Objectives . . . 4
  - 1.5.1 General . . . 4
  - 1.5.2 Specific . . . 4
- 1.6 Limitations . . . 4
- 1.7 Impact of this proposal . . . 4

2 State of the art **6**
- 2.1 Mutation testing . . . 6
- 2.2 Testing in RESTful API . . . 7
- 2.3 Security in RESTful-API . . . 7
- 2.4 Identified Challenges . . . 9

3 Methodology **10**
- 3.1 Researching methodology . . . 10
  - 3.1.1 Review of vulnerabilities in RESTful APIs . . . 10
  - 3.1.2 Description of the mutation operators . . . 11
  - 3.1.3 Prototype implementation and testing . . . 11
- 3.2 Activities . . . 11
  - 3.2.1 Phase I: Review of vulnerabilities in RESTful APIs . . . 11
  - 3.2.2 Phase II: Description of the mutation operators . . . 12
  - 3.2.3 Phase III: Prototype implementation and testing . . . 12

4 Expected Results **13**

Chapter 1

Problem and objectives

This chapter explains the security challenges of RESTful API implementations. It will also clearly state the problem that the research is intended to solve, identify the specific questions that the research will attempt to answer, state the researcher's prediction about the outcome of the research, outline the specific goals of the research, acknowledge the limitations of the research, and discuss the potential impact of the research.

1.1 Context

API-Restful is an architectural style for designing web services. It is based on the principle of using HTTP requests to access resources. This makes it a very flexible and scalable way to exchange information between systems. However, API-Restful also introduces some security challenges.

One of the main challenges is that API-Restful is the exchanging of sensitive data, such as passwords and credit card numbers, can be easily intercepted by attackers. Another challenge is that API-Restful is often used to access resources that are not protected by authentication and authorization mechanisms. This means that attackers can easily gain unauthorized access to sensitive data.

Nowadays, to address these security challenges, the communication is encrypted, it is necessary to authenticate to access to the data and mechanisms of validation input data and restrict the access to the API.

1.2 The problem

Security of the information in RESTful APIs is a sensitive problem because some of them exchange sensitive information and private date, there are some mechanisms to protect the information like authorization, policies of access and encryption, but it is a hard task to block the access of intruders because every day new vulnerabilities appear. Therefore, developers must create mechanisms to configure and secure the information against unauthorized access.

According to the open Worldwide Application Security Project (OWASP) in 2023 there is an increase in the security risks in the operation of API-Restful related to authorization,

resource consumption, security misconfiguration, and unauthorized access to data. Nowadays, companies need to invest significant effort in updating applications, improving security policies, and monitoring the exchange of data in APIs. Nowadays companies need to invest an important effort to up-to-date the applications, improving the security policies and monitoring the exchange of date in APIs, such ash using of encrypt protocols like HTTPS/TLS, authorization mechanisms like OAuth, monitor the activity of the APIs, applying some restrictions of access like Cross-Origin-Resource-Sharing (CORS) and appliyng some policies about their use[5].

The literature revise shows that during the decade has growth the importance of detecting and avoiding vulnerabilities through software testing[6] during the development cycle has grown. This is because software testing can help to identify and fix security vulnerabilities early in the development process, before they can be exploited by attackers. Recent research has shown that the risks of security of APIs are still a major concern. A study by [2] found that the top three API security risks in 2023. The most important is broken object-level authorization, this occurs when an attacker is able to access resources that they should not have access to. Other important risk is the broken user authentication, where an attacker is able to gain unauthorized access to an API by bypassing the authentication mechanism. Third, excessive data exposure, when too much sensitive data is exposed to the public and it is not possible to control effectively the access to them..

Other studies about API security risks[7], injection attacks, these attacks involve injecting malicious code into an API. Rate limiting attacks involve sending too many requests to an API in a short period of time, which can overload the API and cause it to crash. Denial-of-service attacks: These attacks involve flooding an API with requests, which prevents legitimate users from accessing the API. Developers need to be aware of these risks and take steps to mitigate them, some of them are related to using encrypt algorithms.

