Finanziato dall'Unione europea
NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA

Fuzzing Web APIS
for Functional and Security Testing

Mariano Ceccato

mariano.ceccato@univr.it

This work has been done in collaboration mainly with **Davide Corradini** and **Michele Pasqua**

UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA

Outline

Introduction

- Problem definition and research challenges

Functional testing

- Nominal and error testing
- Enhancing REST API testing with NLP Techniques
- Deep Reinforcement Learning-Based REST API Testing

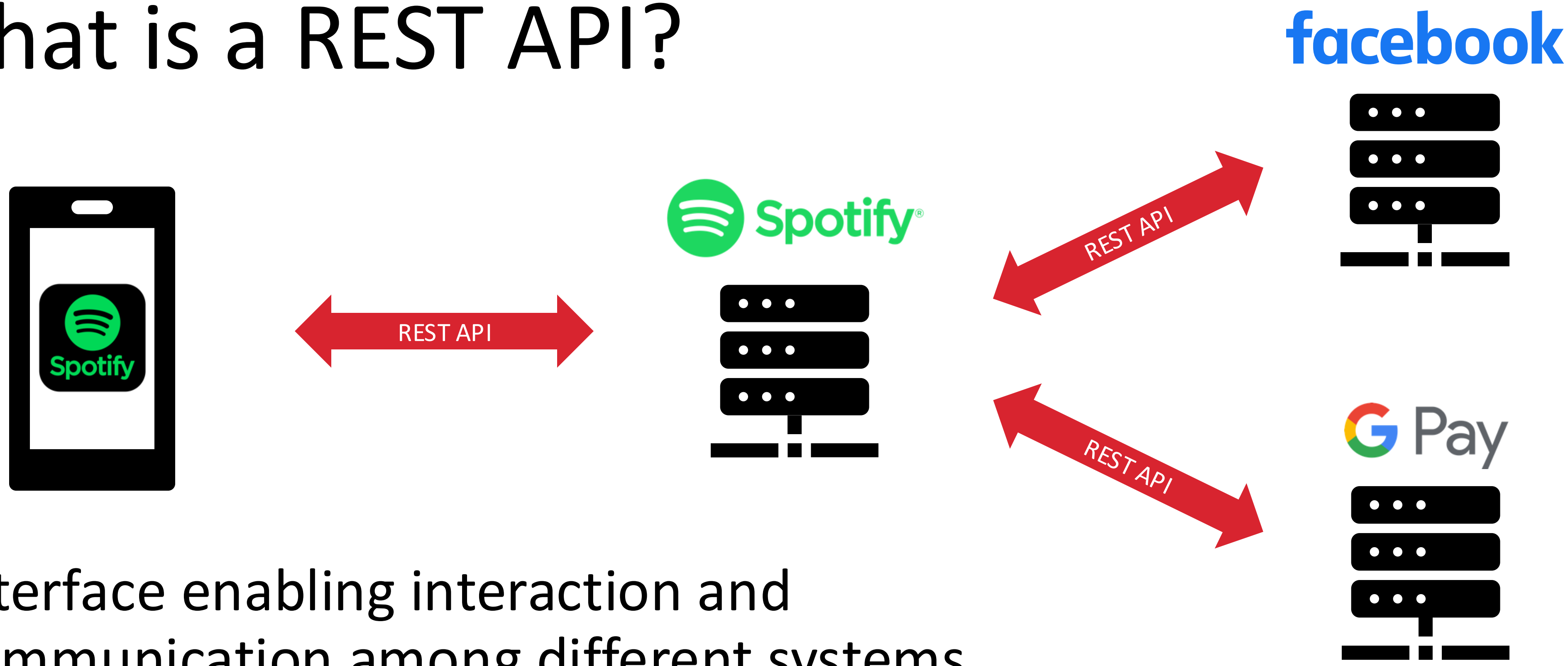
Security testing

- Security testing of mass assignment vulnerabilities

Reusable research tools

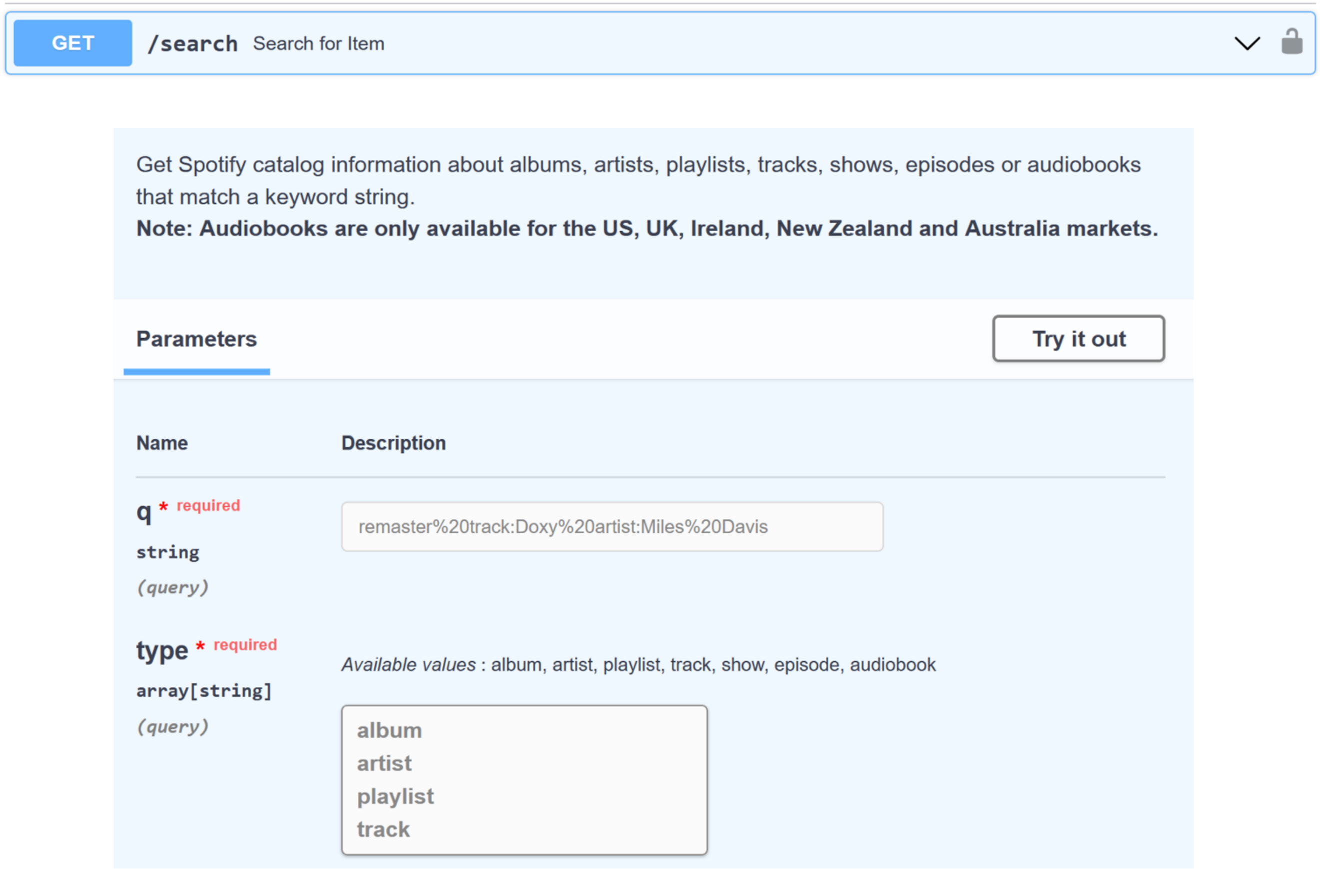
Conclusion

What is a REST API?



- Interface enabling interaction and communication among different systems
- Representational State Transfer Application Programming Interface

OpenAPI Specification

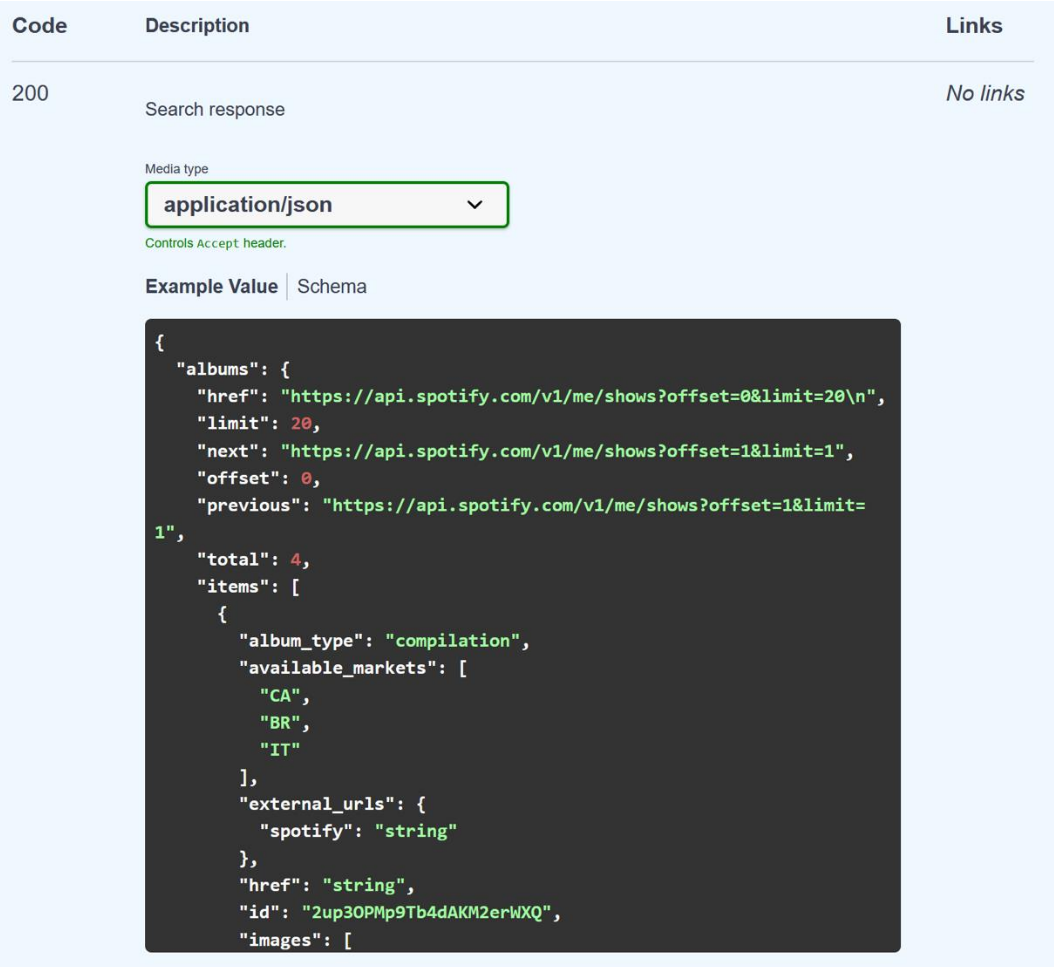


`/search` Search for Item

Get Spotify catalog information about albums, artists, playlists, tracks, shows, episodes or audiobooks that match a keyword string.
Note: Audiobooks are only available for the US, UK, Ireland, New Zealand and Australia markets.

Parameters

Name	Description
q * required `string` *(query)*	remaster%20track:Doxy%20artist:Miles%20Davis
type * required `array[string]` *(query)*	*Available values* : album, artist, playlist, track, show, episode, audiobook album artist playlist track



