Mining REST APIs for Potential Mass Assignment Vulnerabilities

Arash Mazidi
Technische Universität Clausthal
Germany
arash.mazidi@tu-clausthal.de

Davide Corradini
University of Verona
Italy
davide.corradini@univr.it

Mohammad Ghafari
Technische Universität Clausthal
Germany
mohammad.ghafari@tu-clausthal.de

ABSTRACT

REST APIs have a pivotal role in accessing protected resources within cyberspace. Despite the availability of security testing tools, mass assignment vulnerabilities are common, yielding unauthorized access to sensitive data. We propose a lightweight approach to mine the REST API specifications and identify operations and attributes that are prone to mass assignment. We conducted a preliminary study on 100 APIs and found 25 prone to this vulnerability. We confirmed nine real vulnerable operations in six open-source APIs.

CCS CONCEPTS

• **Security and privacy → Web application security**.

KEYWORDS

Mass assignment, REST API security, specification mining

ACM Reference Format:
Arash Mazidi, Davide Corradini, and Mohammad Ghafari. 2024. Mining REST APIs for Potential Mass Assignment Vulnerabilities. In *28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024), June 18–21, 2024, Salerno, Italy*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/xxx

1 INTRODUCTION

REST APIs enable seamless data exchange and functionality integration across different systems. The pivotal role of these APIs in today's software industry has made them an attractive target for attackers. For instance, a recent API vulnerability disclosed 1.8 million user accounts from an insurance company [4]. Additionally, a security breach in the AWS S3 bucket of a digital scheduling platform exposed the personally identifiable information (PII) of 3.7 million user accounts [12]. Furthermore, a major social media platform reported a breach in its API from late 2021 into 2022, revealing the PII of 5.4 million user accounts. The vulnerability originated from an API designed to help users in finding others [13].

Mass assignment is a critical but overlooked vulnerability in REST APIs. It occurs when REST APIs allow the unintended modification of attributes, often yielding unauthorized access to sensitive data. This vulnerability arises due to an incorrect configuration of widely used REST API frameworks that typically facilitate automatic binding between input data fields and the internal data representation, such as database columns.

The support for identifying mass assignment vulnerabilities in REST APIs is limited. Akto [23] and RestTestGen [24] are two examples of tools for detecting mass assignment vulnerabilities in REST APIs. RestTestGen is an automated black-box testing tool, and Akto is semi-automated. Nonetheless, existing tools evaluate a *running* API, and developers have no support to uncover mass assignment vulnerabilities in earlier development stages.

We present LightMass, a tool for mining API endpoints and attributes prone to mass assignment vulnerabilities in REST APIs. Unlike existing tools that interact with a running API, LightMass merely relies on the API specification; therefore, it draws developers' attention to potential mass assignment vulnerabilities as early as the API's specification is known. In particular, LightMass inspects operations that handle similar sets of attributes (assuming they handle the same data model), and compares the attributes that a `GET` operation read and those that a `POST`, `PUT`, or `PATCH` operation writes to. When a `GET` operation has more attributes than the other operation (i.e., `POST`, `PUT`, or `PATCH`), the attributes that are only present in the `GET` operation are considered to be *read-only*, and therefore, these attributes are candidates for mass assignment vulnerabilities.

We conducted a preliminary study on 100 APIs and found 25 candidate APIs (115 endpoints and 133 operations) prone to mass assignment vulnerabilities. We examined potential vulnerabilities in six APIs for which we could access the source code and confirmed the presence of nine vulnerable operations.

In summary, LightMass identifies operations that fulfill the necessary conditions for mass assignment vulnerabilities for later in-depth analysis. The fast and simple nature of its approach is helpful in several scenarios, such as (i) steering code reviewers' focus on potential issues; (ii) enabling tools such as Akto to perform automated testing of mass assignment vulnerabilities; and (iii) mining API specifications at large and estimating the potential for mass assignment vulnerabilities in the wild. LightMass is open-source and publicly available on GitHub.[1]

The rest of this paper is organized as follows. We provide background information about RESTful APIs and the mass assignment vulnerability in Section 2. We introduce our approach to identify potential mass assignment vulnerabilities in Section 3. We present our evaluation in Section 4. We present related work in Section 5, and conclude the paper in Section 6.