Mutation testing is an useful tool to evaluate the capacity of the security tests to detect vulnerabilities and allow to study how to detect potential new vulnerabilities in the software, because mutation testing can be used to create new scenarios that the test suite has not yet encountered. By mutating the code and evaluating if the test process can detect the introduced vulnerabilities, mutation testing can help to identify potential new vulnerabilities that may not have been found by other testing methods.

The strong potential of mutation testing is the detection of unexpected vulnerabilities for the developers because mutation operators can simulate risks situations which are exploited by intruders. Therefore, the specification of RESTful operators secure-aware mutation operators can provide a new framework of security tests for the evaluation of the quality of the protection of the data.

1.3 Research question

¿How to desing secure-aware mutation operators in the coverage of vulnerabilities in exchanging data in RESTful APIs?

1.4 Hypothesis

Design of secure-aware code mutation operators in RESTful APIs can improve the security of the exchanging data.

1.5 Objectives

1.5.1 General

Develop a collection of security-conscious mutation operators designed for safeguarding data integrity within Restful APIs.

1.5.2 Specific

Specific objective	**Expected result**
1. Selection of a common set of vulnerabilities and faults in data integrity in API-Restful	Selected vulnerabilities in API-Restful handle in this thesis
2. Describe a set of security-aware mutation operators for evaluating safeguarding data integrity in API-Restful	Description of the mutation operators
3. Develop a set of security-aware mutation operators for penetration testings in two Python API-Restful Frameworks	Source code of the secure-aware mutation operators
5. Evaluate the proposed security-aware mutation operators in RESTful APIs	Report about the performance of the create operators against tools from the literature.

Table 1.1: Specific objectives and expected results

1.6 Limitations

This research proposal considers the next limitations:

1. This thesis will only cover the most common vulnerabilities of the API-Restful in 2023
2. The focus of the thesis will be frameworks Django Rest and Flask in Python language, because they are highly used on different applications
3. The mutation operators are code-based under the idea to create mutants from the source code

1.7 Impact of this proposal

1. The tools that will be developed in this work, it is support for the specification of specific mutation-aware mutation operators

2. The secure-aware mutation operators can be extended a different web programming languages

3. The work will be an experience about mutation testing and security with the idea to design new tools for mutation testing

Chapter 2

State of the art

This chapter will address the literature review that was conducted for this proposal, which includes three main topics, the first aims to answer What are the challenges in mutation testing, the second What are the trends on testing in RESTful APIs, and the last What is being explored in terms of security in RESTful APIs?

2.1 Mutation testing

In 1972 Richar Lipson introduced the mutation testing[8] to detect fails through applying mutation operators, which is a specified change in the code. After, Acree[9] proposed a method to evaluate the effectiveness of the testing tools using mutation operators, afterwards in 1981, Hucks[10] introduced a collection of mutation operators in COBOL. Then, Morel[11] created a few of strategies to create operator based on the inference of errors, which are difficult to observe directly. In the 80s, some authors created a set of tools, compilers and procedures to apply mutation testing[12, 13]. From the decade of the 90 and 2000 begins the creation of some mutation testing tools for modern languages [14, 15, 16].

Frankl[17, 18] evaluated the effectiveness of the mutation testings considering use cases and data-flow, after introducing metrics such as covering level of the operators in the software components [19, 20, 21]. Also, Kakarlaa[22] tested vulnerabilities in the mutation operators product of the size of the data and the programming language.

In Security testing, Mouelhi[23] evaluated the adaptability of the mutation operators in the testing of security policies in the three layers applications. But, this analysis is empiric and it is possible only observe changes in the user roles modules. Based on this work, some authors developed tools such as Magneto[24] an open source software to automatic evaluation of security vulnerabilities and the Upgrade evaluation tool[25].

Loise[4] introduced 15 operators to the detection of the security fails in Java, these operators consist in modify the arguments of the functions, change the evaluation of conditional structures, vary mathematical operations and remove control structures in user authentication modules and database connections. The design of the operators uses known vulnerabilities such as injection SQL. Loise showed that these operators can reveal some kinds of vulnerabilities in some open source applications. But, there is evidence of the missing of a security evaluation criteria in open source projects.