Code	Description	Links
200	Search response	*No links*

Media type

application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "albums": {
    "href": "https://api.spotify.com/v1/me/shows?offset=0&limit=20\n",
    "limit": 20,
    "next": "https://api.spotify.com/v1/me/shows?offset=1&limit=1",
    "offset": 0,
    "previous": "https://api.spotify.com/v1/me/shows?offset=1&limit=
1",
    "total": 4,
    "items": [
      {
        "album_type": "compilation",
        "available_markets": [
          "CA",
          "BR",
          "IT"
        ],
        "external_urls": {
          "spotify": "string"
        },
        "href": "string",
        "id": "2up3OPMp9Tb4dAKM2erWXQ",
        "images": [
```

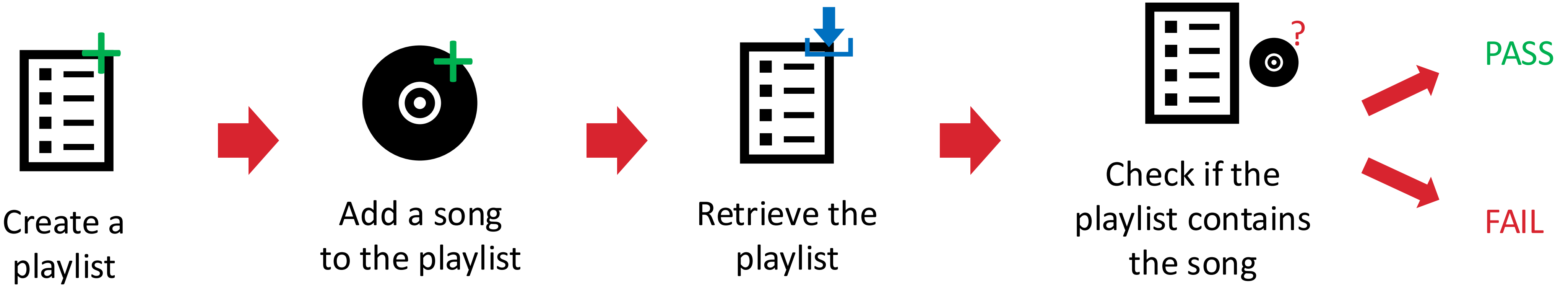
Problem definition

- The number of REST APIs grows larger and larger
- REST APIs contain programming defects and/or vulnerabilities
- Manual writing of test cases is limiting and costly

Solution:

Automated black-box test cases generation for REST APIs

Test case



Create a playlist

Add a song to the playlist

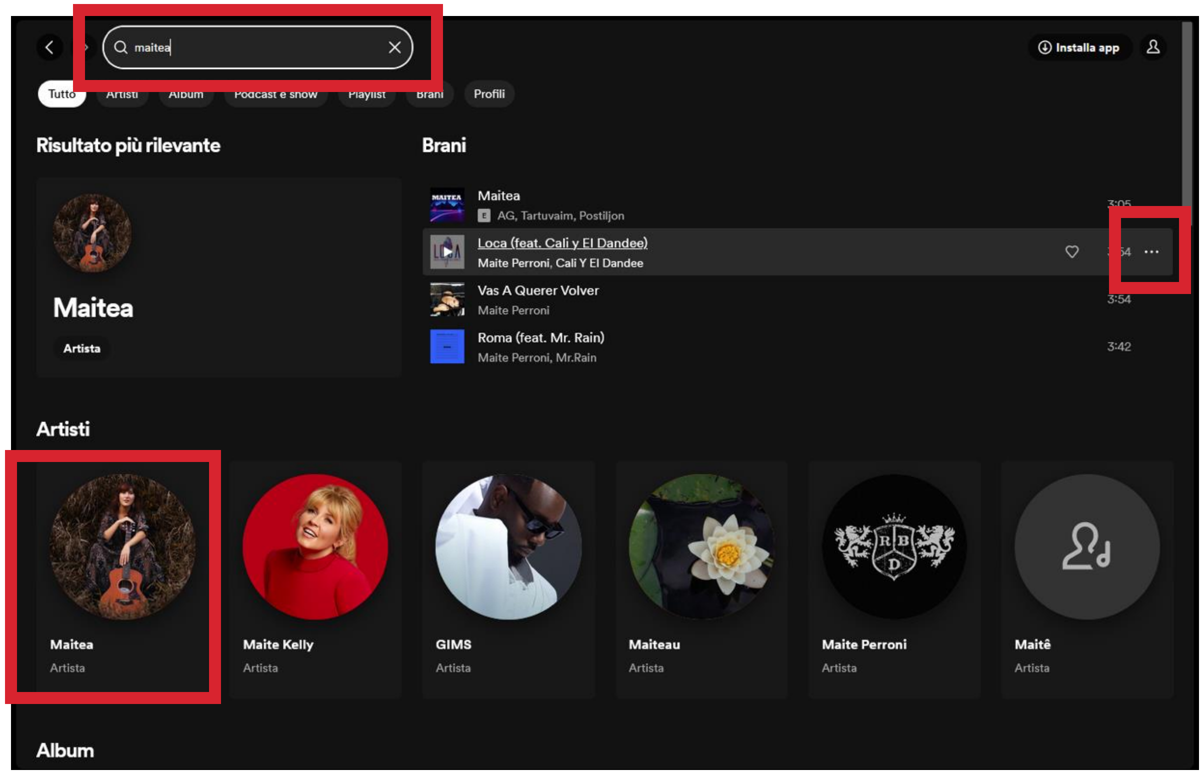
Retrieve the playlist

Check if the playlist contains the song

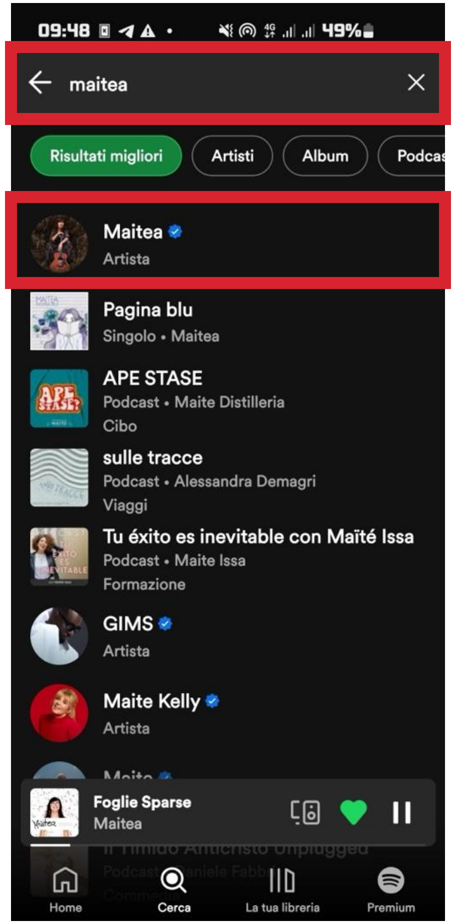
PASS

FAIL

Challenge 1: Operations Testing Order

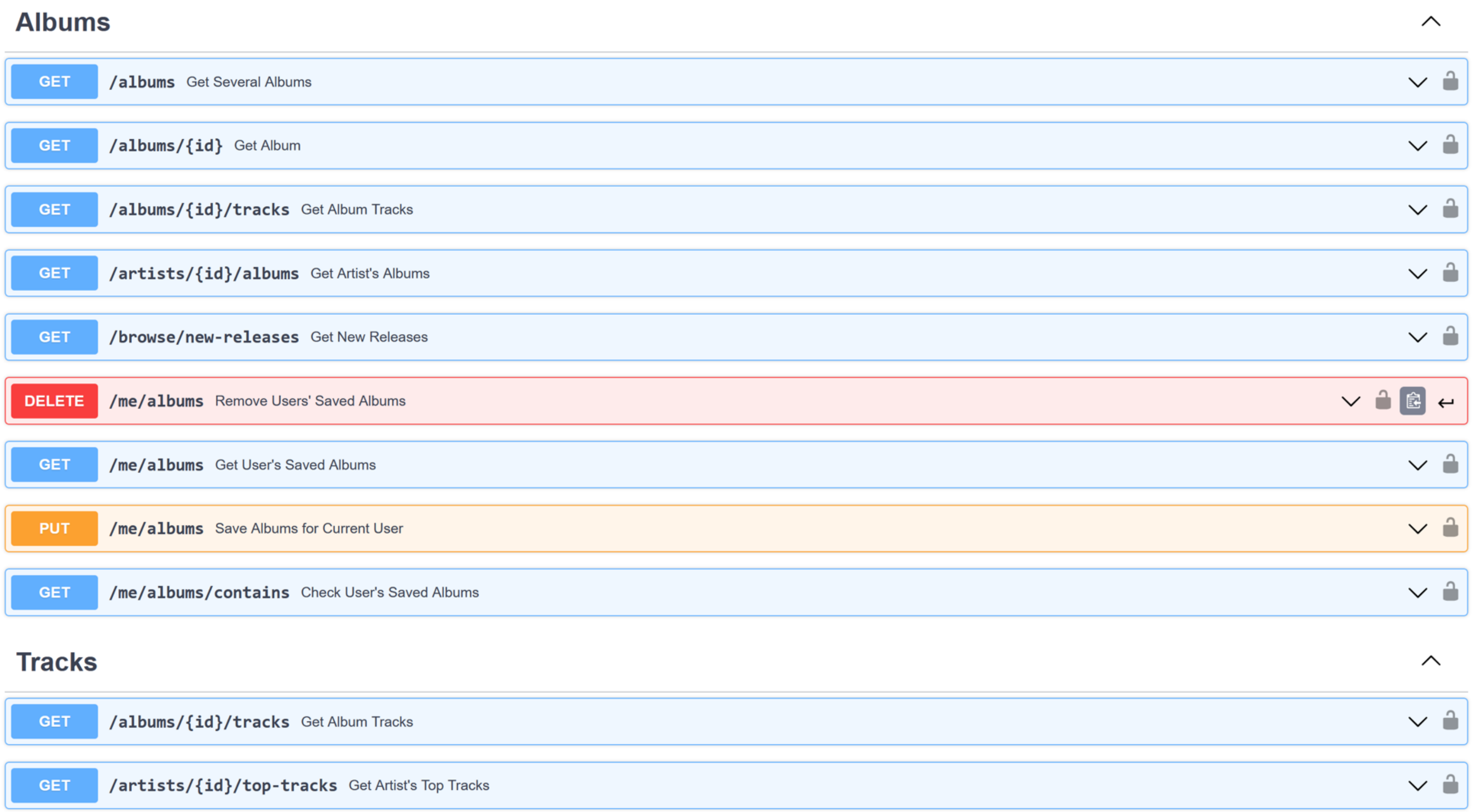


Spotify Web App



Spotify Android App

Challenge 1: Operations Testing Order



Albums

GET `/albums` Get Several Albums

GET `/albums/{id}` Get Album

GET `/albums/{id}/tracks` Get Album Tracks

GET `/artists/{id}/albums` Get Artist's Albums

GET `/browse/new-releases` Get New Releases

DELETE `/me/albums` Remove Users' Saved Albums

GET `/me/albums` Get User's Saved Albums

PUT `/me/albums` Save Albums for Current User

GET `/me/albums/contains` Check User's Saved Albums

Tracks

GET `/albums/{id}/tracks` Get Album Tracks

GET `/artists/{id}/top-tracks` Get Artist's Top Tracks

Spotify’s REST API specification

Challenge 2: Test Input Values

- What are suitable input values for input parameters?
    - API specifications often do not provide example values
    - Validity of values might depend on the state of the API
        - E.g., resource identifiers

Challenge 3: The Oracle Problem

- Did the SUT behave as expected during/after the test scenario?

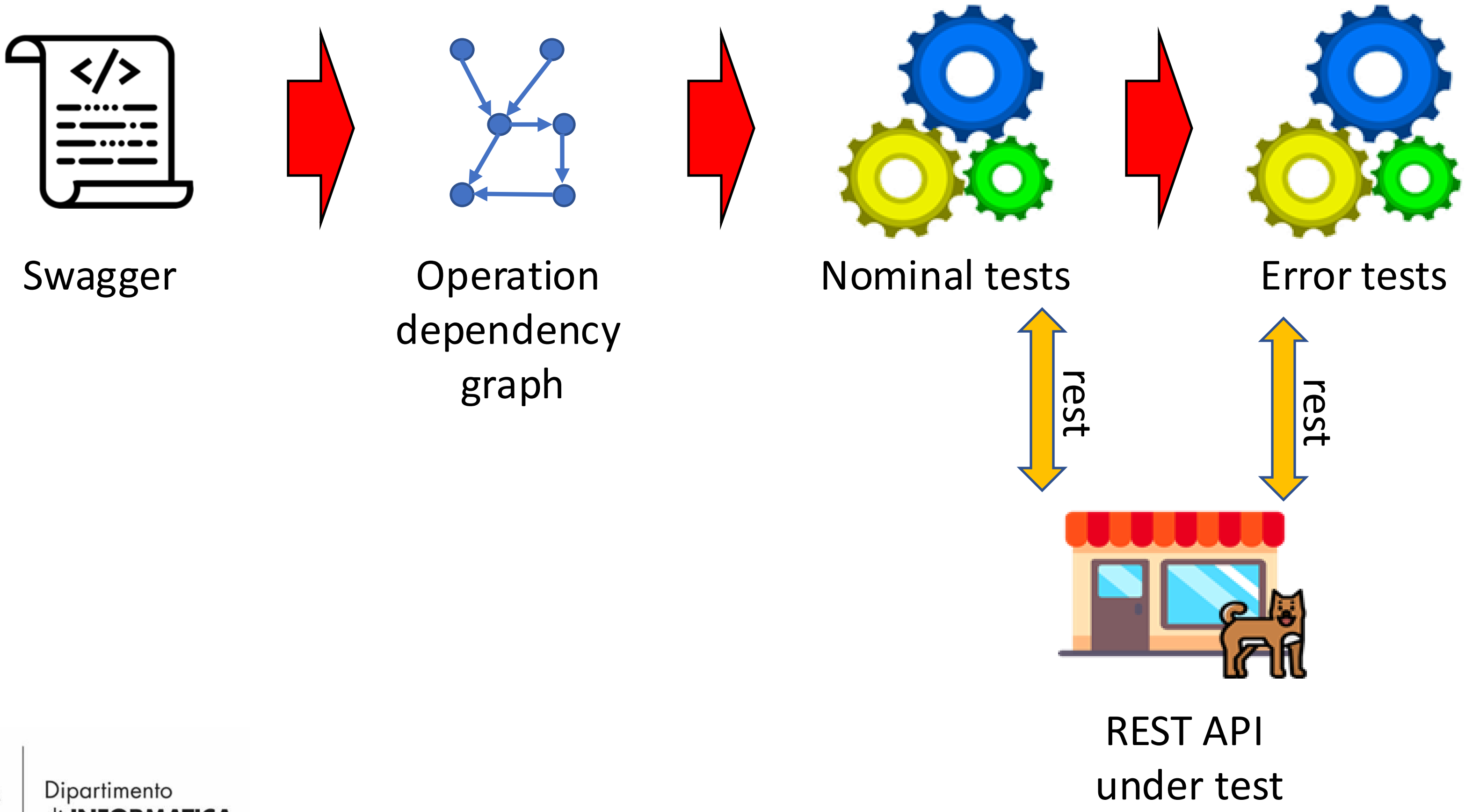
RestTestGen

Automated Black-Box Testing of Nominal and Error Scenarios in RESTful APIs

D. Corradini, A. Zampieri, M. Pasqua, E. Viglianisi, M. Dallago, M. Ceccato

Software Testing, Verification and Reliability (2022)

Approach overview



Swagger

Operation dependency graph

Nominal tests

Error tests

REST API under test

Operation Dependency

```
/pets:
  get:
    summary: List all pets
    operationId: getPets
    tags:
      - pets
    responses:
      '200':
        description: PetIds
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  petId:
                    type: integer
```

output

```
/pets/{petId}:
  get:
    summary: Info for a specific pet
    operationId: getPetById
    tags:
      - pets
    parameters:
      - name: petId
        in: path
        required: true
        schema:
          type: string
```

input

Case mismatch

petID, **petid**, **petId**

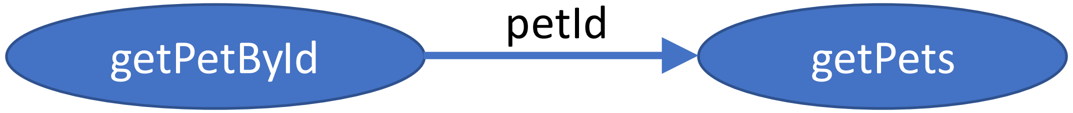
Id completion

/getPet ➠ **Pet**

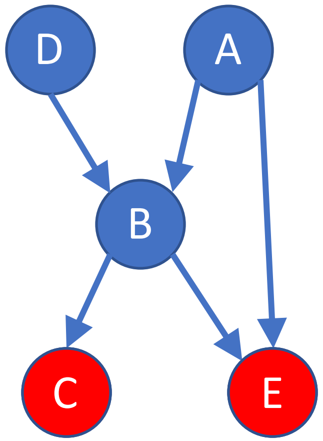
pet.id ➠ **petId**

Stemming

pets ➠ **pet**



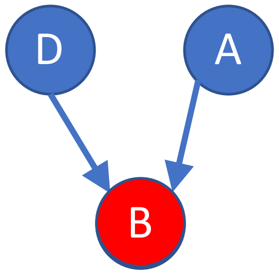
Operation Testing Order



- Leaf nodes are selected (no outgoing edges)
  - No input
  - Input is not available on operations output
- To maximize the likelihood of a successful test, resources might require to be in a certain status
- Leaf nodes are order based on the CRUD semantics

1. head
2. post
3. get
4. put/patch
5. delete

Operation Testing Order



- Tested operations are removed from the graph
- New operations become leaf nodes and can now be tested

The order in which operations are tested can not be precomputed, because it depends on what operations we succeed in testing

Input Value Generation

- Based on response dictionary
  - Map (name→values) of data observed at testing time, while testing previous operations

• Exact name match	petId ✓ petId
• Concatenation of object + field	pet.id ✓ petId
• Name edit distance < threshold	petsId ✓ petId
• Key is a substring	myPetId ✓ petId

- Based on swagger definition
  - Enum, example, default values
  - Random values (compatible with constraints)

HTTP Status Code Oracle

- 2xx means correct execution ✓
  - 200: ok
  - 201: successful resource creation

- 4xx means error that is correctly handled ?
  - 400: bad request
  - 404: not found

- 5xx means error ✗
  - 500: server crash

Schema Validation Oracle

```
responses:
  '200':
    description: Expected response to a valid request
    content:
      application/json:
        schema:
          $ref: "#/components/schemas/Pet"
```

```
components:
  schemas:
    Pet:
      type: object
      required:
        - id
        - name
        - tag
      properties:
        id:
          type: integer
          format: int64
        name:
          type: string
        tag:
          type: string
```

✓