2 BACKGROUND

This section introduces REST APIs and the OpenAPI standard for writing API specifications. Subsequently, we introduce the mass assignment vulnerability.

EASE 2024, June 18–21, 2024, Salerno, Italy
2024. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/xxx

[1] https://github.com/arash-mazidi/LightMass

```yaml
openapi: 3.0.1
info:
  title: Task Management
  description: Task retrieving, creating, and so on.
  version: 1.0.0
  license:
    name: Creative Commons Attribution 3.0
    url: http://creativecommons.org/licenses/by/3.0/
servers:
  url: http://localhost:8080
paths:
  /tasks:
    get:
      summary: Get All Tasks
      responses:
        "200":
          description: Successful response
          content:
            application/json:
              schema:
                properties:
                  title:
                    type: string
                  assignee:
                    type: string
                  status:
                    type: "boolean
    post:
      summary: Create a Task
      requestBody:
        content:
          application/json:
            schema:
              properties:
                title:
                  type: string
                assignee:
                  type: string
      responses:
        "201":
          description: Created successfully
```

Listing 1: Excerpt of specification for a REST API

2.1 REST APIs and OpenAPI specifications

REST APIs are web APIs that adhere to the REST (REpresentational State Transfer) architectural style. They offer a consistent interface for creating, reading, updating, and deleting resources. HTTP URIs identify resources, and operations on resources are typically associated with HTTP methods such as `POST`, `GET`, `PUT` (or `PATCH`), and `DELETE` to, respectively, *create*, *read*, *update* or *delete* resources.

Every REST API should follow the OpenAPI standard to describe the API's structure and behavior. In particular, REST APIs include an OpenAPI specification file, typically structured in JSON or YAML format, which describes the endpoints, operations and their attributes, as well as the request and response schemas.

Listing 1 presents a snippet of the OpenAPI specification for a "task management" API. Following an initial header specifying versions, licenses, and the API's base URL, this specification features an array of paths representing the available URI endpoints in the API. In this example, the HTTP URI leading to a task resource is `/tasks` (line 12), and the HTTP operations `GET` and `POST` (lines 13 and 28) are utilized to retrieve the list of existing tasks and create a new task in the system, respectively. These operations have common attributes such as `title` (lines 22 and 35) and `assignee` (lines 24 and 37), which delineate the task's title and the person responsible for it.

2.2 Mass Assignment Vulnerability

Developers usually rely on frameworks to build REST APIs. These frameworks, such as Spring for Java, Flask for Python, Express.js for JavaScript, and Laravel for PHP, offer a suite of reusable components and features to facilitate REST API development. One of the features typically provided by these frameworks is called *auto-binding*, a mechanism that adopts naming conventions to automatically map input data in HTTP requests (i.e., attributes) to the backend data objects (e.g., database columns) when they share the same name. This feature is typically enabled by default for all attributes. A *mass assignment vulnerability*, also known as "object injection" or "auto-binding vulnerability", occurs when developers neglect to disable this feature for attributes that are meant to be read-only. Therefore, an attacker can add an extra attribute to an HTTP request (one that was not intended to be changed), and the auto-binding feature would automatically link that attribute to its corresponding database column. In principle, this attribute is neither part of the API specification nor the API documentation, and it should have not been processed. Nevertheless, the attacker who exploits this feature, will be able to manipulate and alter data in the database, posing a significant security risk.

For instance, consider the specification of the "Task Management" API shown in Listing 1. Suppose the JSON `task` object within an HTTP request maps to the `tasks` table in the database, which includes a (supposedly) read-only boolean column named `status` and two modifiable columns, namely `title` and `assignee`. If the REST framework lacks a proper configuration, it may automatically link an additional `status` attribute in a "create task" request to the corresponding `status` column in the `tasks` table. That is, an attacker could manipulate a request body of the `POST /tasks` operation by introducing an extra `status` attribute not specified in the OpenAPI specification. The framework would then automatically associate this additional attribute with the `status` column in the `tasks` table, allowing the attacker to overwrite the legitimate value in the database with the manipulated HTTP attribute value.