In recent years, the development of tools in mutation testing has been increasing, MutAPI[26] generate mutation operators from common vulnerabilities in Java getting a precision of the 78% in the detection of potential security problems. A similar tool is available for Python[27] and specific libraries[28]. In common these tools can detect vulnerabilities, that are unexpected for the developers. However there are some limitations in mutation testing, one of them is the difference between artificial and real faults detected by mutation testing[29] this is because some of the vulnerabilities are not realistic because their occurrence is really low, hence it is not easy to know if they are really going to occur in a real scenario. Other limitation is the selection of the mutation strategies, because it is highly possible to have a huge set of equivalent mutants who produces a low performance of this technique of testing[30].

Some tools include artificial intelligence techniques like deep learning[31] using probabilistic methods who allows for a more consistent decision on whether a mutant is killed or not. Other works [32, 33, 34] show the effectiveness of using machine learning and deep learning techniques to select mutants in order to reduce redundancy and improve test coverage.

From this preliminary revision of the state of the art, the mutation operators work only on specific cases, this is an evidence of the need to contribute with a formal framework for the generation of generic testing security mutation operators.

2.2 Testing in RESTful API

The software tests on Restful API[35] are varied and evaluate different aspects of information quality and security metrics that need to be reviewed. There are some strategies to test them, based on regression[36], unit tests[35], White-Box[37], property based[38] and others.

To apply the tests on RESTFul API[39], the actions to be performed must be evaluated, including checking that the HTTP status codes are correct, for example that the response when there is a forbidden access is correct, verifying that the payload is in the correct format and that the responses are given in a reasonable time. Also, certain scenarios such as the happy path, negative situations with correct and incorrect input and security in terms of data encryption, data security and the correct configuration of access permissions must be considered.

Challenges to solve in these tests have been identified[40], among which is the generation of a framework to generate effective unit tests to validate the quality of this type of applications, in addition, limitations have been found corresponding to the description of how a RESTful API works, since there are several standards, among which are XML, JSON or OPENAPI. Some work aims to solve these challenges, as in the case of unit test generation[41, 42, 43, 44] and improving the coverage[45].

2.3 Security in RESTful-API

The OpenAPI Specification(OAS)[1] provides an implementation-independent specification on how APIs should work; however, potential vulnerabilities have been identified, such as identifying sensitive information in data, detecting the level of risk that using RestFul-APi

[1]https://swagger.io/specification/, September 2023

could pose in the transmission of sensitive information, and how to determine the level of exposure of APIs, as of 2021 were identified in the literature review on these risks that this is a challenging situation for information security[46].

Many techniques are currently used in RestFul-API security[47], including data encryption using TLS, authentication using the OAuth protocol, use of security certificates, specification of roles to restrict access to data, decentralization of APIs for different access roles, and the use of software patterns that allow access to information to be delegated to different users and facilitate control of the information exchanged.

The OWASP API Security Risks is an important standard to develop applications, some works considered the implications of the creation of web applications[48, 49, 50] which have found that there are problems in using proper authorization to access data, differentiating sensitive data from non-sensitive data, and in evaluating how the calls made to RESTFul APIs are applied.

The most common forms of attack[51] are the Structured Query Language (SQL) Injection Attack, a Man-in-the-Middle Attack (MitM), Phishing Attack, Denial of Service Attack (Dos), which can represent a risk to the information handled by the RESTFul API, some statistics indicate an increasing number of attacks to this type of applications.

For this reason, some techniques have been developed[52, 53] that developers should follow to mitigate the risks related to unauthorized access to information, including the use of secure protocols such as SSL, use of hashing algorithms, application of access rules and user roles, which should be validated in the respective software tests.

Therefore, organizations must consider security as a fundamental pillar in the development of this type of applications, since many vulnerabilities are not the result of attacks, but of omissions on the part of the software development team or of the tools used for its codification, taking into account that these are the most used information exchange mechanism between applications, so it is a critical component to be considered.

In recent years, different techniques have been proposed for the evaluation of security in RESTFUl APIs, including automatic black box testing[54] , automatic penetration testing[55] , model-based[56] and using Machine Learning[57, 58, 59] . As can be seen, security testing aims to automate its design and execution to improve the coverage of possible vulnerabilities that have been generated in the development process or in the interaction of different components.