```
{
  "id": 1,
  "name": "doggy",
  "tag": "dog"
}
```

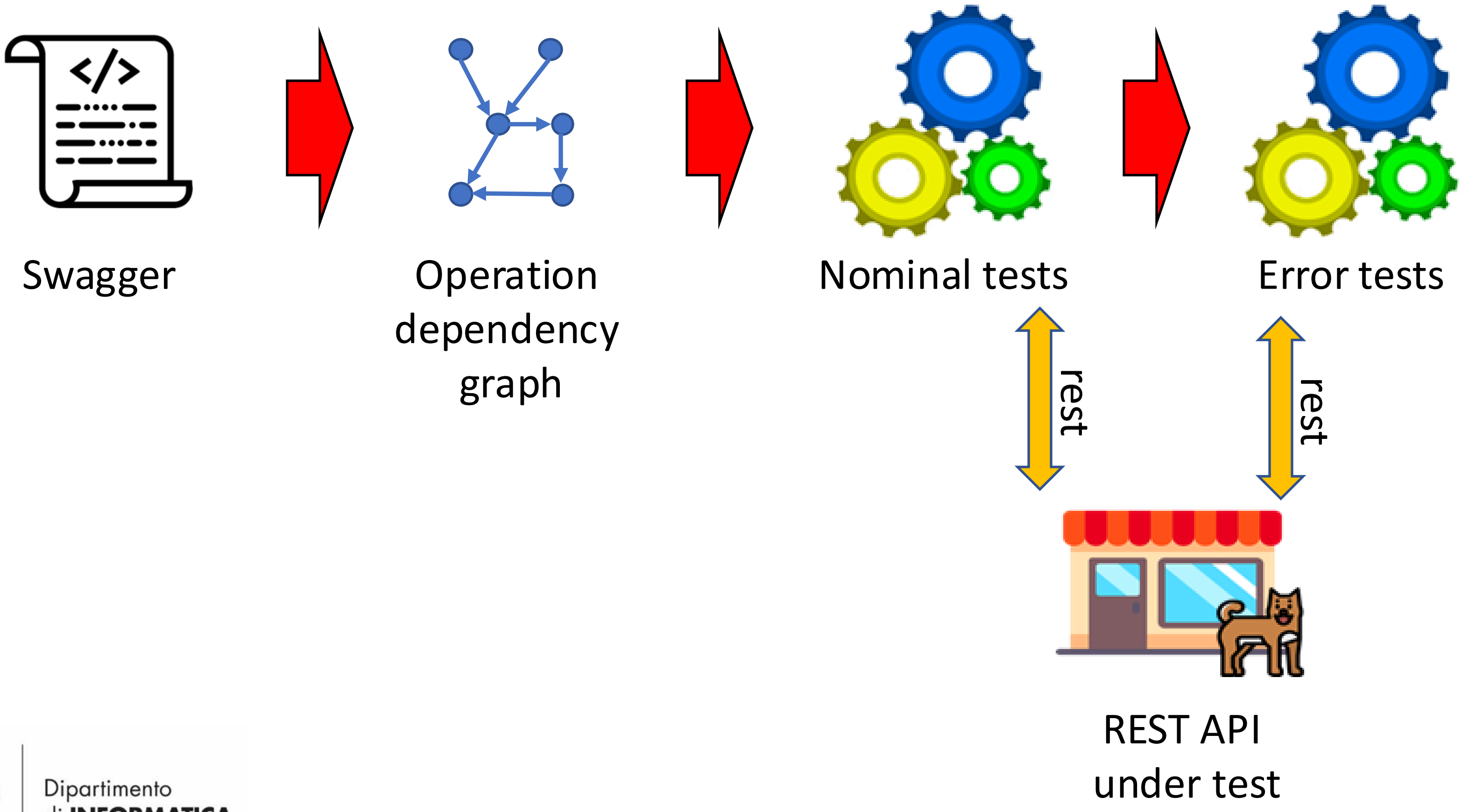
✗

```
{
  "id": 1,
  "name": "doggy"
}
```

✗

```
{
  "id": 1,
  "name": "doggy",
  "tag": 5
}
```

Approach overview



Swagger

Operation dependency graph

Nominal tests

Error tests

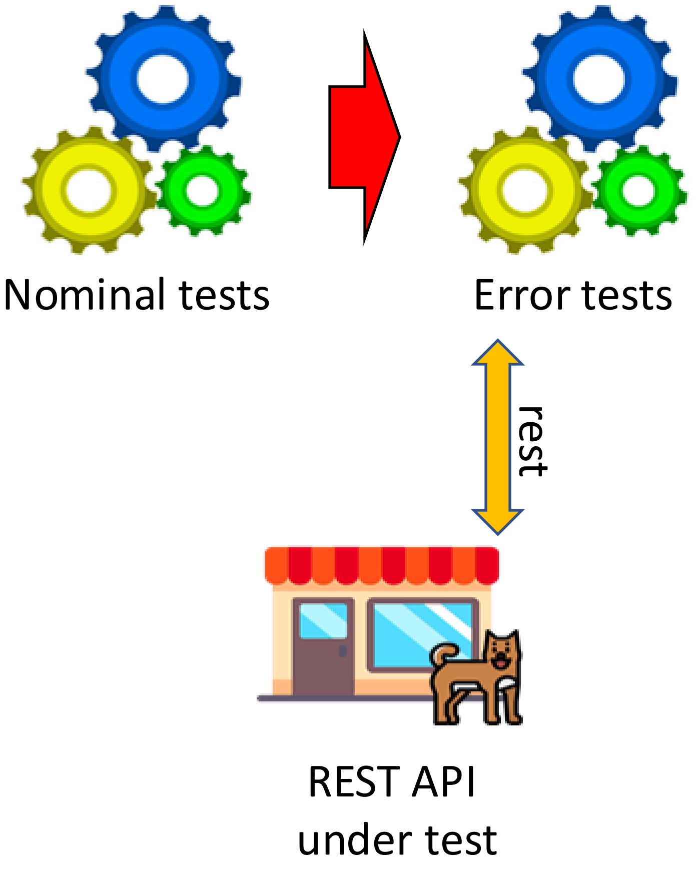
rest

rest

REST API under test

Testing of Error Cases

- Analyses how an API behaves when it is given wrong input data
- Mutation operators
  - Remove a **`required`** input field
  - Change field type
  - Change field value



HTTP Status Code Oracle

- 2xx means correct execution ✗
  - 200: ok
  - 201: successful resource creation

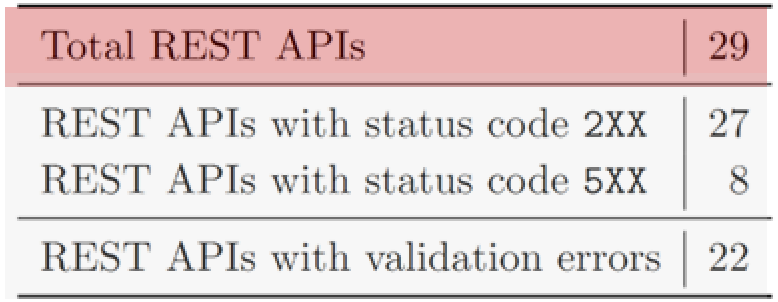
- 4xx means error that is correctly handled ?
  - 400: bad request
  - 404: not found

- 5xx means error ✗
  - 500: server crash

Empirical Validation

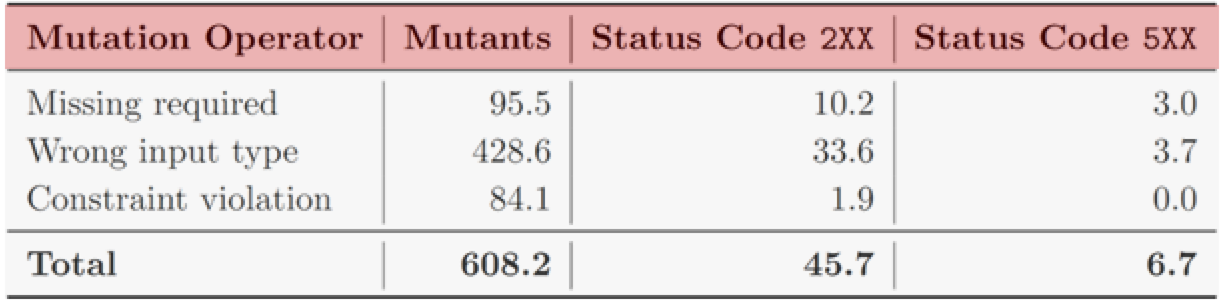
- Nominal Tester
- Error Tester
- Operation Dependency Graph

Results



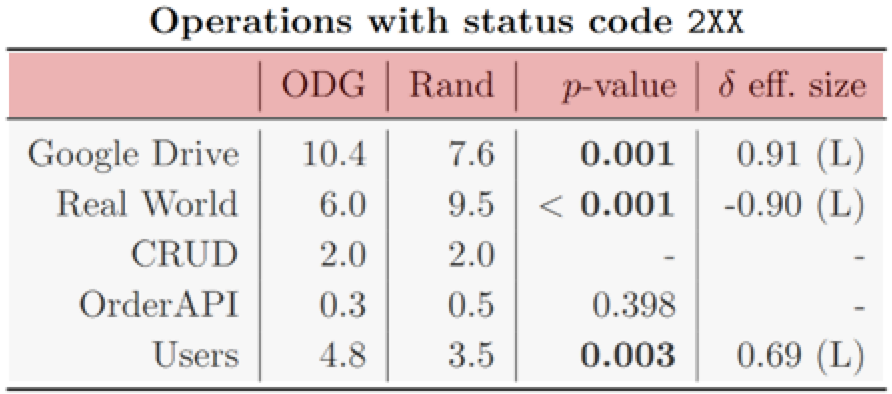
Total REST APIs	29
REST APIs with status code 2XX	27
REST APIs with status code 5XX	8
REST APIs with validation errors	22

Nominal Tester



Mutation Operator	Mutants	Status Code 2XX	Status Code 5XX
Missing required	95.5	10.2	3.0
Wrong input type	428.6	33.6	3.7
Constraint violation	84.1	1.9	0.0
Total	**608.2**	**45.7**	**6.7**

Error Tester



Operations with status code 2XX

	ODG	Rand	p-value	δ eff. size
Google Drive	10.4	7.6	**0.001**	0.91 (L)
Real World	6.0	9.5	$<$ **0.001**	-0.90 (L)
CRUD	2.0	2.0	-	-
OrderAPI	0.3	0.5	0.398	-
Users	4.8	3.5	**0.003**	0.69 (L)

Contribution of ODG

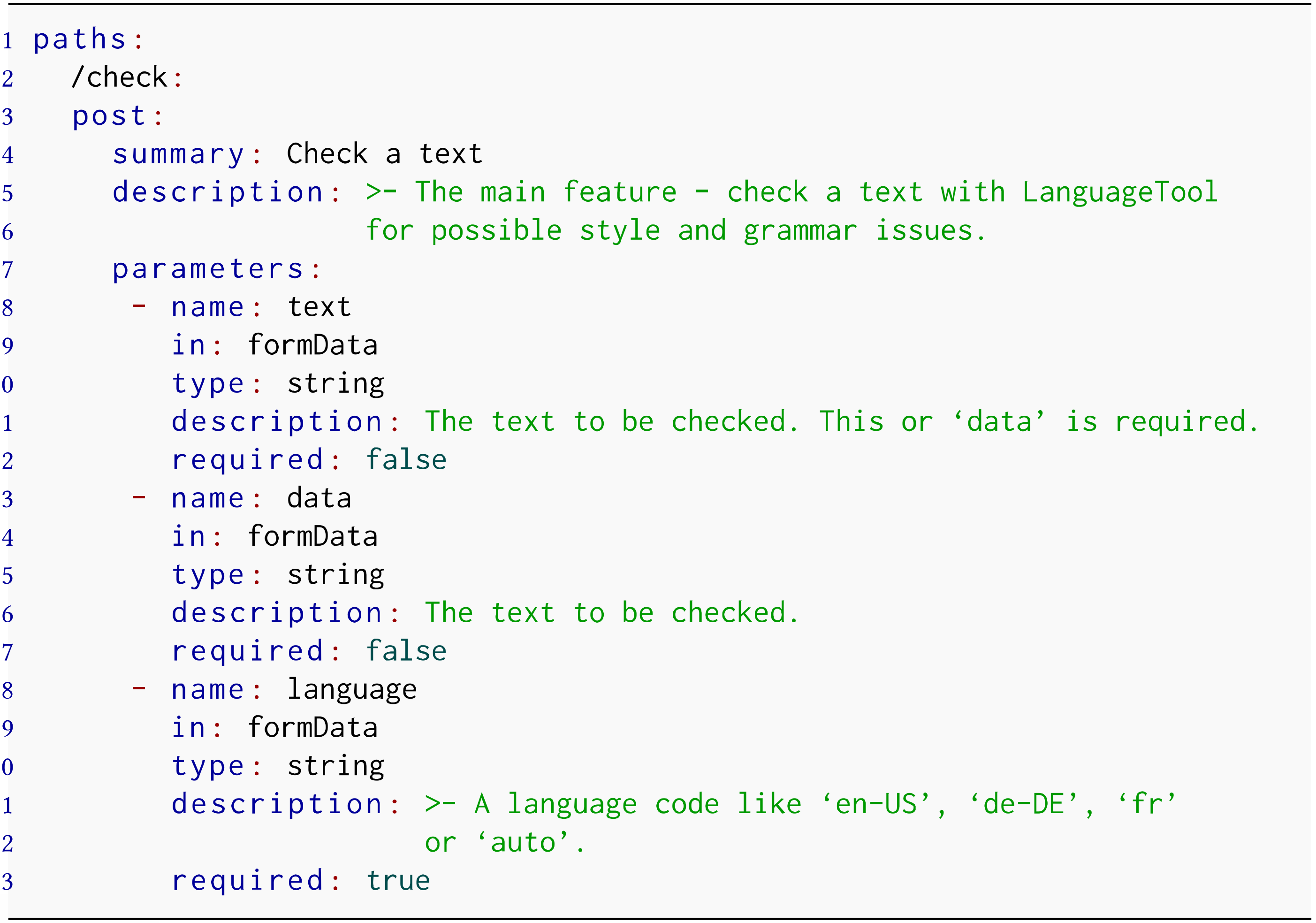
Considerations

- Urgent need for automated test case generation
- Incomplete specifications
- Data dependencies contribute to effectiveness
  - implicit dependencies are ignored
- Security requirements

Enhancing REST API Testing with NLP Techniques

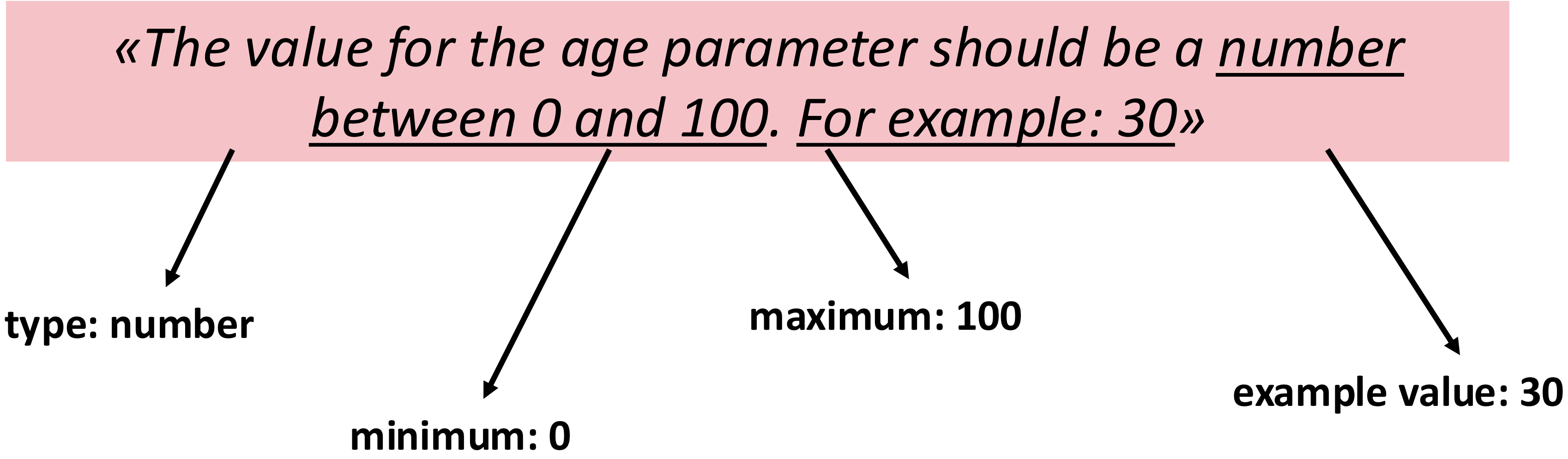
Enhancing REST API Testing with NLP Techniques
M. Kim, D. Corradini, S. Sinha, A. Orso, M. Pasqua, R. Tzoref-Brill, M. Ceccato
32nd International Symposium on Software Testing and Analysis (ISSTA 2023)

NLP-Enhanced REST API testing