To prevent mass assignment, developers should blacklist read-only attributes from being auto-bound to the internal data representation of the API.

3 LightMass

We developed LightMass, a tool that takes an OpenAPI specification file as input and identifies candidate operations and attributes prone to mass assignment vulnerabilities. Figure 1 illustrates the LightMass workflow, and Listing 2 shows the corresponding procedure.

LightMass parses the API specification to identify existing endpoints, operations, and attributes. It relies on the Jackson library to parse the content. Subsequently, it resolves all the cross references (i.e., `$ref`),[2] ensuring that they are replaced with their actual definitions. Then, it navigates through the specification to access information about paths and operations (Listing 2, lines 9-11). The paths object contains details about each endpoint, and under each path, the supported HTTP operations, e.g., `GET`, `POST`, `PUT`, and

[2] In OpenAPI, one can define a component at one location in the specification document and reference (reuse) it in other places, reducing redundancy and making the document more maintainable.

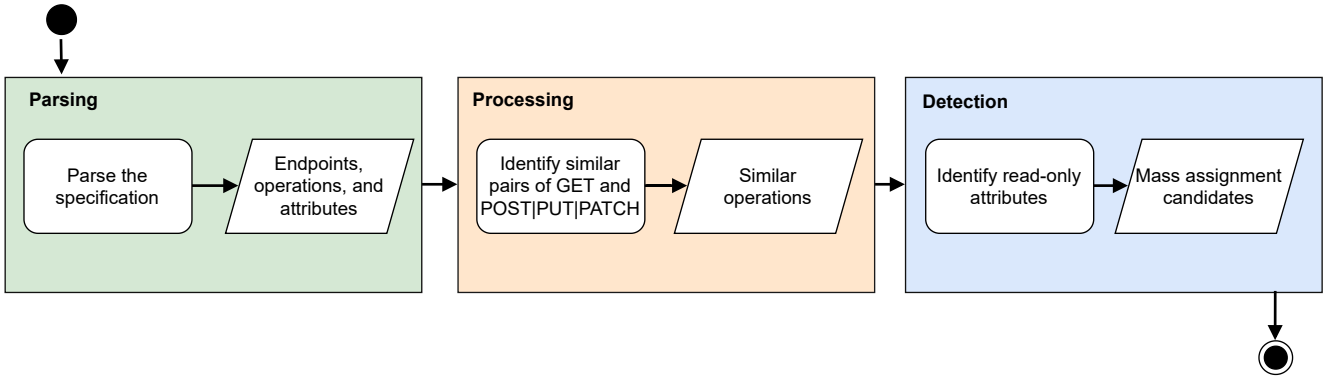

Figure 1: LightMass workflow

```
1  Procedure: LightMass
2   Input: OpenAPI specification
3   Output: Candidate operations and attributes for mass assignment
4
5   POST-PUT-PATCH ← {}
6   GET ← {}
7   MassList ← {}
8
9   For Each EndPoint in Specification.Endpoints
10     POST-PUT-PATCH ← EndPoint.POST|PUT|PATCH
11     GET ← EndPoint.GET
12
13  FOR Each X in GET
14     RES ← X.RESPOSE.Attributes
15     FOR Y in POST-PUT-PATCH
16         REQ ← Y.REQUEST.Attributes
17
18         IF |RES|>|REQ|
19             IF RES and REQ are Similar
20                 MassList ← (Y, RES - (RES ∩ REQ) )
21
22  Return MassList
23
24 EndProcedure
```

Listing 2: The procedure to find mass assignment candidates

PATCH are listed. LightMass extracts all attributes for each operation, which are found within both the request and response bodies, as well as other locations such as path, query, and header (lines 14 and 16). This phase is pivotal since mass assignment vulnerabilities often revolve around the manipulation of input attributes.

LightMass identifies similar operations based on similar attributes among operations. Firstly, to facilitate a uniform comparison, Porter's stemming algorithm [19] is employed to standardize attribute names, i.e., reducing attribute names to their core or root forms. Secondly, it utilizes the Jaccard coefficient to identify similar operations:

$$JaccSim(OP, GET) = \frac{|OP.REQ \cap GET.RES|}{|OP.REQ \cup GET.RES|}$$

Therefore, the similarity measure is the ratio of the number of shared (similar) attributes between two operations to the total number of their distinct attributes (line 20). Specifically, *OP.REQ* comprises the attributes in the request body of a POST, PUT, or PATCH operation, and *GET.RES* is the set of attributes in the response body of a GET operation.

LightMass reports a potential vulnerability when (i) the similarity between two operations is at least 50%,[3] and (ii) the number of attributes in the response of a GET operation exceeds the number of attributes in the request of the other operation (POST, PUT, or PATCH). The additional attributes in the GET operation are supposed to be read-only, making them potential candidates for mass assignment vulnerabilities. In the end, LightMass provides a structured list of candidate endpoints, operations, and attributes prone to mass assignment vulnerabilities.

For instance, consider the API specification in Listing 1. With two attributes (`title` and `assignee` at lines 35 and 37) in the request body of the POST operation (line 28) and three attributes (`title`, `assignee`, and `status` at lines 22, 24, and 26) in the response body of the GET operation (line 13), the Jaccard similarity score is 0.66. The number of attributes in the GET operation exceeds that of the POST operation. Therefore, the `status` attribute in the response body, which is absent in the request body, is a candidate for mass assignment vulnerability.

> *It is noteworthy that an actual vulnerability exists only if enough protection measures are not in place. Therefore, LightMass' report serves as a guide for security analysts, developers, and testers in conducting further investigations. For example, they could verify that they have properly disabled the auto-binding feature for the attributes flagged as potentially vulnerable.*

4 EVALUATION

We applied LightMass to 100 APIs that we randomly collected from previous work [6], GitHub, the Google APIs, APIs Guru,[4] and EMB.[5]

Mining the OpenAPI specifications of these APIs uncovered 25 potentially vulnerable APIs listed in Table 1. Specifically, LightMass reported 495 candidate attributes for mass assignment distributed across 115 endpoints and 133 operations in 25 APIs.

[3] In practice, a vulnerability can exist even with just one extra attribute, but in our experience, a 50% threshold was practical to uncover actual vulnerabilities and avoid false positives. Nonetheless, it is possible to adjust the similarity threshold in each run if needed.
[4] https://apis.guru/
[5] https://github.com/EMResearch/EMB

API Name	Total		Flagged vulnerable		
	# Endpoints	# Operations	# Endpoints	# Operations	# Attributes
VAmPI	10	12	2	2	2
OWASP	4	10	2	2	4
Toggle	8	16	2	2	2
CRUD	1	4	1	2	2
Bookstore	3	5	1	1	1
StudentAPI	5	5	2	2	2
Search Console	7	11	1	1	2
Fitness	7	13	4	4	7
Calendar	22	37	7	10	18
My Business	40	50	9	9	22
Analytics	43	88	22	28	175
Classroom	34	61	17	17	27
YouTube	39	76	12	20	106
spacex-api	52	94	1	1	3
reservations-api	5	7	1	1	1
projectManagement	58	78	1	1	1
alertersystem	186	422	4	4	30
TransferService	3	3	1	1	16
CheckoutService	23	24	1	1	3
registry	20	35	10	10	10
sms	2	5	2	2	24
ats	4	5	1	1	13
autoscaling	65	130	1	1	1
hub	20	26	1	1	3
files	134	222	9	9	20
Total	**795**	**1439**	**115**	**133**	**495**

Table 1: LightMass report for 25 REST APIs

Unfortunately, there is no golden dataset for mass assignment vulnerabilities in REST APIs. To evaluate whether these APIs are actually vulnerable, we should either test the APIs or examine their code. It is unethical to test APIs in production due to the potential risk of launching a successful attack. Hence, we compared LightMass and existing tools against six open-source APIs that we could set up and run locally. These APIs are listed in Table 2.

To identify existing tools for mass assignment detection and compare them with LightMass, we searched the literature and Google with a combination of keywords such as *mass assignment* and *detection, scanner, or analyzer*. We also extended our search to GitHub with keywords such as *mass assignment*, *object injection*, and *autobinding*. Upon obtaining a list of potential tools, we paid close attention to the repository descriptions, README files, and any available documentation to determine the relevance of every search result.

We identified a total of nine (semi-)automated tools. We eliminated two since they had not been updated since 2010, suggesting potential obsolescence. We scrutinized the remaining tools and discovered that five are designed for mass assignment detection in web applications. The two remaining tools, namely RestTestGen [24] and Akto [23], supported mass assignment detection in REST APIs.

Akto cannot automatically identify mass assignment vulnerabilities, so we had to manually input the potential vulnerable endpoints and attributes.[6] Therefore, we applied RestTestGen to the six APIs in Table 2 to build our ground truth for mass assignment vulnerabilities.

Table 3 lists the vulnerability reports by each tool. Akto and RestTestGen provided the same results, whereas LightMass flagged one more attribute prone to mass assignment vulnerability in the VAmPI API. We looked at the source code of VAmPI to learn about the extra attribute (named `owner`) that LightMass had flagged. Upon inspection, we found that the *book* model in the VAmPI API had predefined fields allowed to be set while creating a new book instance, namely `book_title`, `secret_content`, and `user_id`. These were the only permissible fields for setting while creating a new book. Any attempt to include an additional field in the request, such as `owner`, would be unsuccessful due to a server-side restriction and input validation. Therefore, the extra field that LightMass flagged for VAmPI API was a false positive.

It is important to note that the obtained results for these six case studies cannot be generalized to the remaining 19 (unverified) APIs.

[6] We provided Akto with the output from LightMass and checked whether Akto flags them for mass assignment vulnerability or not.

API Name	Language	Description
VAmPI [26]	Python	A vulnerable API which includes all the OWASP top 10 vulnerabilities of APIs.
OWASP [18]	Java	An API vulnerable to broken object-level authorization, excessive data exposure, and mass assignment.
Toggle [25]	ASP.Net	It defines toggles for a list of services.
CRUD [7]	Node.js	A CRUD example with NodeJS, Sequelize, Swagger, and MySQL.
Bookstore [5]	Java	An API designed to expose the features to manage a book store.
StudentAPI [22]	Java	It is a vulnerable API intended for educational purposes, with a focus on addressing mass assignment vulnerabilities.

Table 2: The open-source APIs used as case studies

API Name	Akto	RestTestGen	LightMass
VAmPI	1	1	2
OWASP	4	4	4
Toggle	2	2	2
CRUD	2	2	2
Bookstore	1	1	1
StudentAPI	2	2	2

Table 3: The number of attributes flagged by each tool

In summary, the preliminary evaluation results are promising. Nonetheless, relying on LightMass as a standalone tool requires future studies. Particularly, how it performs in terms of false positives against APIs that are not vulnerable remains for a future work investigation. It is noteworthy that an actual vulnerability exists only if enough protection measures are not in place. Hence, relying merely on the specification is not enough, and false positives are expected. Nonetheless, there is no tool support for early development stages in this domain, and we believe that LightMass draws developers' attention to this overlooked problem. In addition, as we experimented, LightMass enables Akto to act as a fully automated testing tool for mass assignment vulnerabilities. Akto is a popular and comprehensive API testing tool that does not support automated testing for mass assignment. It requires human interventions and input for suspected operations and attributes. Hence, LightMass in its current state enables the community to apply Akto as a fully automated tool to uncover true mass assignment vulnerabilities.

5 RELATED WORK

Gadient et al. [8] mined 9,714 Web APIs from 3,376 mobile apps, and found that in 500 apps, these APIs transmit embedded code (e.g., SQL and JavaScrip commands), exposing the app users and web servers to code injection attacks. In a follow-up study [9], they also discovered that API servers are usually misconfigured. They observed that on average every second server suffers from version information leaks, and worryingly, servers are usually set up once and never touched again, yielding severe security risks.

Atlidakis et al. [1] presented RESTler, a stateful REST API fuzzer that examines the OpenAPI specification. RESTler statically analyzes OpenAPI specification and creates and executes tests by deducing dependencies and examining the responses from previous test runs. They also demonstrated an extension of RESTler with active property checkers, which enables automatic testing and identification of breaches in adherence to these rules [2]. Godefroid et al. [10] conducted a study on the intelligent generation of data payloads within REST API requests, leveraging the OpenAPI specification. They showed that they can detect data-processing vulnerabilities in cloud services. They [11] also presented a method for automated differential regression testing of REST APIs aimed at identifying breaking changes between API versions by comparing the responses of various versions when given the same inputs to identify discrepancies and identifying regressions in these observed differences. Mirabella et al. [17] presented a deep learning model to predict the validity of test inputs in an API request before making the API call.

Mai et al. [16] introduced a tool designed to automatically create executable security test cases from misuse case specifications written in natural language. Reddy et al. [20] introduced an approach centered around sequence models and transformers to discern whether an API request is prone to injection attacks. Barabanov et al. [3] introduced an automated technique for identifying vulnerable endpoints to Insecure Direct Object Reference (IDOR) and Broken Object Level Authorization (BOLA) vulnerabilities which are related to the improper handling of object references, particularly in the context of authorization. This method involves establishing a mapping between attack methodologies and the properties in endpoints found within OpenAPI specifications.

The number of studies on mass assignment vulnerability is limited. Corradini et al. [6] developed an automated black-box testing approach to find mass assignment vulnerabilities in RESTful APIs. It uses EM clustering to group operations within the endpoints. Subsequently, abstract testing templates are instantiated to automatically generate interaction sequences, to exploit potential vulnerabilities. This tool requires the API to be in a running status for interaction with HTTP operations.

Park et al.[14] introduced an automated tool for generating exploits targeting PHP object injection vulnerabilities named FUGIO. Koutroumpouchos et al. [15] introduced ObjectMap, a customizable solution that identifies deserialization and object injection vulnerabilities in web applications using Java and PHP. Shcherbakov et al. [21] introduced SerialDetector, a taint-driven dataflow analysis technique to identify Object Injection Vulnerabilities (OIVs) patterns within .NET assemblies.

6 CONCLUSION

Mass assignment is a critical vulnerability in REST APIs. However, there is a lack of support for developers to identify this security risk in the early stages of API development. We introduced LightMass, a tool that mines REST API specifications for potential mass assignment vulnerabilities. It identifies operations and attributes that fulfill the necessary conditions for mass assignment vulnerabilities. LightMass is not dependent on an active API, and it can alert developers as soon as the API's specification is known. It also enables Akto, the popular open-source API testing tool, to execute fully automated API testing for mass assignment vulnerabilities.

REFERENCES

[1] Vaggelis Atlidakis, Patrice Godefroid, and Marina Polishchuk. 2019. Restler: Stateful rest api fuzzing. *In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 748–758.

[2] Vaggelis Atlidakis, Patrice Godefroid, and Marina Polishchuk. 2020. Checking security properties of cloud service REST APIs. *In 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, 387–397.

[3] Alexander Barabanov, Denis Dergunov, Denis Makrushin, and Aleksey Teplov. 2022. Automatic detection of access control vulnerabilities via API specification processing. *arXiv preprint arXiv:2201.10833*.

[4] Jason Beeferman. 2022. *Attack to Insurance APIs.* https://www.texastribune.org/2022/05/16/texas-insurance-data-breach/

[5] Bookstore. 2022. *Bookstore.* https://github.com/todo/Bookstore

[6] Davide Corradini, Michele Pasqua, and Mariano Ceccato. 2023. Automated Black-box Testing of Mass Assignment Vulnerabilities in RESTful APIs. *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE).*

[7] CRUD. 2023. *CRUD.* https://github.com/lucianopereira86/CRUD-NodeJS-Sequelize-Swagger-MySQL

[8] Pascal Gadient, Mohammad Ghafari, Marc-Andrea Tarnutzer, and Oscar Nierstrasz. 2020. Web APIs in Android through the Lens of Security. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 13–22. https://doi.org/10.1109/SANER48275.2020.9054850

[9] Pascal Gadient, Marc-Andrea Tarnutzer, Oscar Nierstrasz, and Mohammad Ghafari. 2021. Security Smells Pervade Mobile App Servers. In *Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (Bari, Italy) *(ESEM '21)*. https://doi.org/10.1145/3475716.3475780

[10] Patrice Godefroid, Bo-Yuan Huang, and Marina Polishchuk. 2020. Intelligent REST API data fuzzing. *In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 725–736.

[11] Patrice Godefroid, Daniel Lehmann, and Marina Polishchuk. 2020. Differential regression testing for REST APIs. *In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 312–323.

[12] Jonathan Greig. 2022. *Attack to digital scheduling platform.* https://www.zdnet.com/article/flexbooker-apologizes-for-breach-of-3-7-million-user-records-credit-card-information/

[13] Tim Keary. 2022. *Attack to social media platform.* https://venturebeat.com/security/twitter-breach-api-attack/

[14] Sunnyeo Park Daejun Kim, Suman Jana, and Sooel Son. 2022. FUGIO: Automatic Exploit Generation for PHP Object Injection Vulnerabilities. *In 31st USENIX Security Symposium (USENIX Security 22)*, 197–214.

[15] Nikolaos Koutroumpouchos, Georgios Lavdanis, Eleni Veroni, Christoforos Ntantogian, and Christos Xenakis. 2019. ObjectMap: Detecting insecure object deserialization. *In Proceedings of the 23rd Pan-Hellenic Conference on Informatics*, 67–72.

[16] Phu X Mai, Fabrizio Pastore, Arda Goknil, and Lionel C. Briand. 2019. MCP: A security testing tool driven by requirements. *In 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 55–58.

[17] A. Giuliano Mirabella, Alberto Martin-Lopez, Sergio Segura, Luis Valencia-Cabrera, , and Antonio Ruiz-Cortés. 2021. Deep learning-based prediction of test input validity for restful apis. *In 2021 IEEE/ACM Third International Workshop on Deep Learning for Testing and Testing for Deep Learning (DeepTest)*, 9–16.

[18] OWASP. 2023. *OWASP.* https://github.com/mattiasanti99/vulnerabilityOWASPproject

[19] Martin F Porter. 1980. An algorithm for suffix stripping. *Program* 14, 3 (1980), 130–137.

[20] A. Sujan Reddy and Bhawana Rudra. 2022. Detection of injections in API requests using recurrent neural networks and transformers. *International Journal of Electronic Security and Digital Forensics*, 638–658.

[21] Mikhail Shcherbakov and Musard Balliu. 2021. Serialdetector: Principled and practical exploration of object injection vulnerabilities for the web. *In Network and Distributed Systems Security (NDSS) Symposium.*

[22] StudentAPI. 2023. *StudentAPI.* https://github.com/arash-mazidi/StudentAPI

[23] The AKTO Team. 2023. *Instant, Open source API security → API discovery, automated business logic testing and runtime detection.* https://github.com/akto-api-security/akto

[24] The RESTTESTGEN Team. 2023. *RestTestGen: A tool and framework for automated black-box testing of RESTful APIs.* https://github.com/SeUniVr/RestTestGen

[25] Toggle. 2023. *Toggle.* https://github.com/pdonatilio/ToggleAPI

[26] VAmPI. 2023. *vampi.* https://github.com/erev0s/VAmPI