Some challenges have been identified in the security of RESTFul APIs[60] , including: How to handle interactions with third-party components? How to ensure good development practices, such as emergent development styles? How to apply an appropriate access configuration? How to test data access with white-box testing? The latter because black box testing is insufficient to identify vulnerabilities.

2.4 Identified Challenges

According to the literature review we identified some current challenges in the interception between mutation testing and security in the development of RESTFul APIs.

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.

Therefore, it is identified that mutation testing can be a strategy to contribute to information security in RESTFul APIs by providing a framework for assessing test coverage in terms of vulnerabilities.

Chapter 3

Methodology

The methodology involves three major phases for the execution of this project, each of which has a duration of one year.

1. Review of vulnerabilities in RESTful API, this is focused on evaluating and analyzing their current state and consequently selecting the ones that will be used in this thesis.

2. Description of the secure-aware mutation operators on RESTful API.

3. Prototype implementation and testing, tests will be applied on selected open source projects and the effectiveness of the mutation operators will be evaluated.

3.1 Researching methodology

3.1.1 Review of vulnerabilities in RESTful APIs

For the research review, the Snowball methodology[61] will be used, which will start with the most recent state of the art reviews on mutation testing[62], testing challenges for RESTful APIs[40], and software security testing[35], which allow to have a compilation of different works related to this thesis. Subsequently, the most cited papers of these state of the art reviews will be identified and we will reach papers on specific vulnerabilities, this review system will allow structuring the most common vulnerabilities and the strategies that are being used for their management.

Within these articles, special emphasis will be placed on works related to vulnerabilities reported by OSWAP for the year 2023, whose top 10 are grouped as follows:

1. Broken object-level authorization: An intruder can break access to data by modifying the ID or UUID of the exchange data.

2. Broken authentication: Abuse of authentication tokens to obtain the identity of other users.

3. Unrestricted resource consumption: Abuse of resources such as bandwidth, CPU, memory and space.

4. Broken authorization at the role level: Lack of separation of groups and roles accessing a RESTful API.

5. Unrestricted access to sensitive business flows: Some automated functions can be exploited by intruders.

6. Server-side request forgery (SSRF): sending information to an unexpected destination.

7. Security misconfiguration: security risks related to the incorrect application of security policies in an API.

8. Inadequate inventory management: exposure of obsolete or insecure endpoints due to poor API management.

9. Insecure API consumption: cost of trust in date with any validation or verification of security policies, an attacker can impersonate a known endpoint and demand sensitive information.

3.1.2 Description of the mutation operators

Once the vulnerabilities to be worked on in this work have been identified, the mutation operators will be specified in the following steps.

1. Strategy to introduce the vulnerability in the source code.

2. Variations of the mutation operator to produce a vulnerability effect.

3. Analysis of possible redundant mutants produced by the mutation operator.

3.1.3 Prototype implementation and testing

For the implementation of the prototype, a TDD methodology[63] will be applied, which will allow to generate the test cases from the description of the mutation operators and thus guarantee that they will produce the desired effect in the software to be tested.

3.2 Activities

3.2.1 Phase I: Review of vulnerabilities in RESTful APIs

1. Identify elements of the 10 most common vulnerabilities in 2023 in Restful APIs

2. Identify strategies for handling these software vulnerabilities

3. Determine the elements used in software testing for the assessment of these vulnerabilities

3.2.2 Phase II: Description of the mutation operators

1. Describe how mutation is applied to introduce vulnerability.
2. Specify the different cases of variation of the mutation operator.
3. Determine how to apply modifications to the source code to apply the mutation operator.

3.2.3 Phase III: Prototype implementation and testing

1. Selected case studies on Python language frameworks.
2. Coding mutation operators in a mutation testing tool for Python.
3. Analyze coverage and redundancy metrics of applied operators.

Chapter 4

Expected Results

Mutation testing offers an interesting option to evaluate potential vulnerabilities in software, in the case of RESTFul APIs there is the possibility to evaluate how access to sensitive data can be managed. However, there are certain challenges related to the fact that in the development cycle problems have been identified in the implementation of OSWAP recommendations and that mutation testing can be time consuming, so it is necessary to specify mutation operators that can address these problems and at the same time be feasible within the development cycle.