```
paths:
  /check:
  post:
    summary: Check a text
    description: >- The main feature - check a text with LanguageTool
                   for possible style and grammar issues.
    parameters:
     - name: text
       in: formData
       type: string
       description: The text to be checked. This or ‘data’ is required.
       required: false
     - name: data
       in: formData
       type: string
       description: The text to be checked.
       required: false
     - name: language
       in: formData
       type: string
       description: >- A language code like ‘en-US’, ‘de-DE’, ‘fr’
                    or ‘auto’.
       required: true
```

NLP-Enhanced REST API testing



«The value for the age parameter should be a number between 0 and 100. For example: 30»

type: number

minimum: 0

maximum: 100

example value: 30

NLP-Enhanced REST API testing

the maximum value is 50

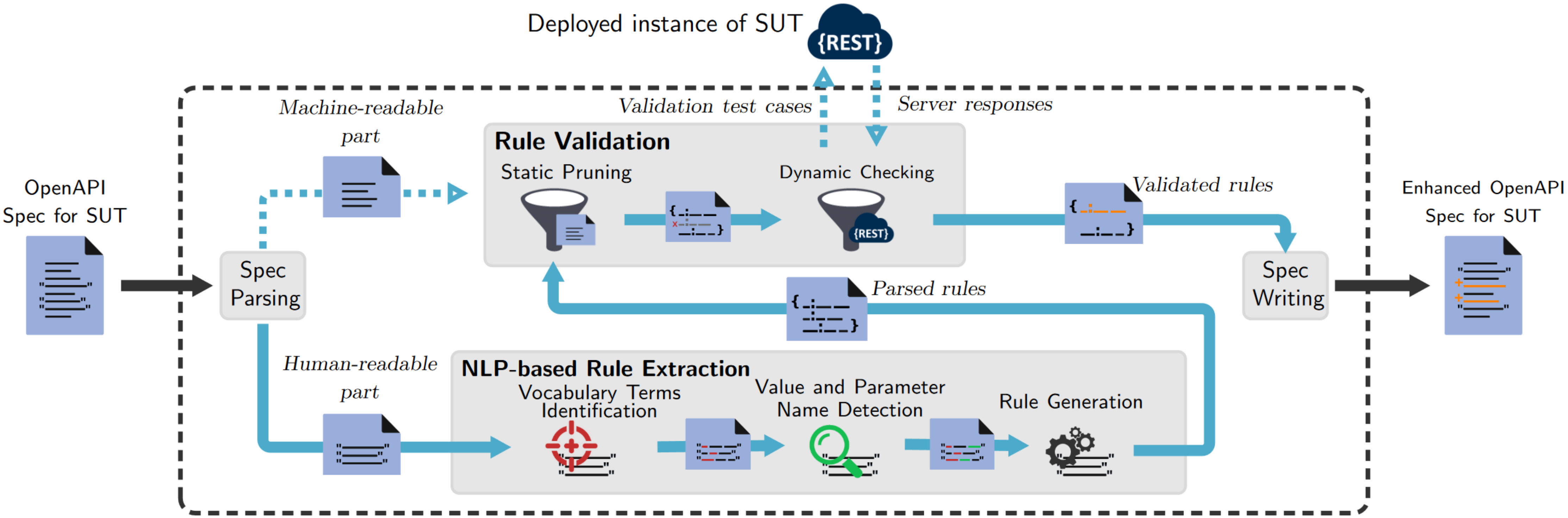
the value is up to 50

the value can’t be larger than 50

you must also set X

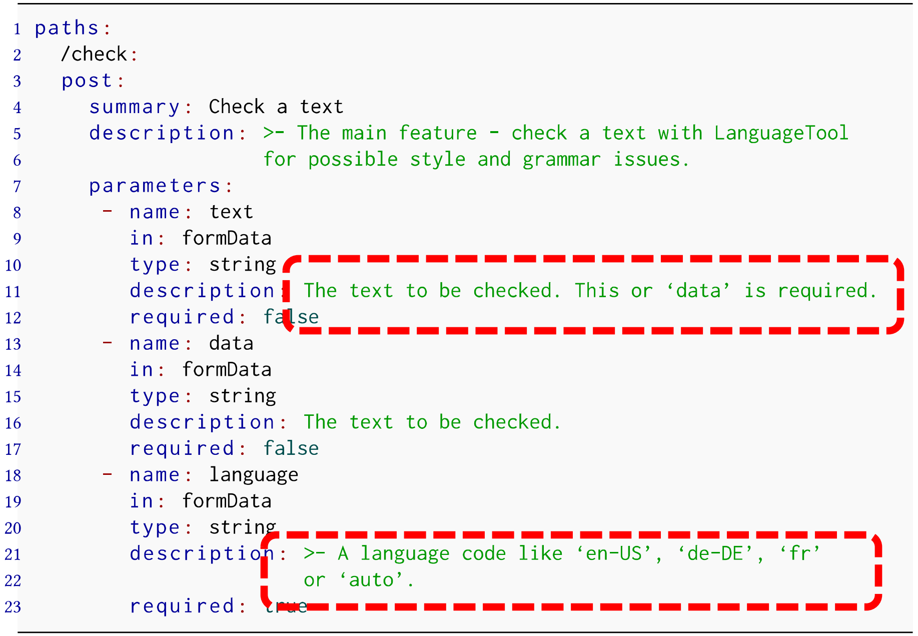
X must also be specified

NLP2REST


Deployed instance of SUT
{REST}
Machine-readable part
Validation test cases
Server responses
Rule Validation
Static Pruning
Dynamic Checking
{REST}
Validated rules
OpenAPI Spec for SUT
Spec Parsing
Parsed rules
Spec Writing
Enhanced OpenAPI Spec for SUT
Human-readable part
NLP-based Rule Extraction
Vocabulary Terms Identification
Value and Parameter Name Detection
Rule Generation

Rules from human readable constraints

1. Potential rules
   - Word2Vec model pre-trained on OpenAPI terminology
   - **`Cos.sim(sentence,term) > 0.7`**

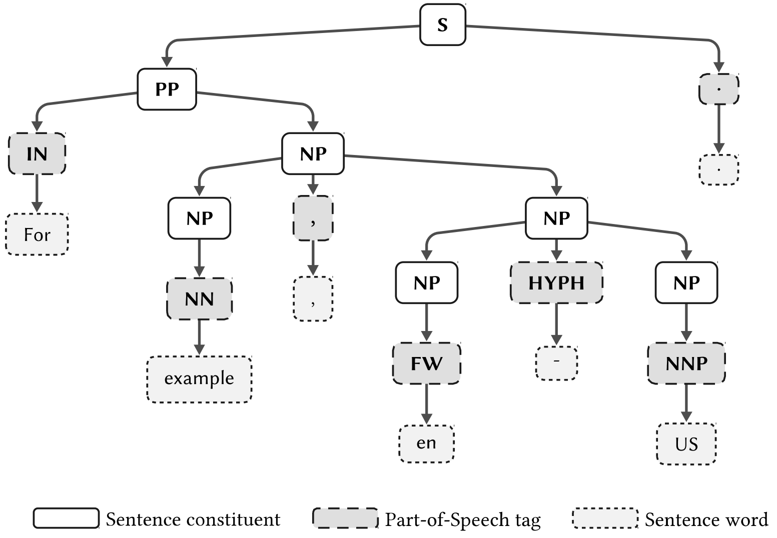


```yaml
paths:
  /check:
  post:
    summary: Check a text
    description: >- The main feature - check a text with LanguageTool
                 for possible style and grammar issues.
    parameters:
     - name: text
       in: formData
       type: string
       description: The text to be checked. This or ‘data’ is required.
       required: false
     - name: data
       in: formData
       type: string
       description: The text to be checked.
       required: false
     - name: language
       in: formData
       type: string
       description: >- A language code like ‘en-US’, ‘de-DE’, ‘fr’
                    or ‘auto’.
       required: true
```

Rules from human readable constraints

2. Value and Parameter Name Detection
   - Constituents grammar
     - parameter names
     - values (regular expressions)
     - terms relevant to inter-parameter dependencies (word2vec)

For example, en-US

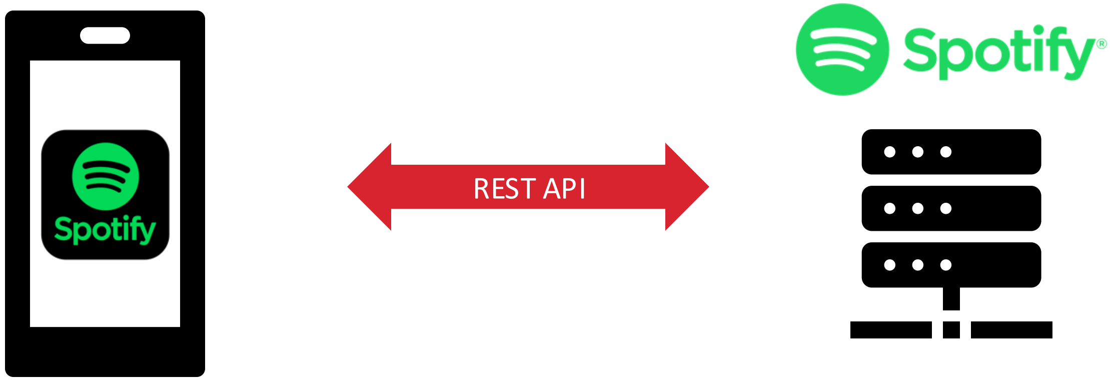


examples: {1:en-US}

Static rule validation

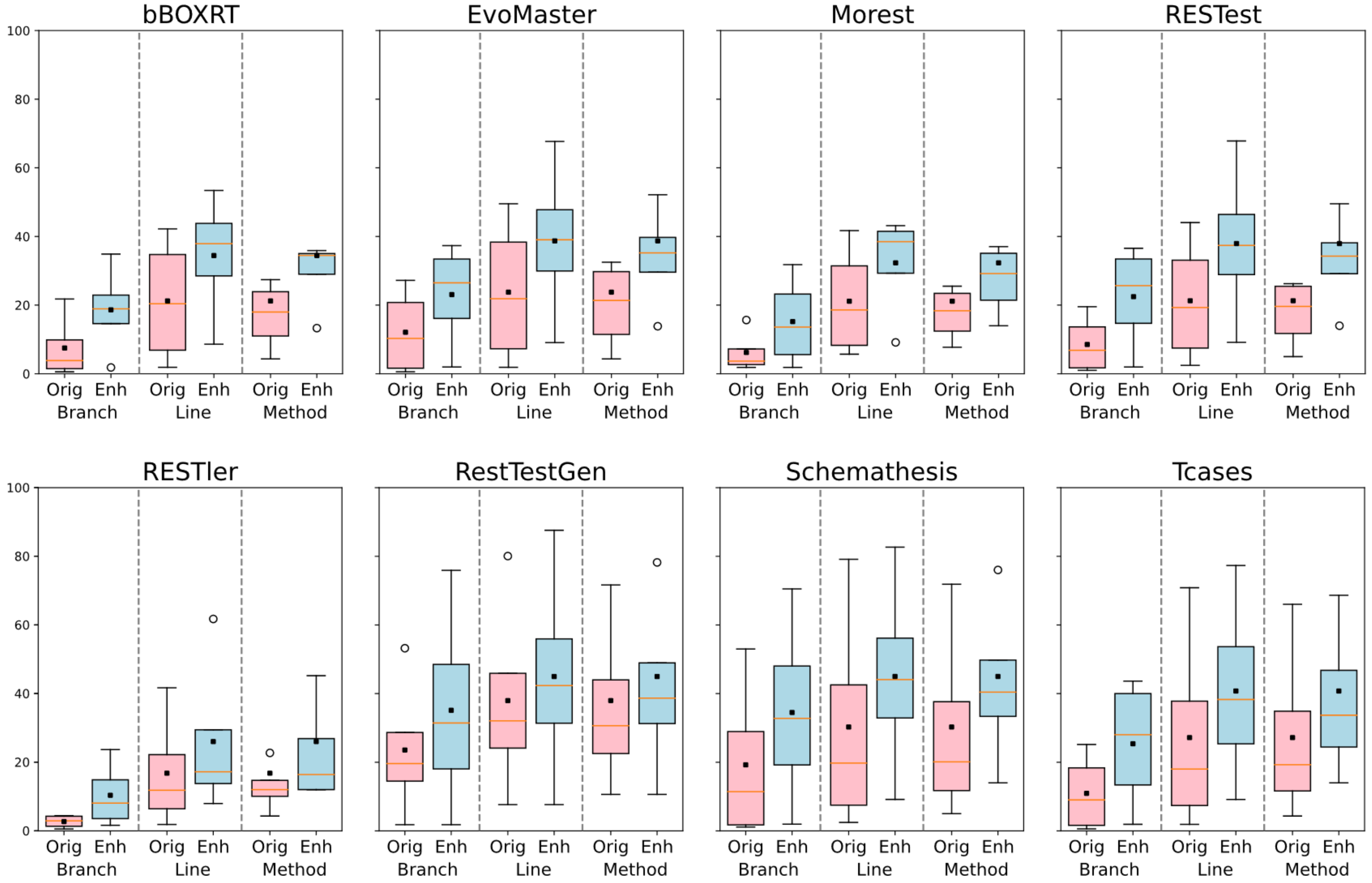
1. Discard incompatible rules:
   - Mandatory parameters not in **`Or`**, **`OnlyOne`**, **`ZeroOrOne`**, **`AllOrNone`** rules
   - **`maximum`** only for numeric, and compatible with **`minumum`**
   - **`Example`**, **`enum`**, and **`default`** values must match the type
   - No multiple **`default`** values

Dynamic Rule validation



NLP2REST: Results

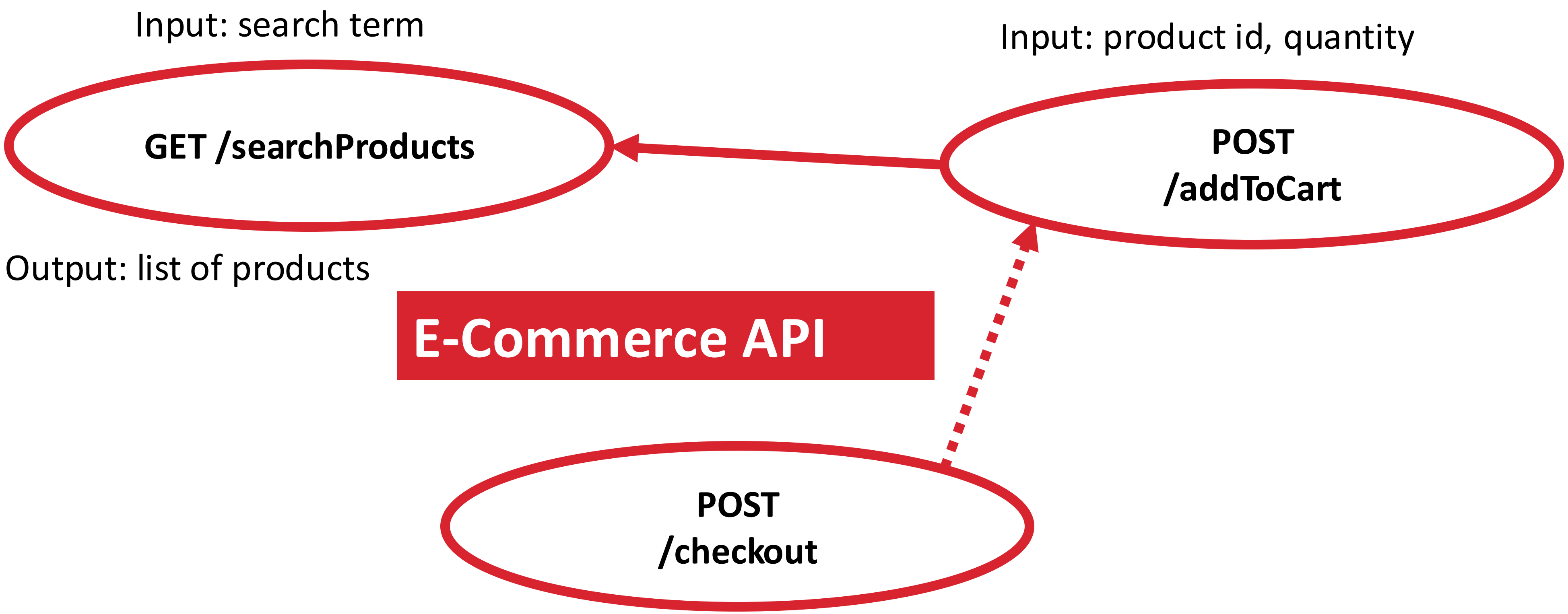
- Code coverage with enhanced spec.
  - +20% succ. reqs.
  - +101% branch cov.
  - +52% method cov.
  - +50% line cov.



Deep Reinforcement Learning-Based REST API Testing

DeepREST: Automated Test Case Generation for REST APIs Exploiting Deep Reinforcement Learning
D. Corradini, Z. Montolli, M. Pasqua, M. Ceccato
39th International Conference on Automated Software Engineering (ASE 2024)

E-Commerce API



Input: search term

GET /searchProducts

Output: list of products

Input: product id, quantity

POST /addToCart

E-Commerce API

POST /checkout

Problem: implicit dependencies are ignored!

SotA: Input Generation

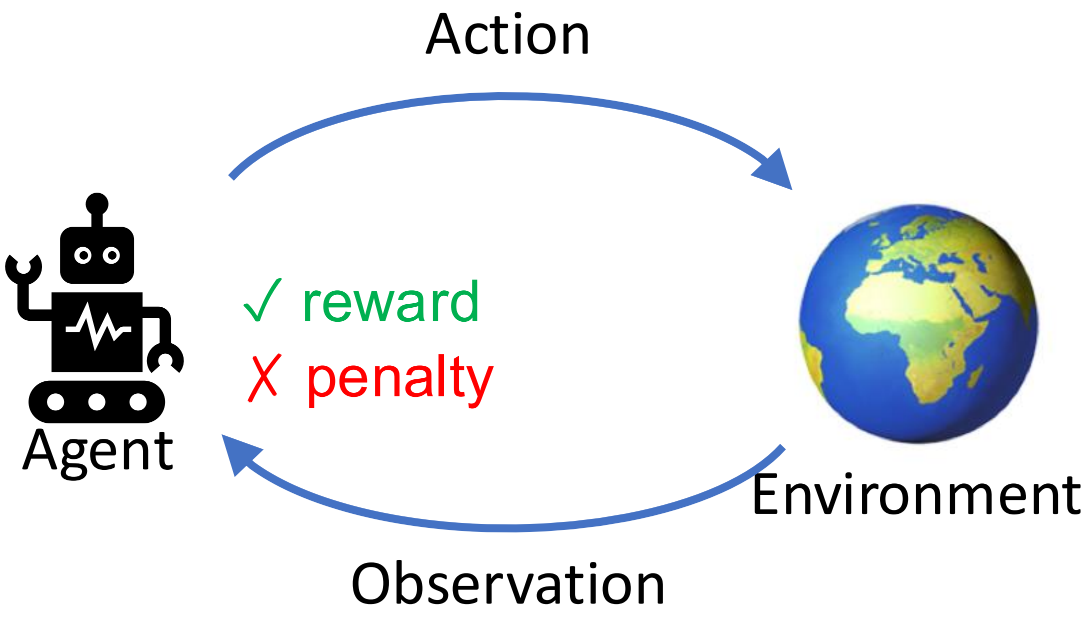
- Random values compliant with data formats
- Values from the OpenAPI specification
  - Default values
  - Enum values
  - Example values
- Values observed in previous HTTP interactions

- These strategies are picked randomly
- Non-mandatory parameters are used with probability P (typically < 0.2)

Problem: many failures due to wrong input!

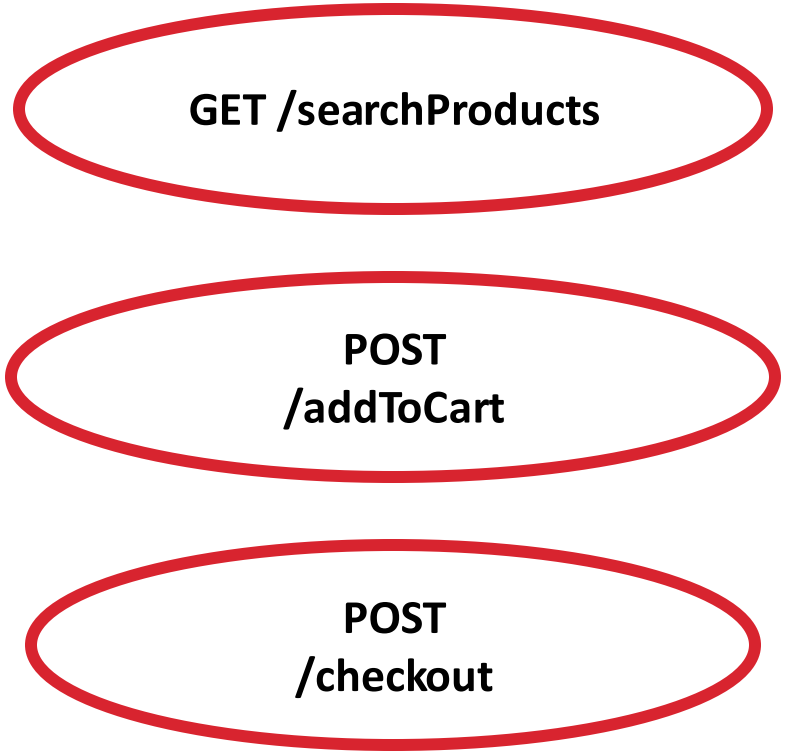
Reinforcement Learning

- **Action Space**: what we can act on
- **State Space**: what we measure of the environment