This work aims to specify generic mutation operators that can be applied in different software testing tools to analyze the quality of security in this type of data exchange applications.

Bibliography

[1] T. Nelson, N. Danas, T. Giannakopoulos, and S. Krishnamurthi, "Synthesizing mutable configurations: Setting up systems for success," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)*, pp. 81–85, 2019.

[2] M. Büchler, "Security testing with fault-models and properties," in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, pp. 501–502, 2013.

[3] I. Kravets and D. Tsafrir, "Feasibility of mutable replay for automated regression testing of security updates," in *Runtime Environments/Systems, Layering, & Virtualized Environments workshop (RESoLVE)*, 2012.

[4] T. Loise, X. Devroey, G. Perrouin, M. Papadakis, and P. Heymans, "Towards security-aware mutation testing," in *2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, IEEE, Mar. 2017.

[5] H. Riggs, S. Tufail, I. Parvez, M. Tariq, M. A. Khan, A. Amir, K. V. Vuda, and A. I. Sarwat, "Impact, vulnerabilities, and mitigation strategies for cyber-secure critical infrastructure," *Sensors*, vol. 23, p. 4060, Apr. 2023.

[6] Y. Wang, J. Yao, and X. Yu, "Information security protection in software testing," in *2018 14th International Conference on Computational Intelligence and Security (CIS)*, pp. 449–452, 2018.

[7] Sattam J Alharbi and T. Moulahi, "Api security testing: The challenges of security testing for restful apis," *International Journal of Innovative Science and Research Technology*, 2023.

[8] R. Lipton, "Fault diagnosis of computer programs. student report," *Carnegie Mellon University*, vol. 2, p. 2, 1971.

[9] A. T. Acree, *On Mutation*. PhD thesis, Department of Computer Science, USA, 1980.

[10] J. M. Hanks, *Testing Cobol Programs by Mutation*. phdthesis, Georgia Institute of Technology, Atlanta, Georgia, 1980.

[11] L. J. Morell, *A Theory of Error-Based Testing*. PhD thesis, Department of Computer Science, USA, 1983.

[12] A. J. Offutt and R. A. Demillo, *Automatic Test Data Generation*. PhD thesis, Department of Computer Science, USA, 1988.

[13] A. J. Offutt and K. N. King, "A fortran 77 interpreter for mutation analysis," in *Papers of the Symposium on Interpreters and interpretive techniques 87*, ACM Press, 1987.

[14] R. A. DeMilli and A. J. Offutt, "Constraint-based automatic test data generation," *IEEE Transactions on Software Engineering*, vol. 17, no. 9, pp. 900–910, 1991.

[15] R. A. DeMillo, E. W. Krauser, and A. P. Mathur, "Compiler-integrated program mutation," in *[1991] Proceedings The Fifteenth Annual International Computer Software Applications Conference*, pp. 351–356, 1991.

[16] J. C. Maldonado, M. E. Delamaro, S. C. P. F. Fabbri, A. S. Simão, T. Sugeta, A. M. R. Vincenzi, and P. C. Masiero, "Proteum: A family of tools to support specification and program testing based on mutation," in *Mutation Testing for the New Century*, pp. 113–116, Springer US, 2001.

[17] P. G. Frankl and S. N. Weiss, "An experimental comparison of the effectiveness of the all-uses and all-edges adequacy criteria," in *Proceedings of the symposium on Testing, analysis, and verification - TAV4*, ACM Press, 1991.

[18] P. Frankl and S. Weiss, "An experimental comparison of the effectiveness of branch testing and data flow testing," *IEEE Transactions on Software Engineering*, vol. 19, no. 8, pp. 774–787, 1993.

[19] A. J. Offutt, J. Pan, K. Tewary, and T. Zhang, "An experimental evaluation of data flow and mutation testing," *Software: Practice and Experience*, vol. 26, pp. 165–176, Feb. 1996.

[20] P. G. Frankl, S. N. Weiss, and C. Hu, "All-uses vs mutation testing: An experimental comparison of effectiveness," *Journal of Systems and Software*, vol. 38, pp. 235–253, Sept. 1997.

[21] P. G. Frankl and O. Iakounenko, "Further empirical studies of test effectiveness," in *Proceedings of the 6th ACM SIGSOFT international symposium on Foundations of software engineering - SIGSOFT 98/FSE-6*, ACM Press, 1998.

[22] S. Kakarla, S. Momotaz, and A. S. Namin, "An evaluation of mutation and data-flow testing: A meta-analysis," in *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*, IEEE, Mar. 2011.

[23] T. Mouelhi, Y. L. Traon, and B. Baudry, "Mutation analysis for security tests qualification," in *Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION (TAICPART-MUTATION 2007)*, IEEE, Sept. 2007.

[24] L. Thomas, W. Xu, and D. Xu, "Mutation analysis of magento for evaluating threat model-based security testing," in *2011 IEEE 35th Annual Computer Software and Applications Conference Workshops*, IEEE, July 2011.

[25] O. Ozdemir, T. S. Ingec, T. Erdinc, A. Roy, and M. Durran, "Upgrade verification tool," Mar. 12 2019.

[26] M. Wen, Y. Liu, R. Wu, X. Xie, S.-C. Cheung, and Z. Su, "Exposing library API misuses via mutation analysis," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, IEEE, May 2019.

[27] Z. Zhang, H. Zhu, M. Wen, Y. Tao, Y. Liu, and Y. Xiong, "How do python framework APIs evolve? an exploratory study," in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, IEEE, Feb. 2020.

[28] M. Kechagia, X. Devroey, A. Panichella, G. Gousios, and A. van Deursen, "Effective and efficient API misuse detection via exception propagation and search-based testing," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis - ISSTA 2019*, ACM Press, 2019.

[29] M. Ojdanic, A. Garg, A. Khanfir, R. Degiovanni, M. Papadakis, and Y. L. Traon, "Syntactic versus semantic similarity of artificial and real faults in mutation testing studies," *IEEE Transactions on Software Engineering*, vol. 49, pp. 3922–3938, July 2023.

[30] R. Pitts, "Mutant selection strategies in mutation testing," in *2023 International Conference on Code Quality (ICCQ)*, IEEE, Apr. 2023.

[31] F. Tambon, F. Khomh, and G. Antoniol, "A probabilistic framework for mutation testing in deep neural networks," *Information and Software Technology*, vol. 155, p. 107129, Mar. 2023.

[32] M. R. Naeem, T. Lin, H. Naeem, F. Ullah, and S. Saeed, "Scalable mutation testing using predictive analysis of deep learning model," *IEEE Access*, vol. 7, pp. 158264–158283, 2019.

[33] D. Mao, L. Chen, and L. Zhang, "An extensive study on cross-project predictive mutation testing," in *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*, IEEE, Apr. 2019.

[34] Z. Tian, J. Chen, Q. Zhu, J. Yang, and L. Zhang, "Learning to construct better mutation faults," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ACM, Oct. 2022.

[35] A. Golmohammadi, M. Zhang, and A. Arcuri, "Testing restful apis: A survey," *ACM Trans. Softw. Eng. Methodol.*, aug 2023.

[36] P. Godefroid, D. Lehmann, and M. Polishchuk, "Differential regression testing for REST APIs," in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ACM, July 2020.

[37] A. Arcuri, "Automated black- and white-box testing of restful apis with evomaster," *IEEE Software*, vol. 38, no. 3, pp. 72–78, 2021.

[38] S. Karlsson, A. Čaušević, and D. Sundmark, "Quickrest: Property-based test generation of openapi-described restful apis," in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, pp. 131–141, 2020.

[39] B. De, *API Testing Strategy*. Apress, 2017.

[40] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal, and D. Mishra, "RESTful API testing methodologies: Rationale, challenges, and solution directions," *Applied Sciences*, vol. 12, p. 4369, Apr. 2022.

[41] A. Arcuri, "Restful api automated test case generation with evomaster," *ACM Transactions on Software Engineering and Methodology*, vol. 28, jan 2019.

[42] S. Segura, J. A. Parejo, J. Troya, and A. Ruiz-Cortés, "Metamorphic testing of restful web apis," in *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, (New York, NY, USA), p. 882, Association for Computing Machinery, 2018.

[43] A. Arcuri, "Restful api automated test case generation," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pp. 9–20, 2017.

[44] E. Viglianisi, M. Dallago, and M. Ceccato, "Resttestgen: Automated black-box testing of restful apis," in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, pp. 142–152, 2020.

[45] H. Wu, L. Xu, X. Niu, and C. Nie, "Combinatorial testing of restful apis," in *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, (New York, NY, USA), p. 426–437, Association for Computing Machinery, 2022.

[46] R. Sun, Q. Wang, and L. Guo, "Research towards key issues of api security," in *Cyber Security* (W. Lu, Y. Zhang, W. Wen, H. Yan, and C. Li, eds.), (Singapore), pp. 179–192, Springer Nature Singapore, 2022.

[47] P. Siriwardena, *Advanced API Security*. Apress, 2020.

[48] M. Idris, I. Syarif, and I. Winarno, "Development of vulnerable web application based on owasp api security risks," in *2021 International Electronics Symposium (IES)*, pp. 190–194, 2021.

[49] M. Idris, I. Syarif, and I. Winarno, "Web application security education platform based on owasp api security project," *EMITTER International Journal of Engineering Technology*, vol. 10, pp. 246–261, Dec. 2022.

[50] C. Cheh and B. Chen, "Analyzing openapi specifications for security design issues," in *2021 IEEE Secure Development Conference (SecDev)*, pp. 15–22, 2021.

[51] B. Modi, U. Chourasia, and R. Pandey, "Design and implementation of RESTFUL API based model for vulnerability detection and mitigation," *IOP Conference Series: Materials Science and Engineering*, vol. 1228, p. 012010, Mar. 2022.

[52] A. Munsch and P. Munsch, "The future of API (application programming interface) security: The adoption of APIs for digital communications and the implications for cyber security vulnerabilities," *Journal of International Technology and Information Management*, vol. 29, pp. 24–45, Jan. 2021.

[53] Sattam J Alharbi and T. Moulahi, "Api security testing: The challenges of security testing for restful apis," *International Journal of Innovative Research in Science Engineering and Technology*, 2023.

[54] D. Corradini, M. Pasqua, and M. Ceccato, "Automated black-box testing of mass assignment vulnerabilities in restful apis," in *International Conference on Software Engineering (ICSE 2023)*, arXiv, 2023.

[55] N. Auricchio, A. Cappuccio, F. Caturano, G. Perrone, and S. P. Romano, "An automated approach to web offensive security," *Computer Communications*, vol. 195, pp. 248–261, Nov. 2022.

[56] B. O. EMEKA, S. HIDAKA, and S. LIU, "A practical model driven approach for designing security aware RESTful web APIs using SOFL," *IEICE Transactions on Information and Systems*, vol. E106.D, pp. 986–1000, May 2023.

[57] M. C. Ghanem and T. M. Chen, "Reinforcement learning for efficient network penetration testing," *Information*, vol. 11, p. 6, Dec. 2019.

[58] Z. Hu, R. Beuran, and Y. Tan, "Automated penetration testing using deep reinforcement learning," in *2020 IEEE European Symposium on Security and Privacy Workshops*, IEEE, Sept. 2020.

[59] J. Schwartz and H. Kurniawati, "Autonomous penetration testing using reinforcement learning," 2019.

[60] L. Zhong, "A survey of prevent and detect access control vulnerabilities," 2023.

[61] C. Noy, "Sampling knowledge: The hermeneutics of snowball sampling in qualitative research," *International Journal of Social Research Methodology*, vol. 11, pp. 327–344, Oct. 2008.

[62] M. Papadakis, M. Kintis, J. Zhang, Y. Jia, Y. L. Traon, and M. Harman, *Mutation Testing Advances: An Analysis and Survey*, vol. 112. Elsevier Inc., 1 ed., 2019.

[63] L. Williams, E. M. Maximilien, and M. Vouk, "Test-driven development as a defect-reduction practice," in *14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003.*, pp. 34–45, IEEE, 2003.