- **Reward Function**: the feedback signal



Action

- GET /searchProducts
- POST /addToCart
- POST /checkout



GET /searchProducts

POST /addToCart

POST /checkout

State

[0,0,0]

Search

Add to cart

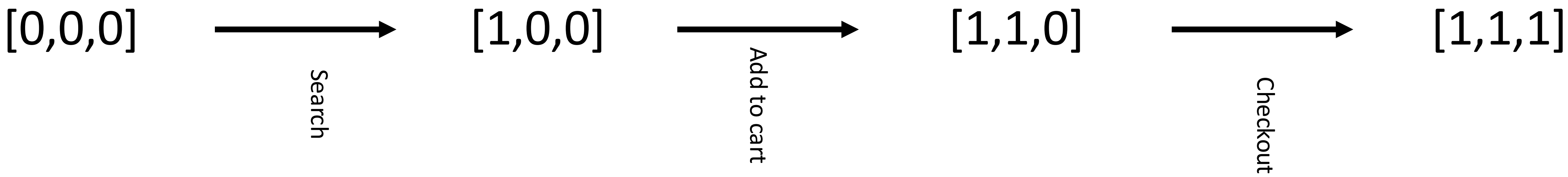
Checkout

GET /searchProducts

POST /addToCart

POST /checkout

State transition



Reward: curiosity driven

Agent

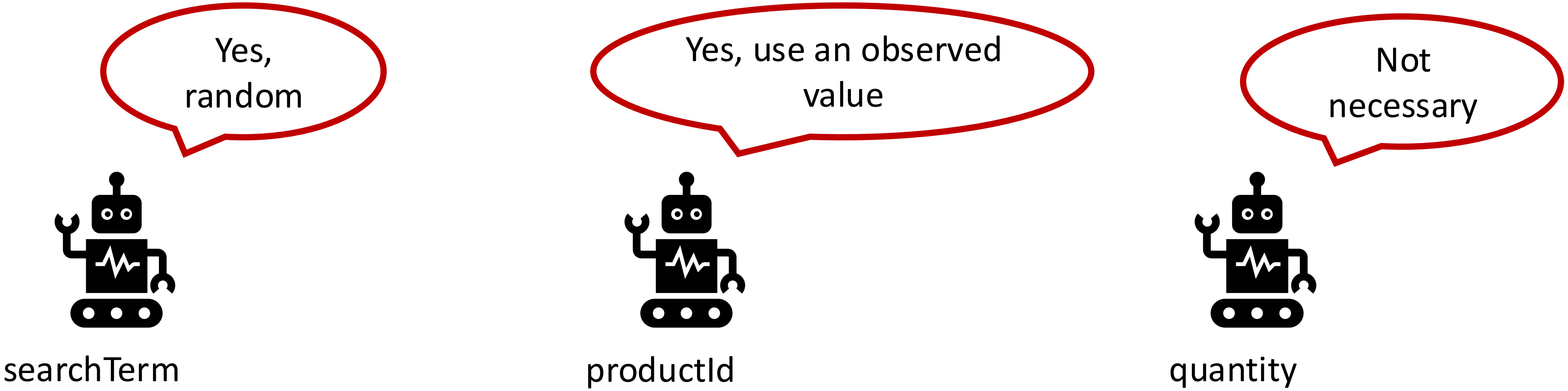
✓ reward

✗ penalty

- Positive: Successfully tested a new operation (never visited so far)
- Negative: Successfully tested an operation that was already tested
- Slightly negative: Fail in testing an operation

Input Generation: Experience Driven

- Random, example values, response dictionary, etc.



Yes, random

searchTerm

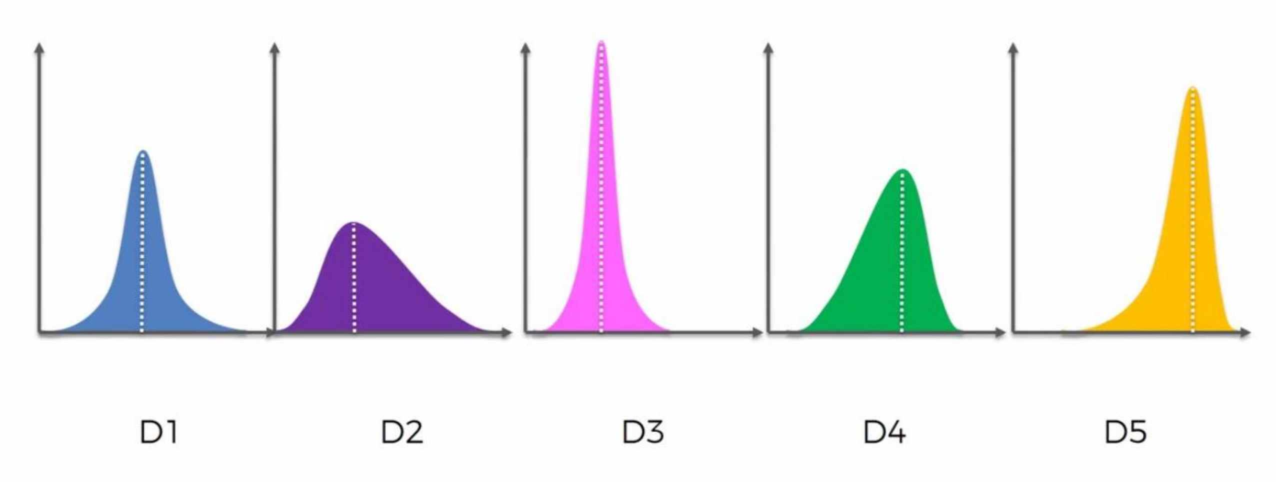
Yes, use an observed value

productId

Not necessary

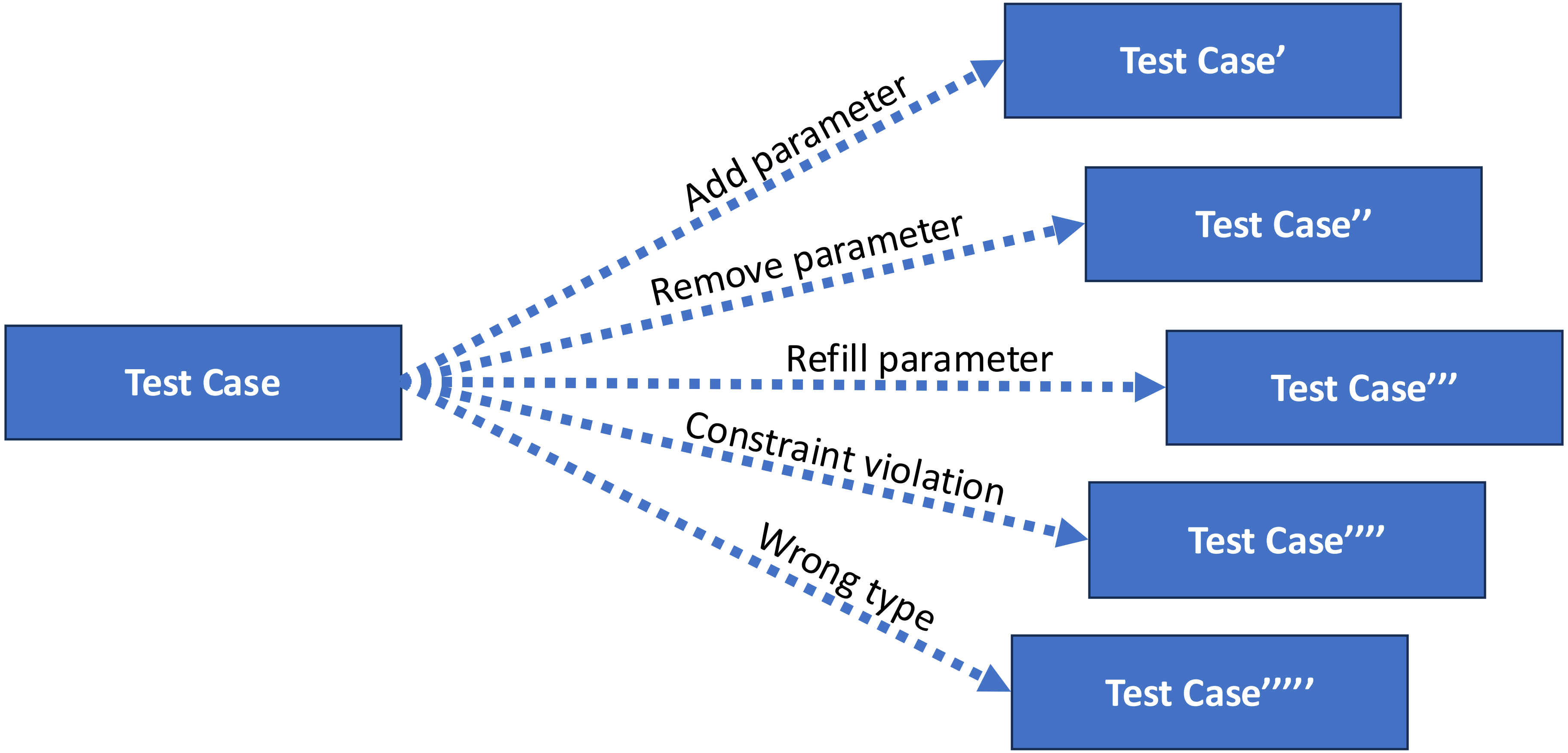
quantity

The Multi-Armed Bandit Problem

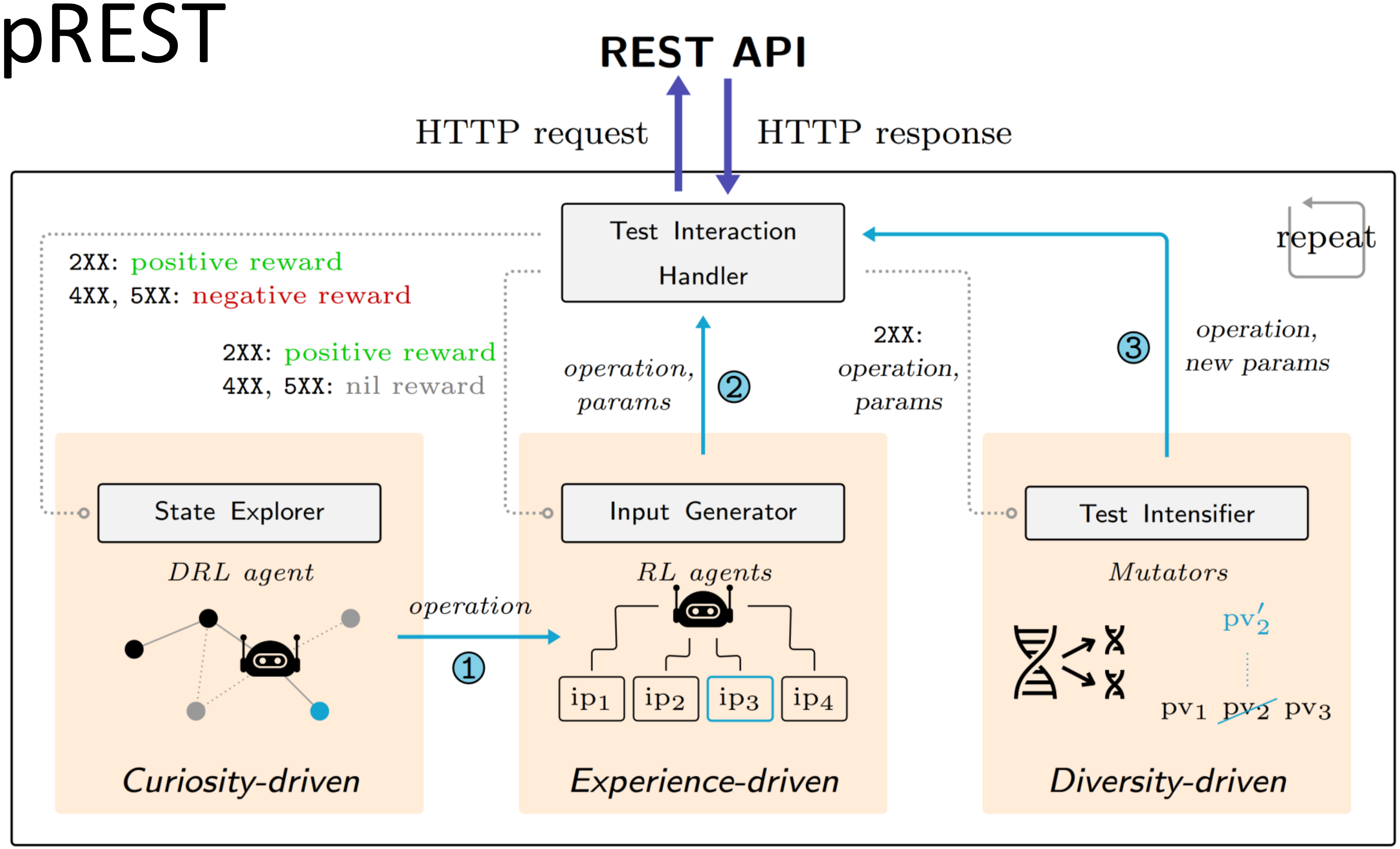


- Epsilon-Greedy algorithms

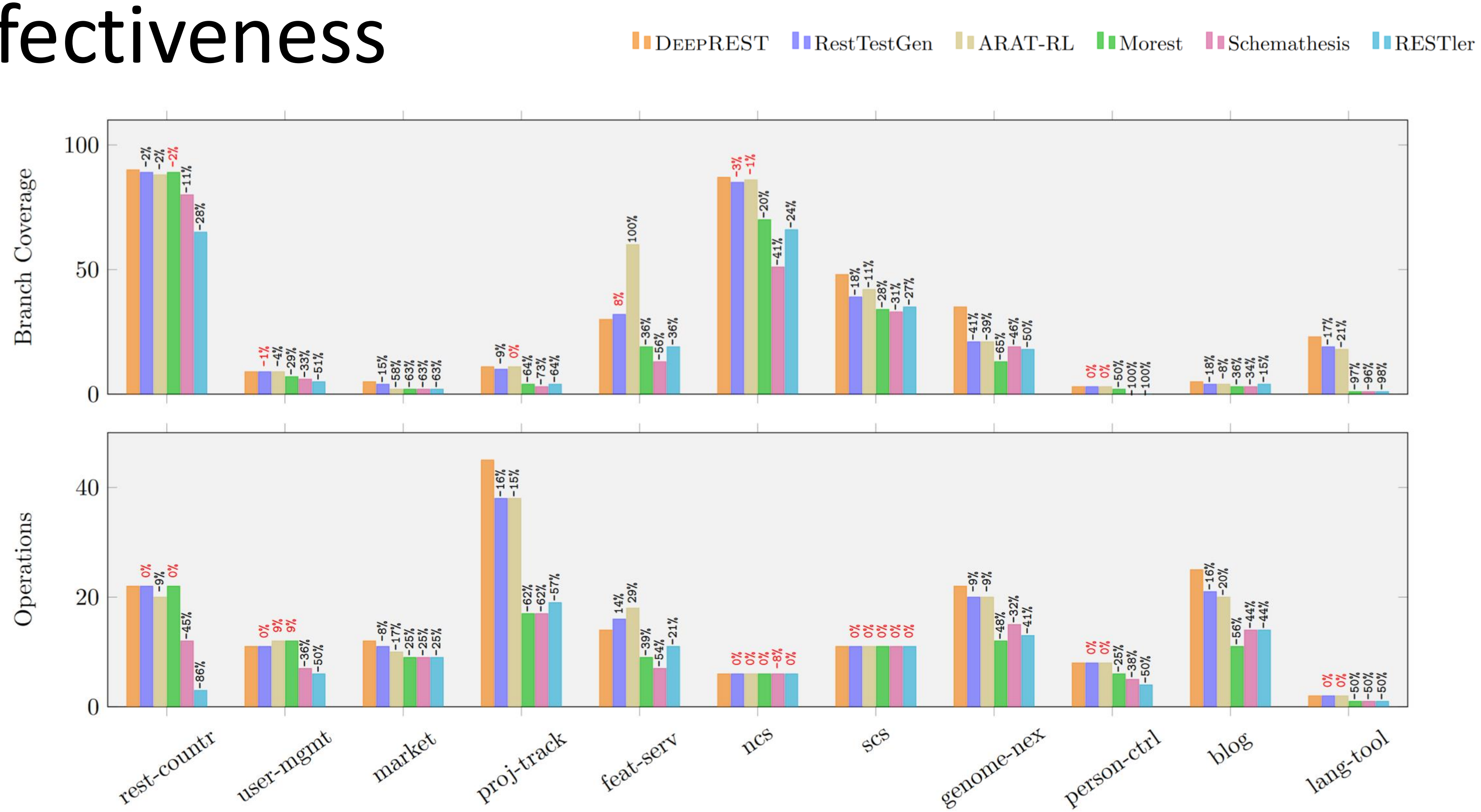
Test Intensification



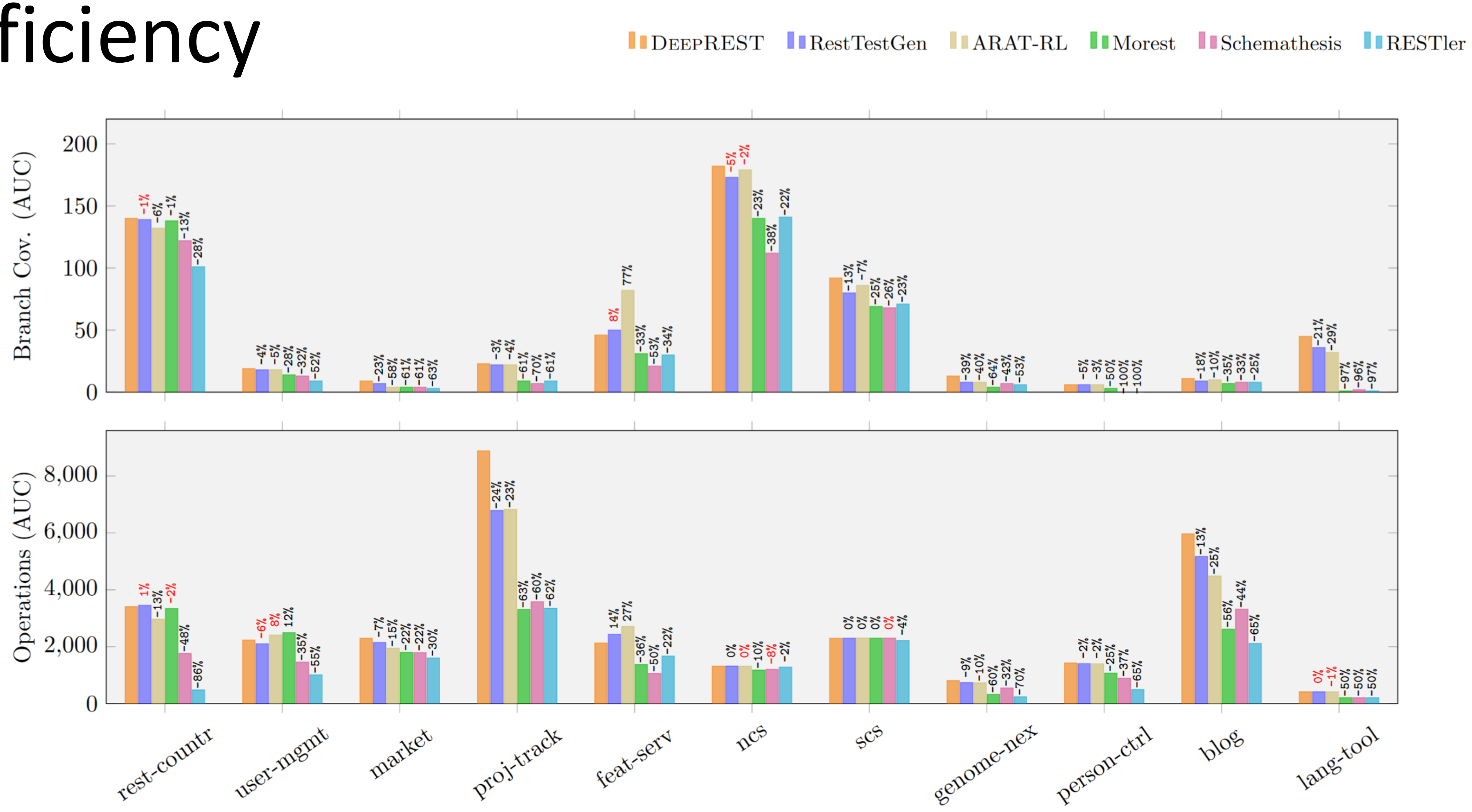
DeepREST


REST API
HTTP request
HTTP response
Test Interaction Handler
repeat
2XX: positive reward
4XX, 5XX: negative reward
2XX: positive reward
4XX, 5XX: nil reward
operation, params
2
2XX: operation, params
3
operation, new params
State Explorer
DRL agent
Curiosity-driven
operation
1
Input Generator
RL agents
ip1
ip2
ip3
ip4
Experience-driven
Test Intensifier
Mutators
pv'2
pv1
~~pv2~~
pv3
Diversity-driven

Effectiveness



Efficiency


DeepREST
RestTestGen
ARAT-RL
Morest
Schemathesis
RESTler
Branch Cov. (AUC)
0
50
100
150
200
-1%
-6%
-1%
-13%
-28%
-4%
-5%
-28%
-32%
-52%
-23%
-58%
-61%
-61%
-63%
-3%
-4%
-61%
-70%
-61%
8%
77%
-33%
-53%
-34%
-5%
-2%
-23%
-38%
-22%
-13%
-7%
-25%
-26%
-23%
-39%
-40%
-64%
-43%
-53%
-5%
-3%
-50%
-100%
-100%
-18%
-10%
-35%
-33%
-25%
-21%
-29%
-97%
-96%
-97%
Operations (AUC)
0
2,000
4,000
6,000
8,000
1%
-13%
-2%
-48%
-86%
-6%
8%
12%
-35%
-55%
-7%
-15%
-22%
-22%
-30%
-24%
-23%
-63%
-60%
-62%
14%
27%
-36%
-50%
-22%
0%
0%
-10%
-8%
-2%
0%
0%
0%
0%
-4%
-9%
-10%
-60%
-32%
-70%
-2%
-2%
-25%
-37%
-65%
-13%
-25%
-56%
-44%
-65%
0%
-1%
-50%
-50%
-50%
rest-countr
user-mgmt
market
proj-track
feat-serv
ncs
scs
genome-nex
person-ctrl
blog
lang-tool

Security Testing of Mass Assignment Vulnerabilities

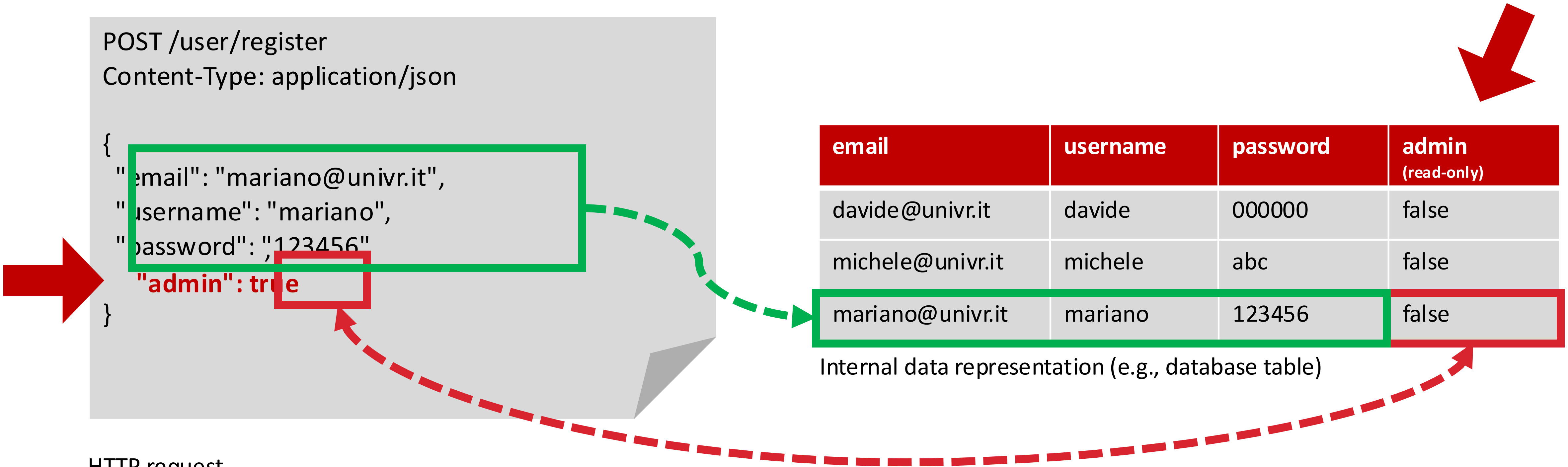
Automated Black-Box Testing of Mass Assignment Vulnerabilities in RESTful APIs

D. Corradini, M. Pasqua, M. Ceccato

45th International Conference on Software Engineering (ICSE 2023)

UNIVERSITÀ di VERONA | Dipartimento di INFORMATICA

Auto-binding



```
POST /user/register
Content-Type: application/json

{
  "email": "mariano@univr.it",
  "username": "mariano",
  "password": "123456",
    "admin": true
}
```

HTTP request

email	username	password	admin (read-only)
davide@univr.it	davide	000000	false
michele@univr.it	michele	abc	false
mariano@univr.it	mariano	123456	false

Internal data representation (e.g., database table)

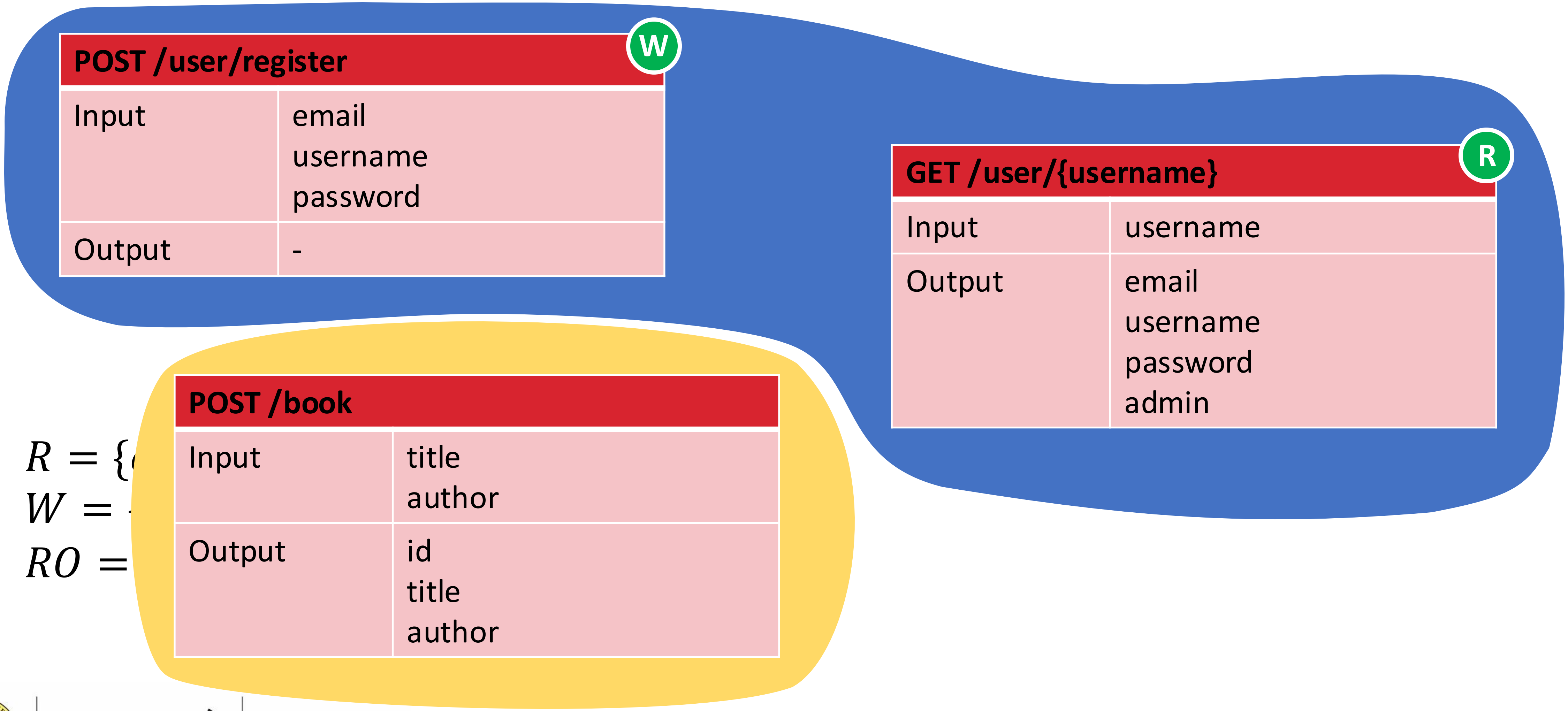
Approach

1. Identification of read-only fields

2. Generation of security test cases

3. Security testing oracle

1. Identification of read-only fields



POST /user/register	W
Input	email username password
Output	-

GET /user/{username}	R
Input	username
Output	email username password admin

POST /book	
Input	title author
Output	id title author

$R = \{$

$W =$

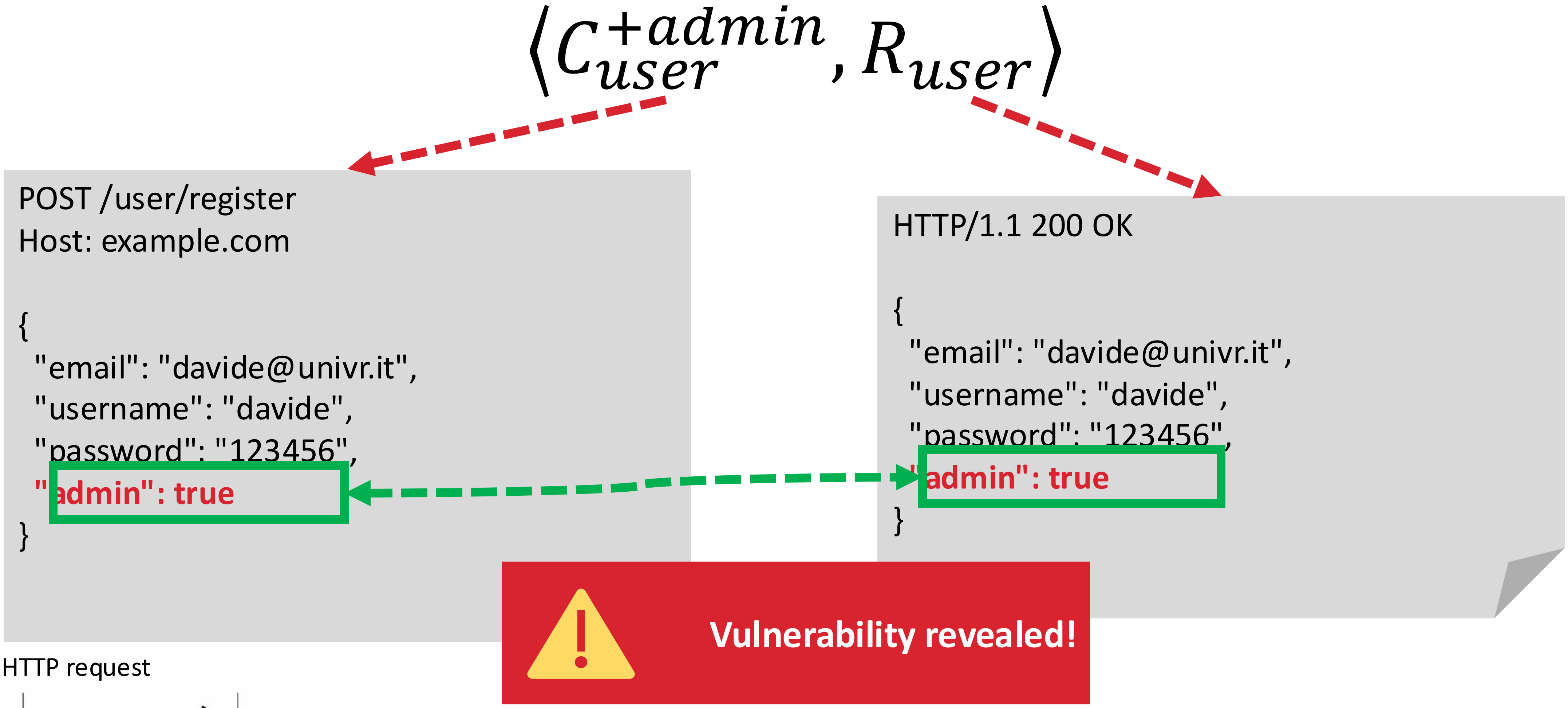
$RO =$

2. Test case generation

Abstract test templates

$$\begin{tikzcd} \langle C_{\tau}^{+f}, \arrow[d, dashed] & R_{\tau} \rangle \arrow[d, dashed] \\ \langle C_{user}^{+admin}, & R_{user} \rangle \end{tikzcd}$$

3. Security oracle



$$\langle C_{user}^{+admin}, R_{user} \rangle$$

```
POST /user/register
Host: example.com

{
  "email": "davide@univr.it",
  "username": "davide",
  "password": "123456",
  "admin": true
}
```

HTTP request

```
HTTP/1.1 200 OK

{
  "email": "davide@univr.it",
  "username": "davide",
  "password": "123456",
  "admin": true
}
```

Vulnerability revealed!

Evaluation

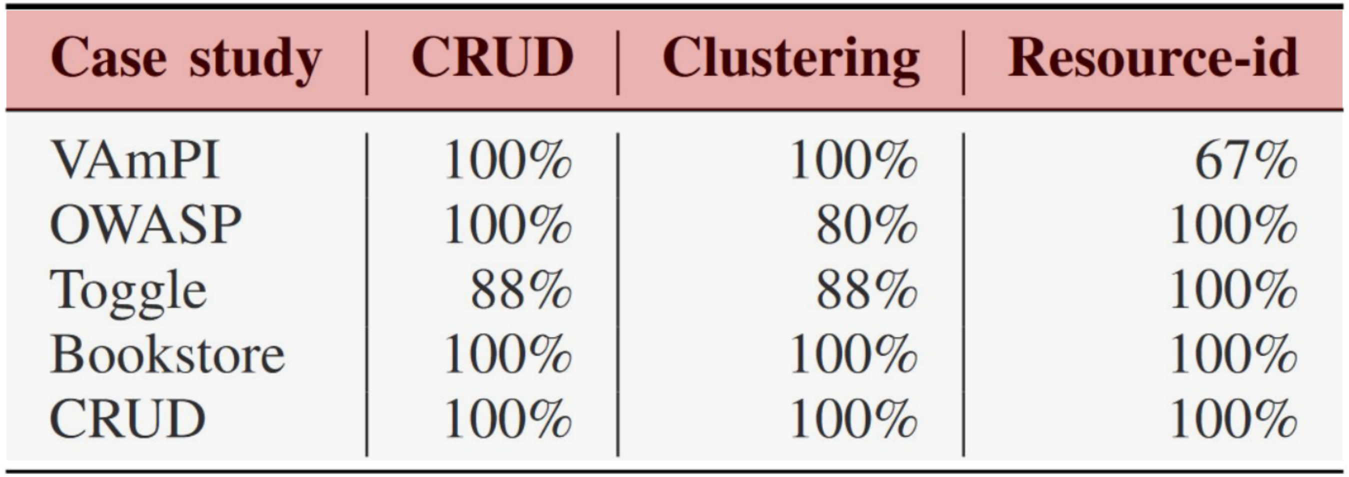
- **RQ1:** What is the accuracy of the automated identification of operations CRUD semantics, resource types, and resource-id parameters?
- **RQ2:** What is the accuracy in revealing mass assignment vulnerabilities in REST APIs?
- **RQ3:** Does the proposed approach to detect mass assignment vulnerabilities scale to large REST APIs?

Benchmark APIs

- Open-source
- Not read-only
- With OpenAPI specification

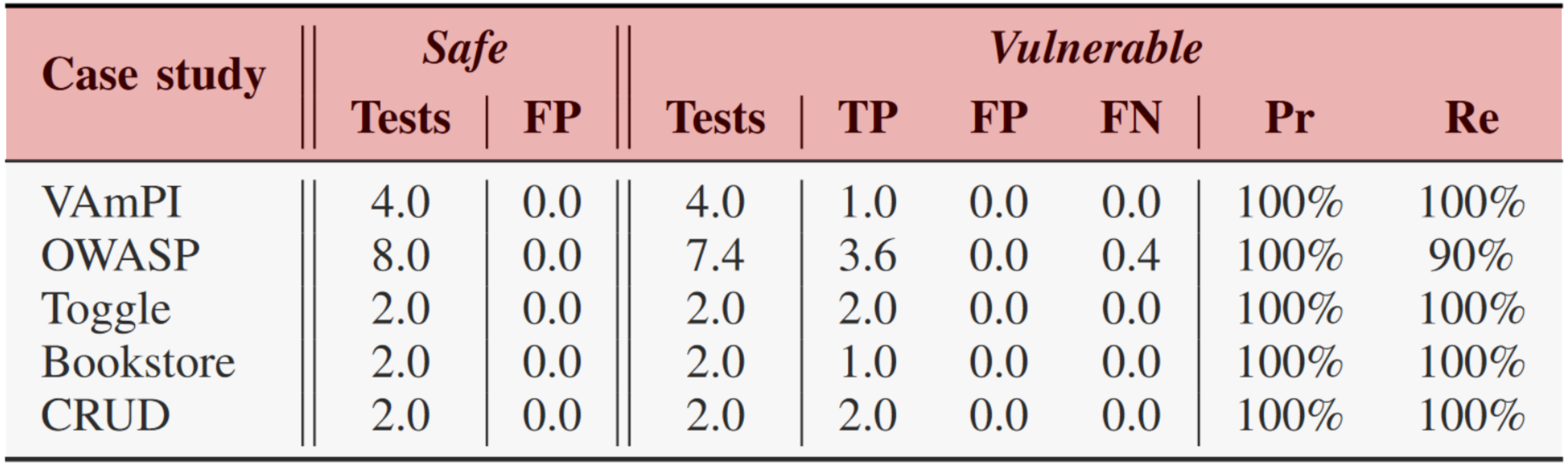
API	Prog. Lang.	REST framework	No. Of Operations	No. Of Vulnerabilities
VAmPI	Python	Flask	12	1
OWASP	Java	Spring	10	4
Toggle	ASP.NET	.NET Code	16	2
Bookstore	Java	Spring	5	1
CRUD	JavaScript	Express	4	2

Results: accuracy of CRUD extraction, clustering, and resource-id identification



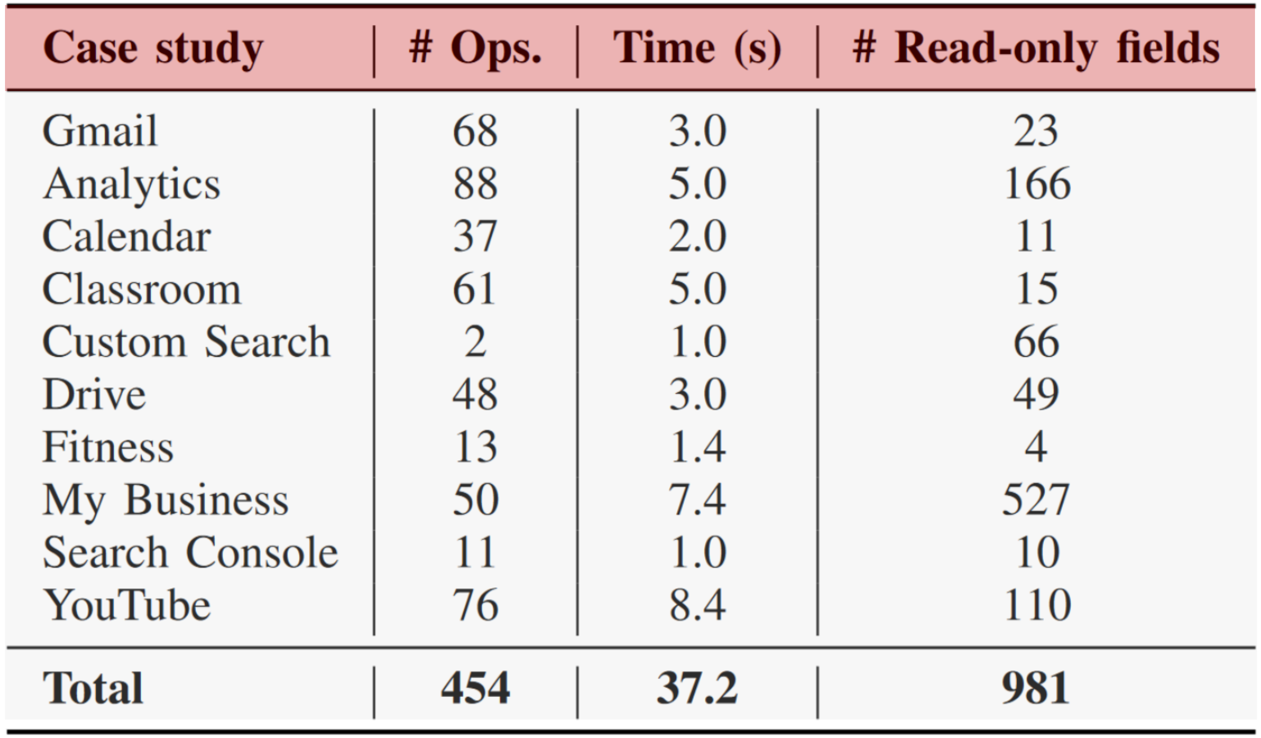
Case study	CRUD	Clustering	Resource-id
VAmPI	100%	100%	67%
OWASP	100%	80%	100%
Toggle	88%	88%	100%
Bookstore	100%	100%	100%
CRUD	100%	100%	100%

Results: accuracy of vulnerability detection



Case study	*Safe*		*Vulnerable*					
	Tests	FP	Tests	TP	FP	FN	Pr	Re
VAmPI	4.0	0.0	4.0	1.0	0.0	0.0	100%	100%
OWASP	8.0	0.0	7.4	3.6	0.0	0.4	100%	90%
Toggle	2.0	0.0	2.0	2.0	0.0	0.0	100%	100%
Bookstore	2.0	0.0	2.0	1.0	0.0	0.0	100%	100%
CRUD	2.0	0.0	2.0	2.0	0.0	0.0	100%	100%

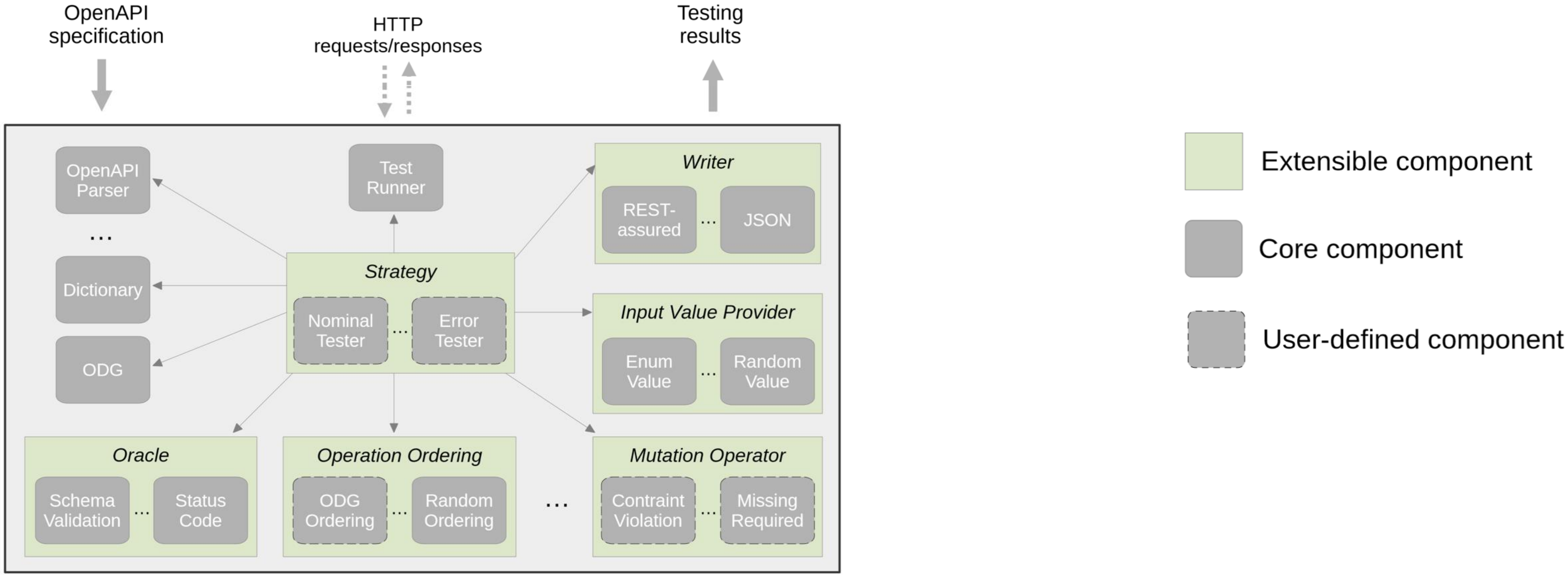
Results: scalability of the approach



Case study	# Ops.	Time (s)	# Read-only fields
Gmail	68	3.0	23
Analytics	88	5.0	166
Calendar	37	2.0	11
Classroom	61	5.0	15
Custom Search	2	1.0	66
Drive	48	3.0	49
Fitness	13	1.4	4
My Business	50	7.4	527
Search Console	11	1.0	10
YouTube	76	8.4	110
Total	**454**	**37.2**	**981**

Reusable research tools

RestTestGen Framework



OpenAPI specification

HTTP requests/responses

Testing results

OpenAPI Parser

…

Dictionary

ODG

Test Runner

Strategy

Nominal Tester … Error Tester

Writer

REST-assured … JSON

Input Value Provider

Enum Value … Random Value

Oracle

Schema Validation … Status Code

Operation Ordering

ODG Ordering … Random Ordering

…

Mutation Operator

Contraint Violation … Missing Required

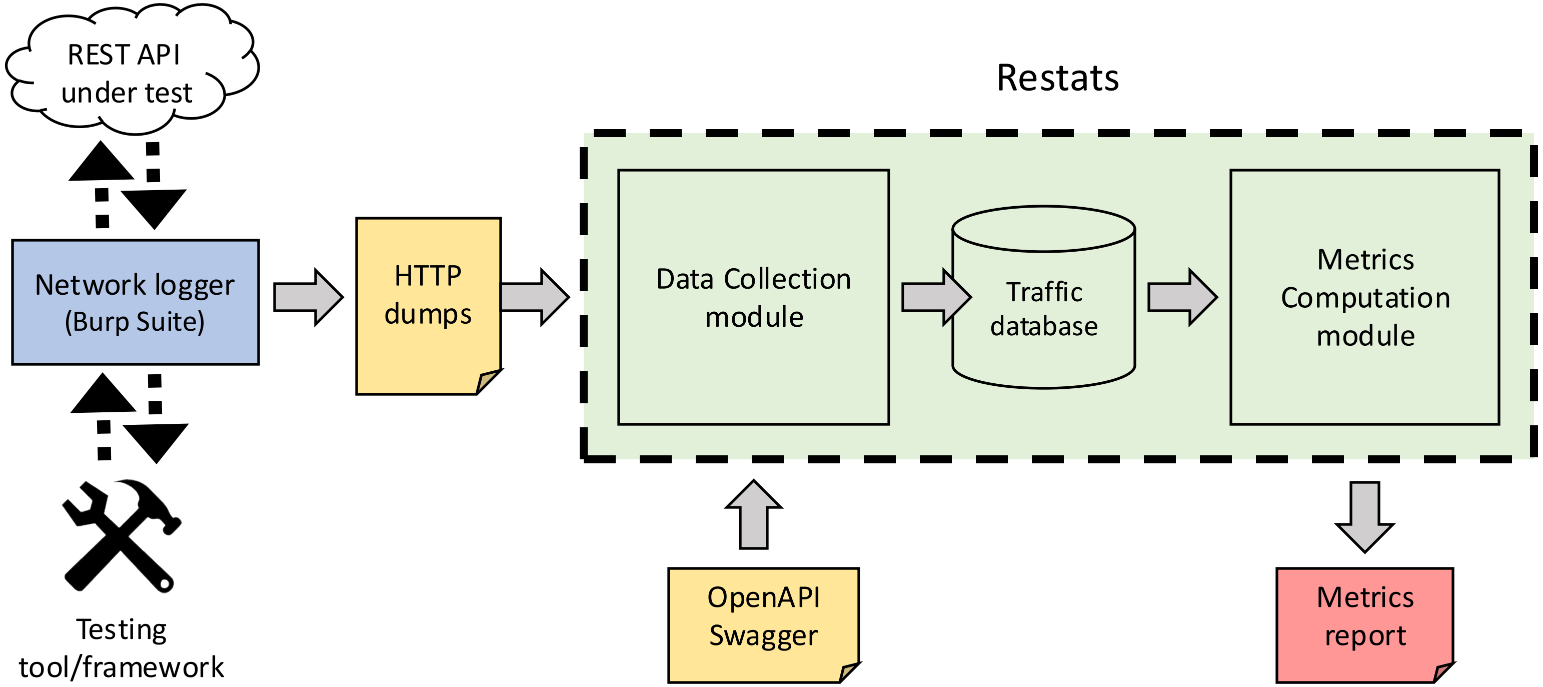
Extensible component

Core component

User-defined component

RestTestGen: An Extensible Framework for Automated Black-box Testing of RESTful APIs
D. Corradini, A. Zampieri, M. Pasqua, M. Ceccato
38th International Conference on Software Maintenance and Evolution (ICSME 2022), Tool Demo Track

Restats: test coverage tool



Restats: A Test Coverage Tool for RESTful APIs
D. Corradini, A. Zampieri, M. Pasqua, M. Ceccato
37th International Conference on Software Maintenance and Evolution (ICSME 2021), Tool Demo Track

Coverage metrics

Input coverage metrics

- Path coverage
- Operation coverage
- Parameter coverage
- Parameter value coverage
- Request content-type coverage

Output coverage metrics

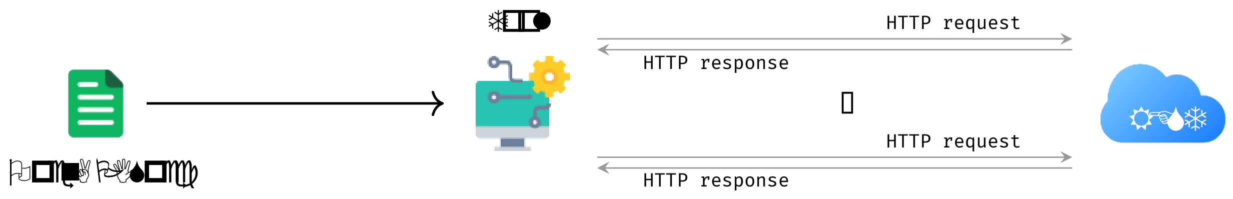
- Status code class coverage
- Status code coverage
- Response content-type coverage

Metrics are computed as defined by Martin-Lopez et al. [12], with adaptations in some cases to make them operative.

[12] A. Martin-Lopez, S. Segura, and A. Ruiz-Cortés, “Test coverage criteria for RESTful web APIs,” in Proceedings of the 10th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation, 2019, pp. 15–21.

Resarch on fuzzing REST APIs

- Defining effective testing strategies
- Find working, testable case studies
- Compute testing metrics
- Compare with the baseline



RESTler SwaggerFuzzer Morest DeepREST

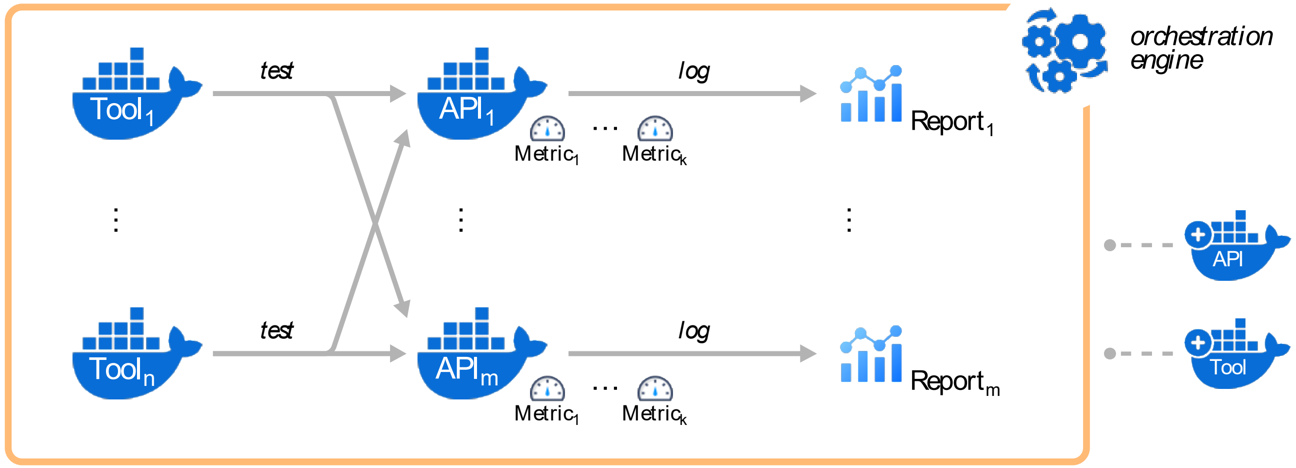
ARAT-RL RESTest EvoMaster APIFuzzer

Dredd bBOXRT RestCT TnTFuzzer

RestPL RestTestGen QuickREST Schemathesis

RestGym: a compassion testbed for researches

- Extensible container-based testing infrastructure
- Automated orchestration engine
- 6 Built-in test case generation tools and APIs
- 11 Built-in testing metrics
- Aggregate results and provide detailed reports



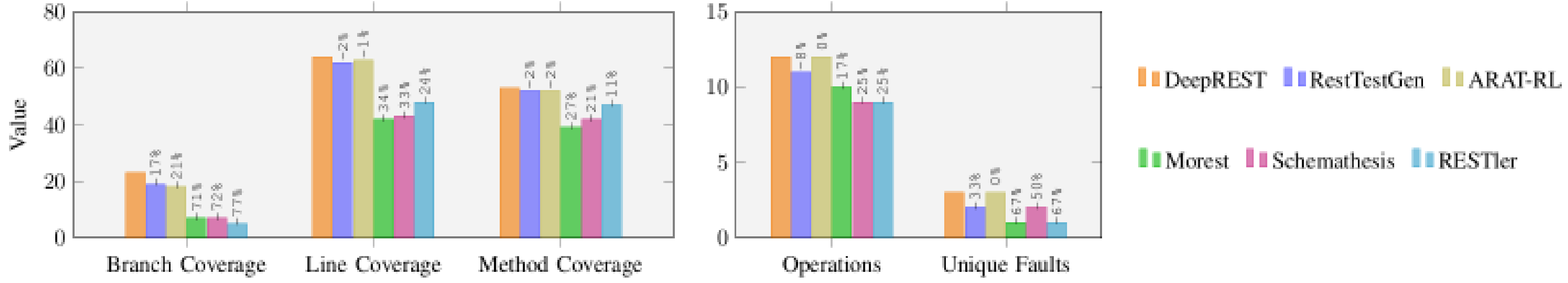
RESTgym: A Flexible Infrastructure for Empirical Assessment of Automated REST API Testing Tools

D. Corradini, M. Pasqua, M. Ceccato

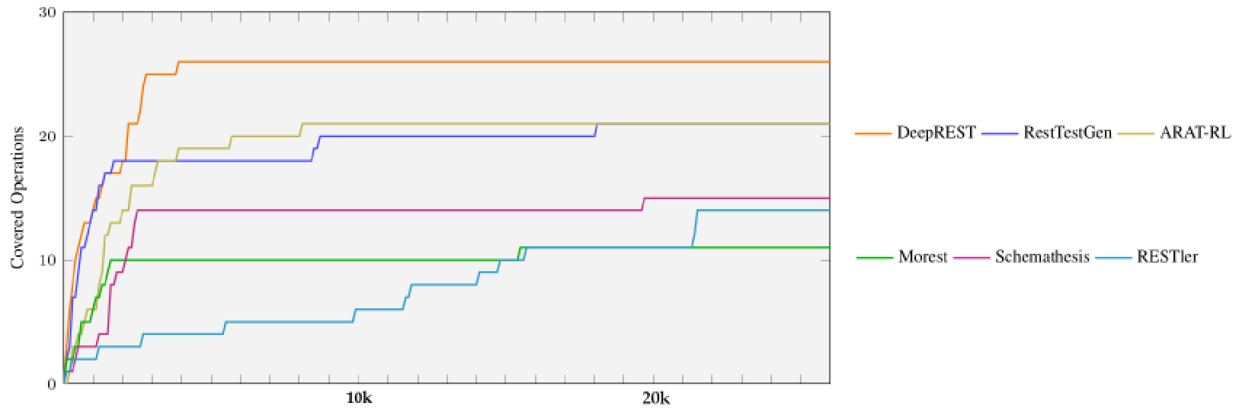
IEEE International Conference on Software Testing, Verification and Validation (ICST) 2025, Tool Demo

Testing reports

- Aggregate effectiveness results on all the APIs



- Efficiency trends on a single API



Contributions

- Fully automated generation of test cases to reveal mass assignment vulnerabilities in REST APIs
  - Black-box approach

- Open-source tool implementation
  - https://github.com/SeUniVr/RestTestGen

Conclusion

Challenges

- How to fully automate test case generation for REST APIs?
  - It must work on any REST API
  - No available knowledge on the business logic
- How to determine the optimal order to test operations?
  - Minimal time, maximal number of defects
- How to generate valid input values?
- How to determine if a test interaction exposed a faulty behavior or a vulnerability (i.e., the test case passed or failed)?

- Possibly, in the most efficient and effective way

Ongoing work

- New algorithms to decide the operations testing order
- New techniques for input values generation
  - E.g., ML/AI based techniques
- Support for new vulnerabilities detection
- OpenAPI specifications not available, outdated, or incomplete
- Usability of automatically generated tests

Thank you!

RTG

Finanziato dall'Unione europea
NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA

Fuzzing Web APIS for Functional and Security Testing

Mariano Ceccato

mariano.ceccato@univr.it

This work has been done in collaboration mainly with **Davide Corradini** and **Michele Pasqua**

UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA