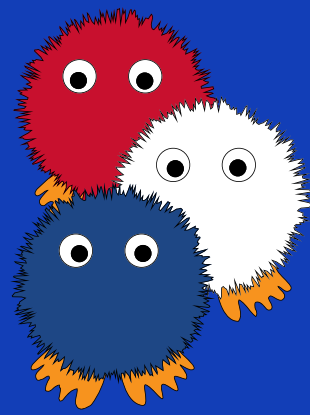
TNO innovation for life

Automated REST API vulnerability detection

with



Wuppie Fuzz

$ whoami
Thomas Rooijakkers

Interests

Focus of today

- Cybersecurity-by-design
- Fuzzing
- DevSecOps → SecDevOps

Job

- Senior Scientist Integrator at TNO
  - Cybersecurity researcher

Overview

TNO: Innovation for life

Facts

- The Netherlands Organisation for applied scientific research
- Not-for-profit
- Independent statutory (by law) research organisation
- Approx. 5000 employees

Goals

- Bridge the gap between academia and industry
- Support companies and governments

Mission

- Create impactful innovations for the sustainable wellbeing and prosperity of society

Key principles

- In fuzzing, unexpected, faulty and *seemingly* random inputs are fed to the software under test in an attempt to trigger unexpected, breaking, behaviour.

- Driven by
  - High volume (many different inputs)
  - Decreased testing bias
  - Testing the unexpected
  - Automation

- Conceptually simple, complex in practice

Image Credits: Paweł Rzepa

Key principles EXAMPLE

- In fuzzing, unexpected, faulty and *seemingly* random inputs are fed to the software under test in an attempt to trigger unexpected, breaking, beh

- Driven by
  - High volume (many different in
  - Decreased testing bias
  - Testing the unexpected
  - Automation

- Conceptually simple, complex in

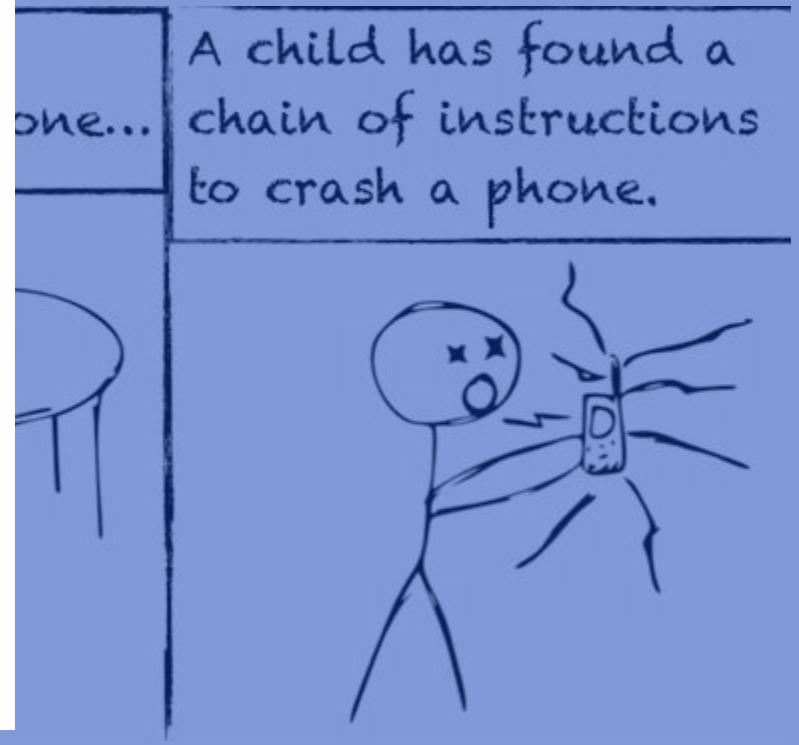


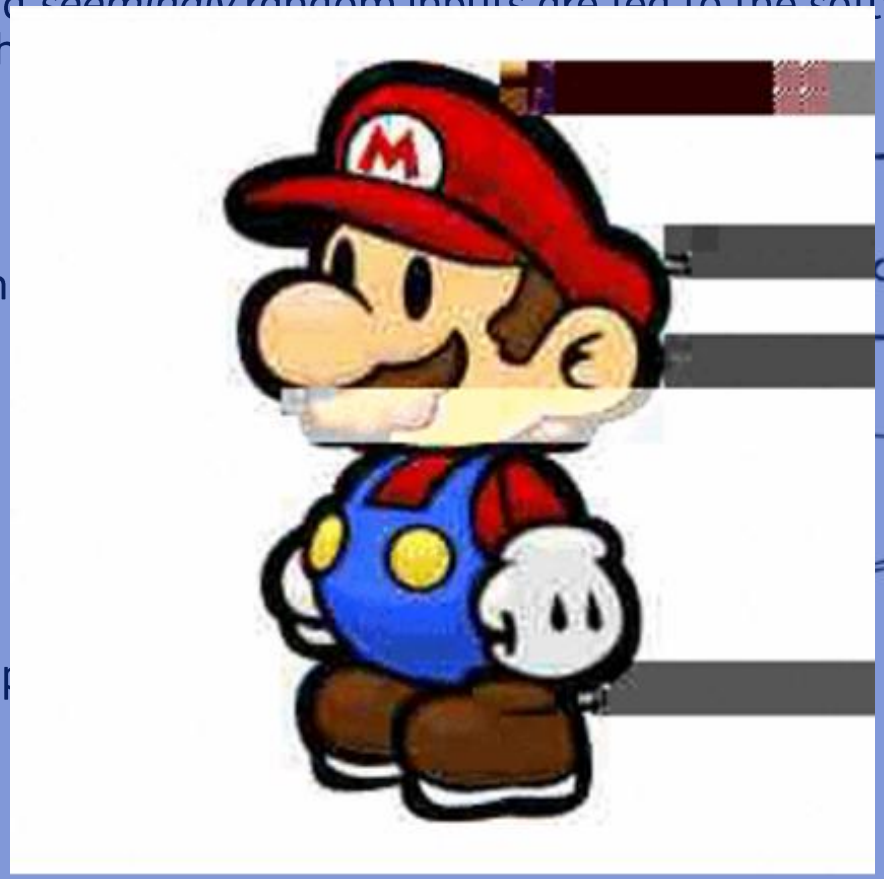
Image Credits: Paweł Rzepa

Key principles EXAMPLE

- In fuzzing, unexpected, faulty and *seemingly* random inputs are fed to the software under test in an attempt to trigger unexpected, breaking, beh

- Driven by
  - High volume (many different in
  - Decreased testing bias
  - Testing the unexpected
  - Automation

- Conceptually simple, complex in p



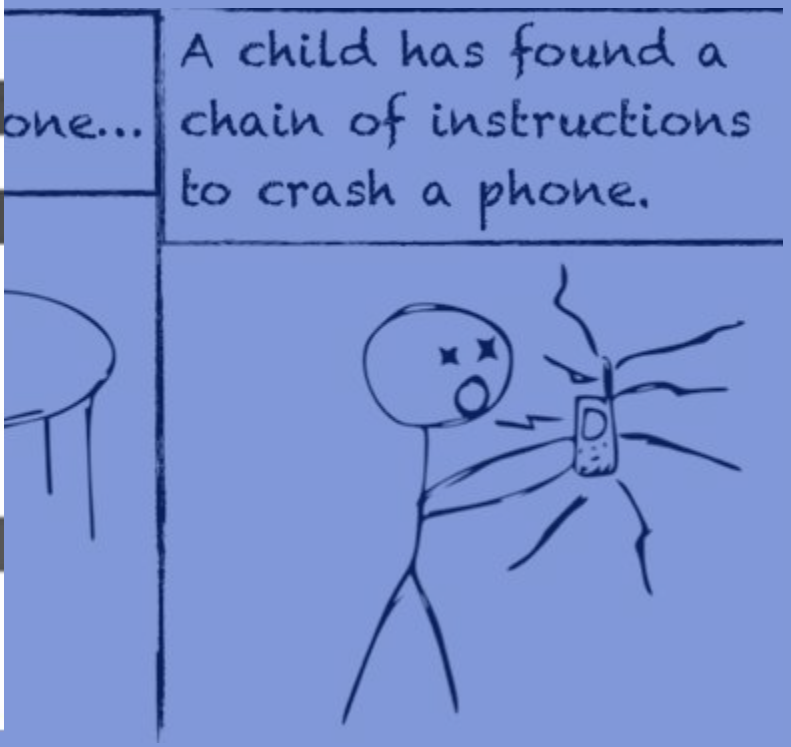


Image Credits: Paweł Rzepa

When / where?

Some common misconceptions

- Fuzzing is a technique similar to stress testing
- Fuzzing is used to attack a system
- Fuzzing is most valuable to enhance pen-testing
- Fuzzing is only for security testing

When / where?

<u>During development</u>

- Unit testing
- Integration testing
- In CI (continuous integration)

<u>Pre-release</u>

- Acceptance testing
- Penetration testing
- Supply chain security
- OSS security testing

<u>Post-release</u>

- Security incidents
- Legacy software
- Continuous & maintenance
- Regression testing

Why?

Detecting flaws in software increases

- Reliability
- Security

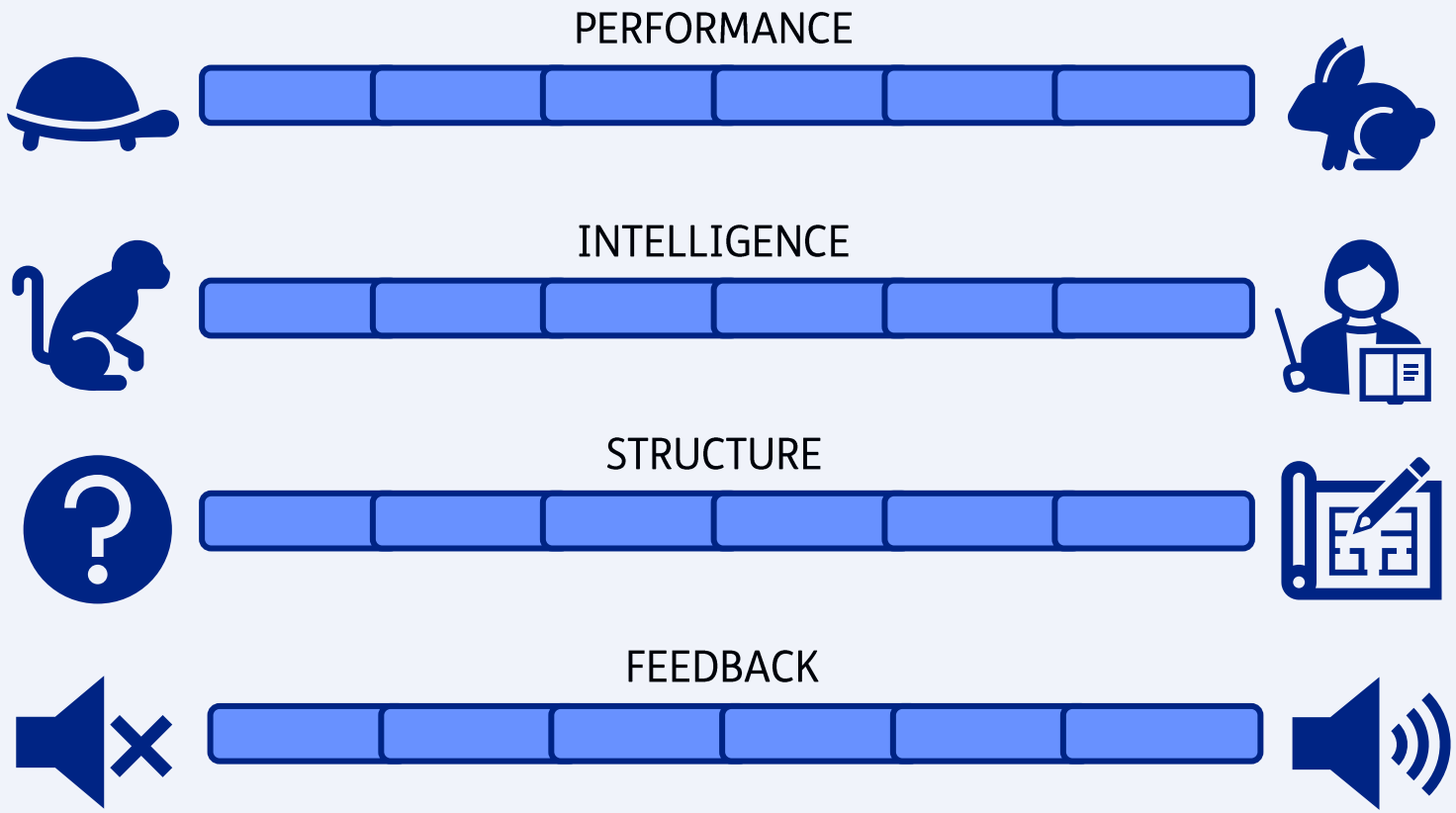
Fuzzer results are repeatable

- Simplifies debugging and patching (payload available)
- Aids regression testing

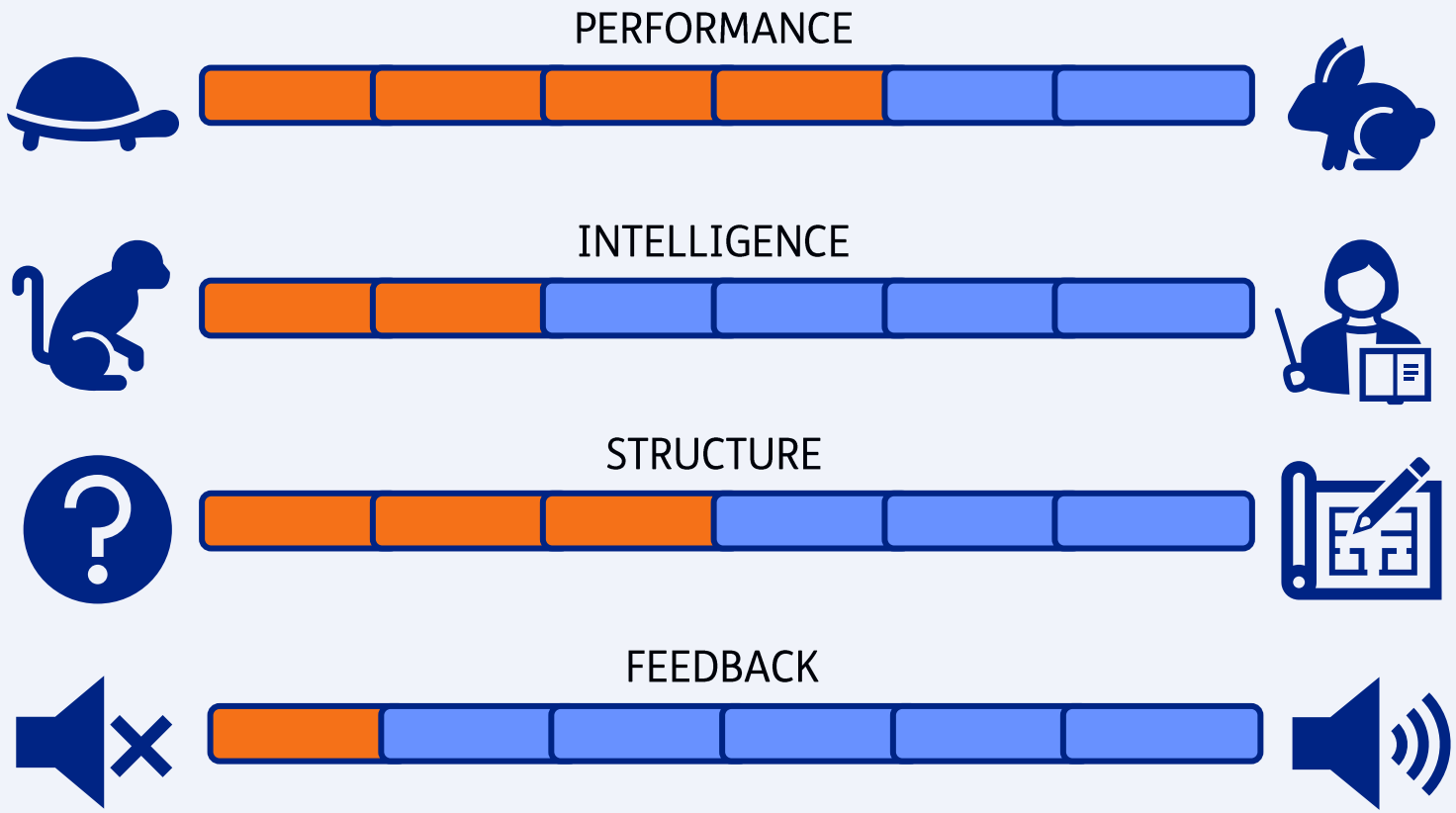
After setup

- Cheap (requires minimal effort from human experts)
- Automation
- Fuzzers can run in the background (24/7)

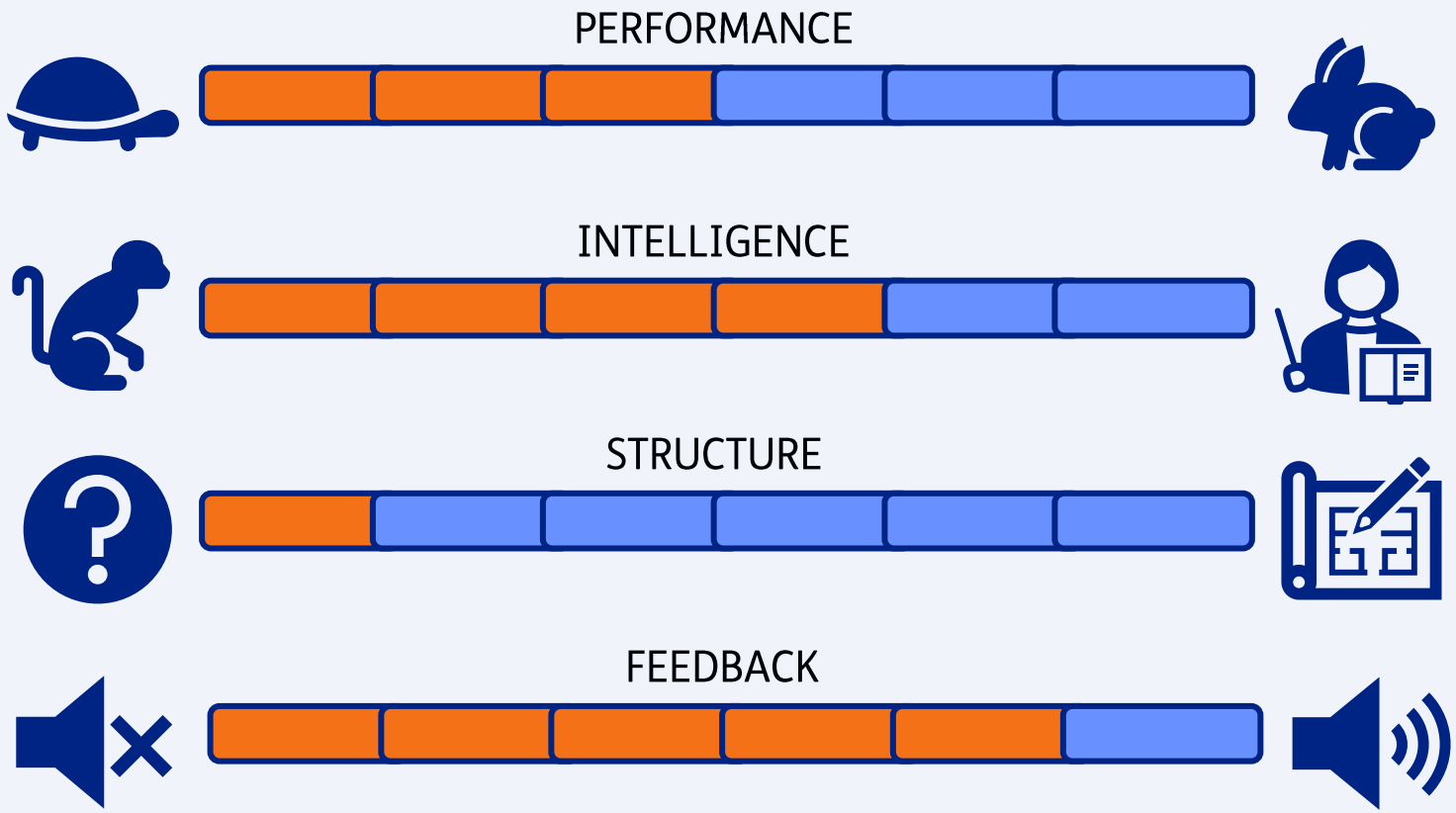
Trade-offs



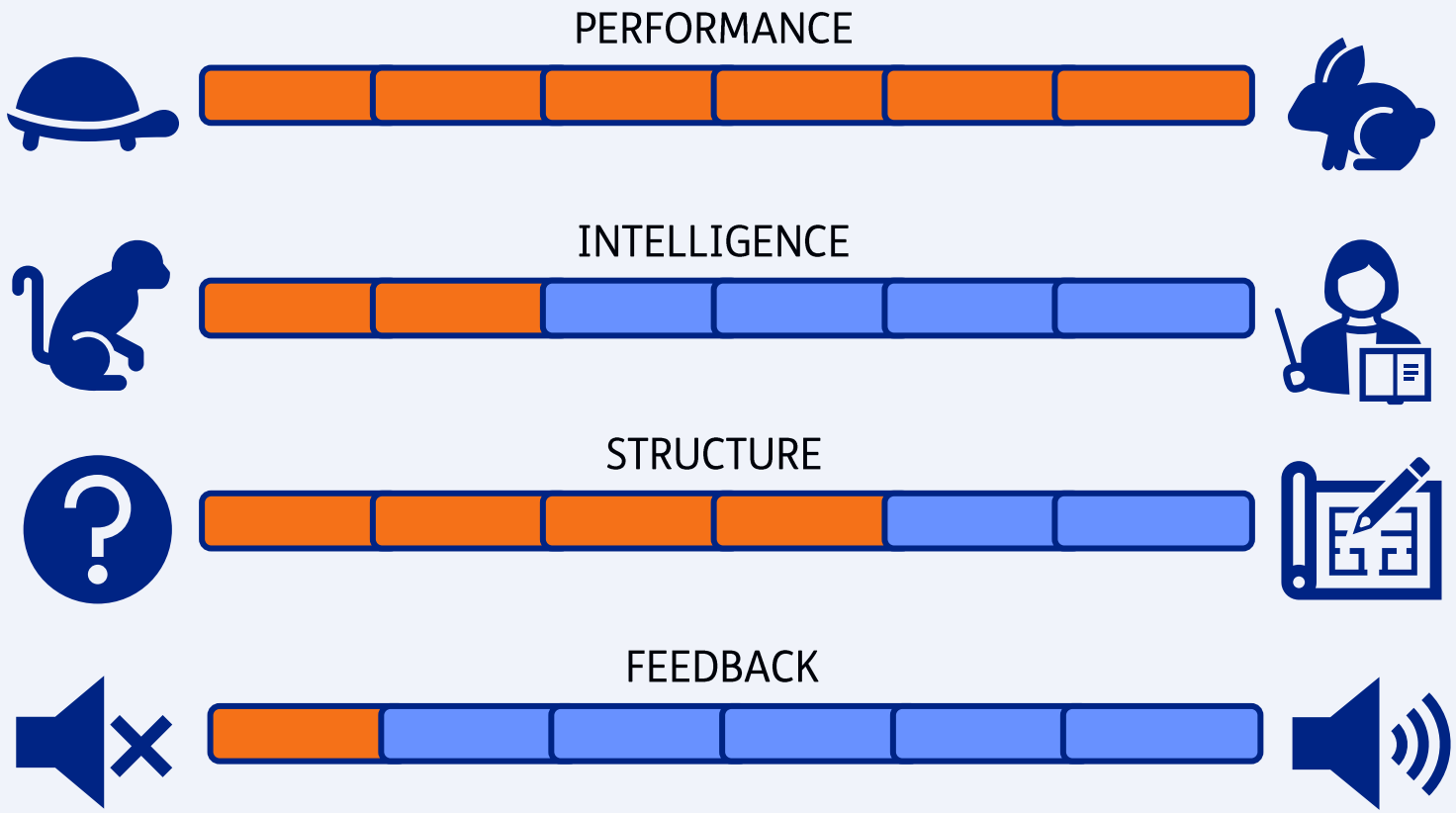
Trade-offs



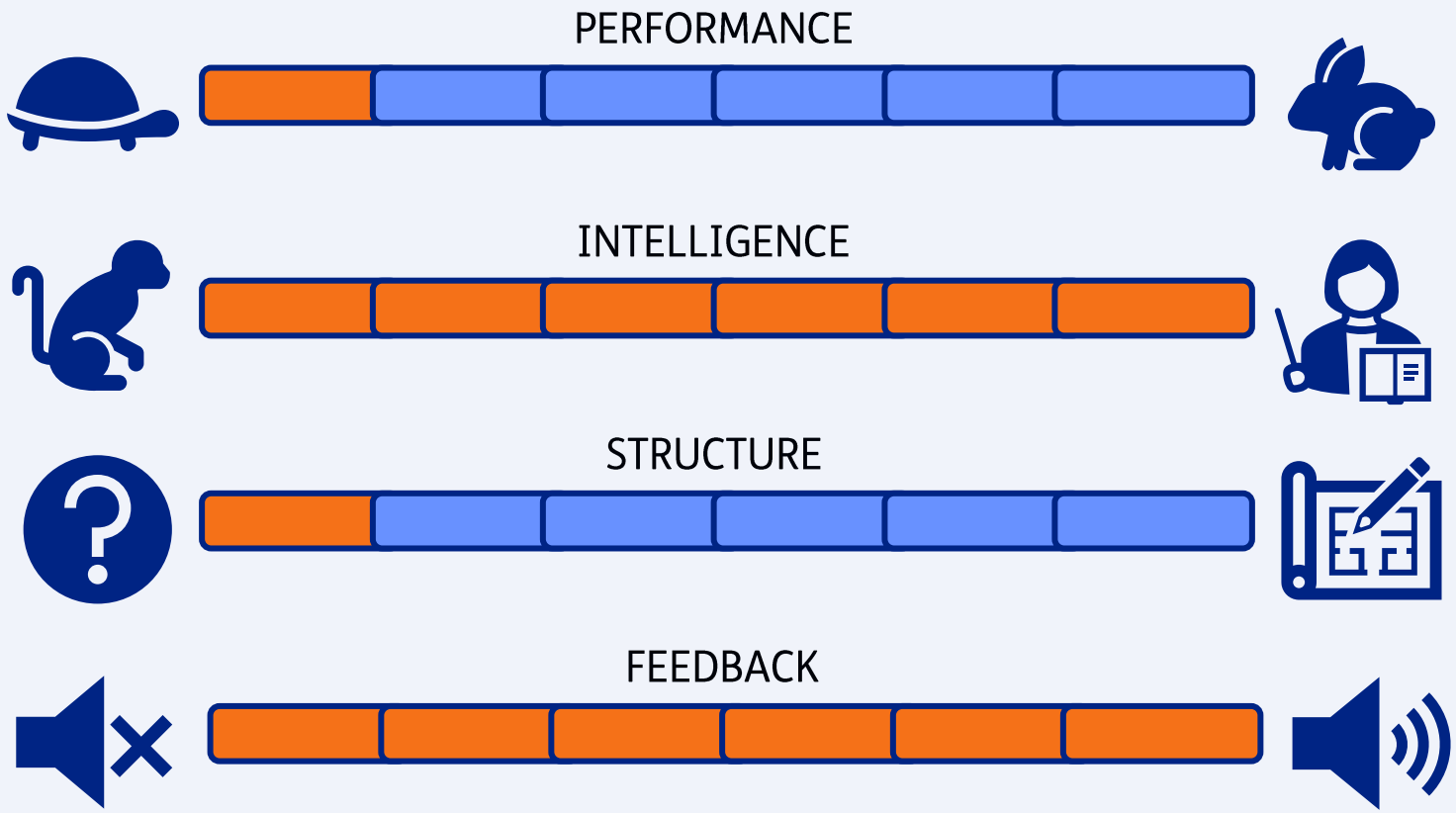
Trade-offs



Trade-offs



Trade-offs



Why?

Fuzzing REST APIs

Business Perspective


- Common attack service
  - Often exposed (publicly)
- Loads of (user) input handling
- Enhance trust and reduce risks (security posture ++)

For the Techies

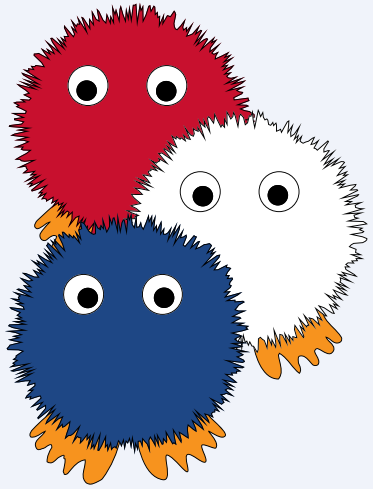

- Uncover hidden bugs
  - E.g., unhandled exceptions, crashes, unexpected / undefined behaviour
- Automation!
- Complements other testing methods

Launch

🚀 WuppieFuzz released 🎉 – September 2024

Coverage-guided REST API fuzzing

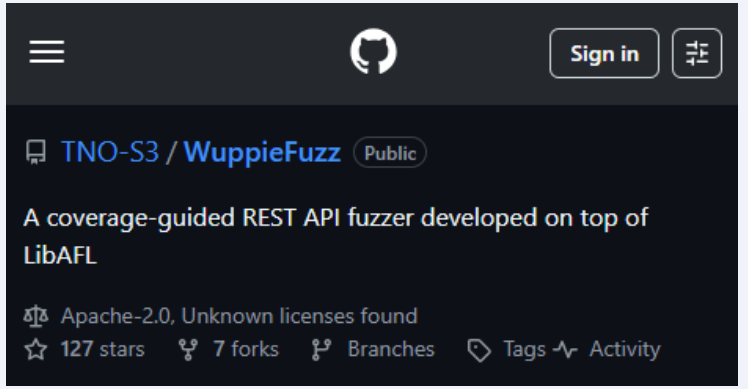
- Test your (public-facing) interfaces at ease
- Supports black box, grey box and white box testing
- Code coverage as well as endpoint coverage



Wuppie Fuzz

Key focus 🔍

- Ease-of-use
- Interpretation of results
- Modularity and extensibility
- Open source



open source initiative
Approved License

https://tno.to/wuppiefuzz

High-level video

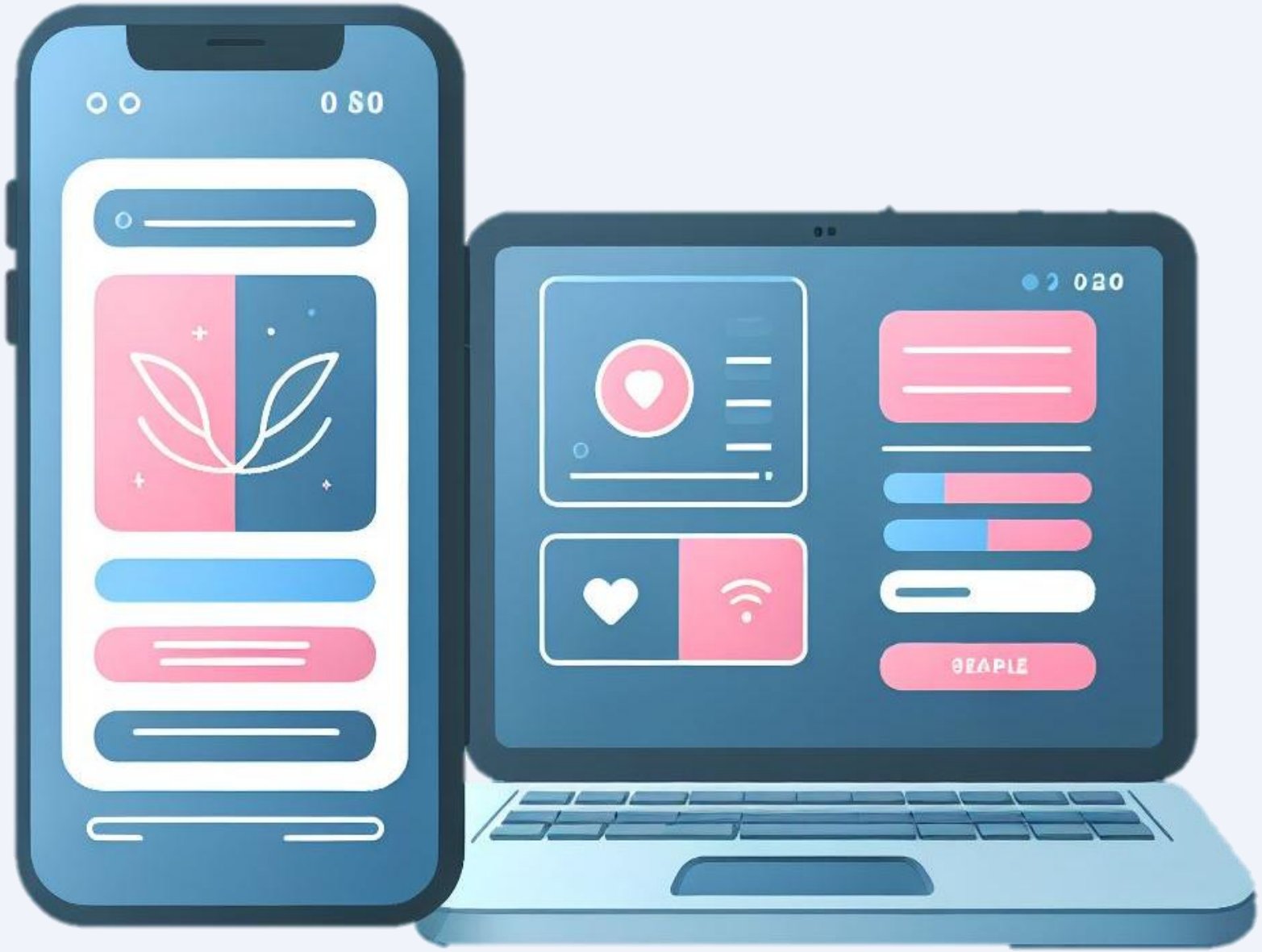
Key challenges in fuzzing REST APIs

1. Structured input required
   - Create a harness / standardised means of communication with target API?
   - Utilise specification?

2. Notion of state and interrelations
   - Use create, read, update, delete (CRUD)?

3. Intelligent runtime guidance using feedback
   - Use coverage information?

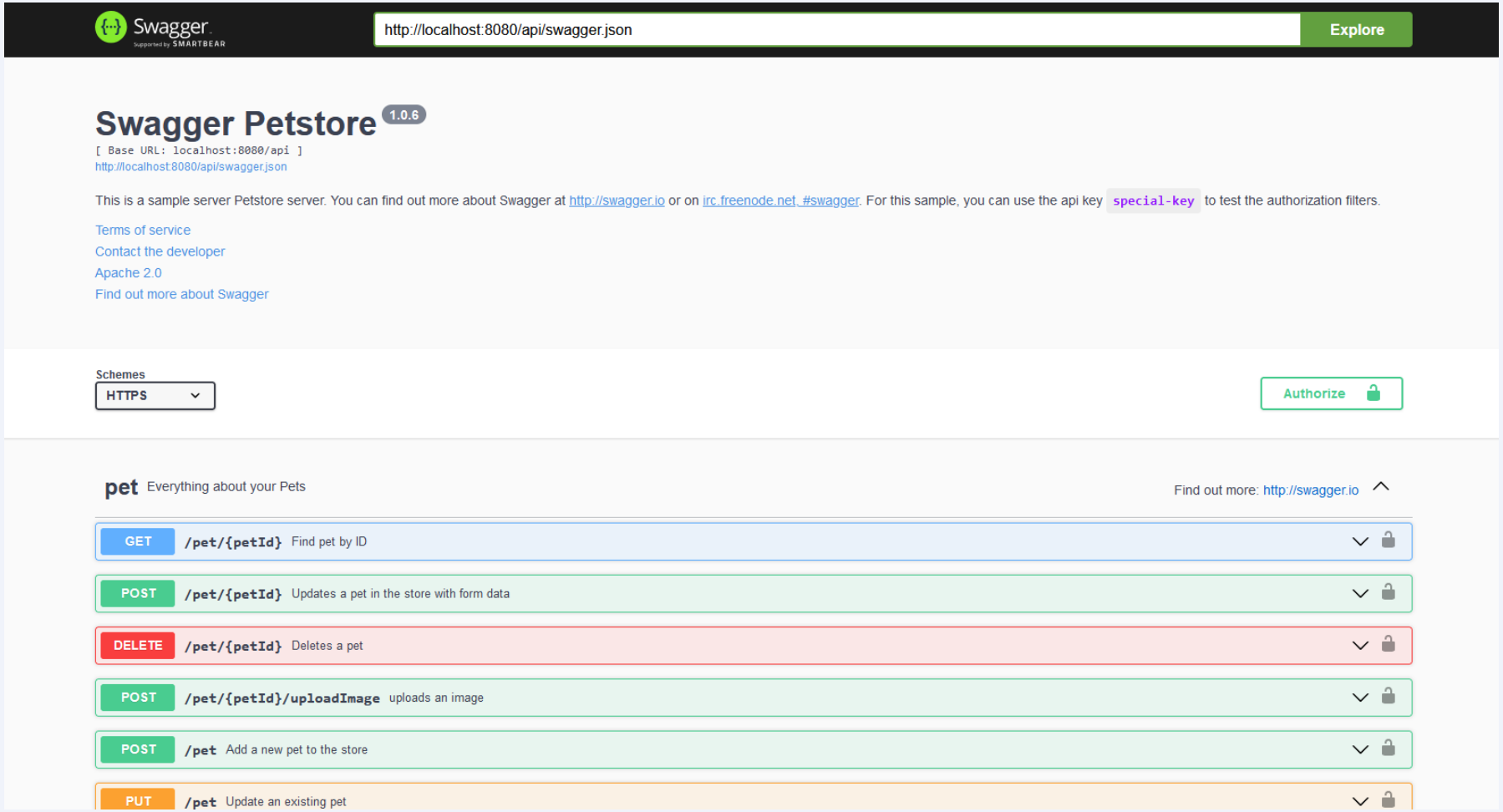
Harness



#1

Harness

We have the spec!

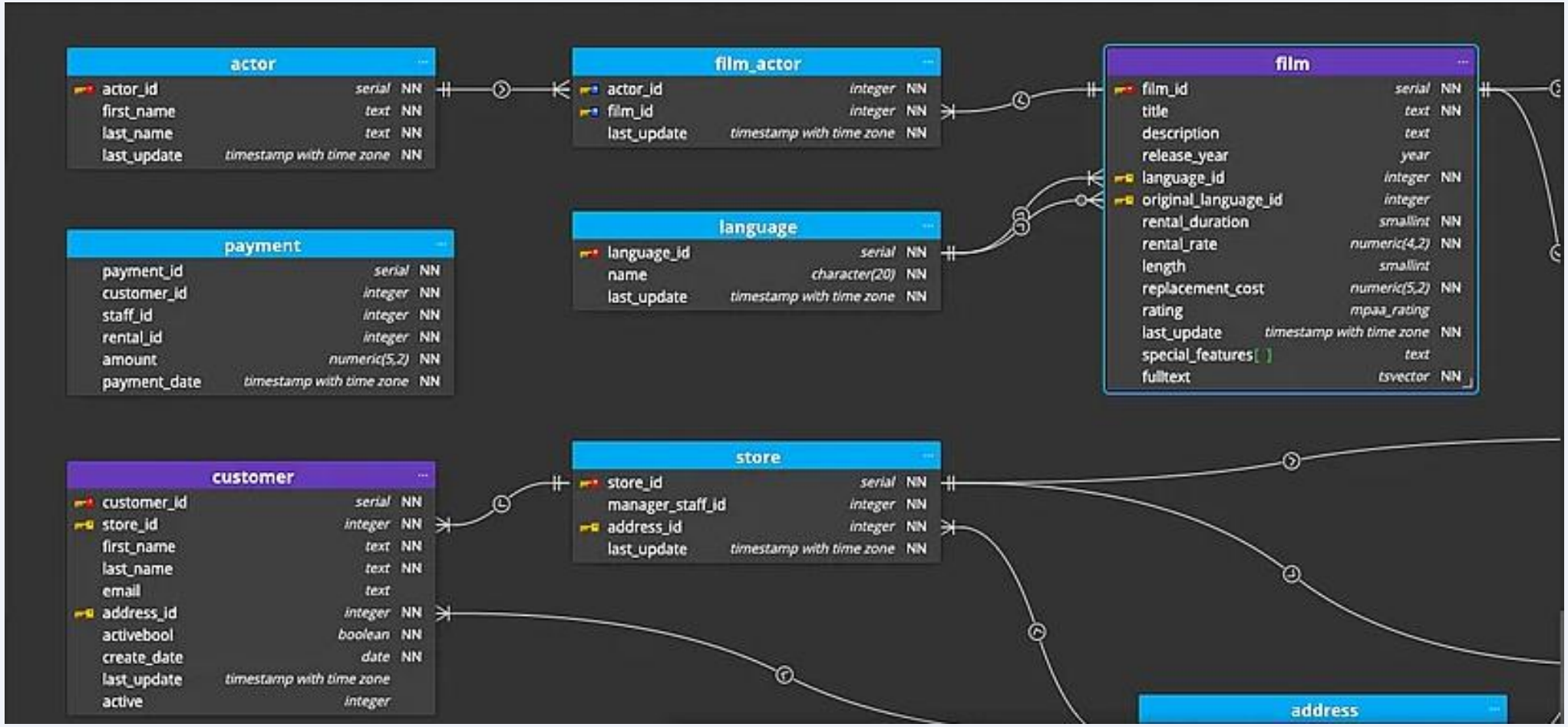


Harness

Spec leads to code

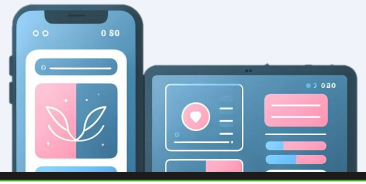

```python
@app.route('/pets/<int:pet_id>', methods=['PUT'])
def update_pet(pet_id):
    pet = next((p for p in pets if p['id'] == pet_id), None)
    if pet:
        pet.update(request.json)
        return jsonify(pet)
    else:
        return jsonify({"error": "Pet not found"}), 404

@app.route('/pets/<int:pet_id>', methods=['DELETE'])
def delete_pet(pet_id):
    global pets
    pets = [p for p in pets if p['id'] != pet_id]
    return jsonify({"message": "Pet deleted"}), 200

@app.route('/stores', methods=['GET'])
def get_stores():
    return jsonify(stores)

@app.route('/stores/<int:store_id>', methods=['GET'])
def get_store(store_id):
    store = next((s for s in stores if s['id'] == store_id), None)
    if store:
        return jsonify(store)
    else:
        return jsonify({"error": "Store not found"}), 404

@app.route('/stores', methods=['POST'])
def create_store():
    new_store = request.json
    new_store['id'] = len(stores) + 1
    stores.append(new_store)
    return jsonify(new_store), 201

@app.route('/stores/<int:store_id>', methods=['PUT'])
def update_store(store_id):
    store = next((s for s in stores if s['id'] == store_id), None)
    if store:
        store.update(request.json)
        return jsonify(store)
    else:
        return jsonify({"error": "Store not found"}), 404
```

Interaction with persistent storage!

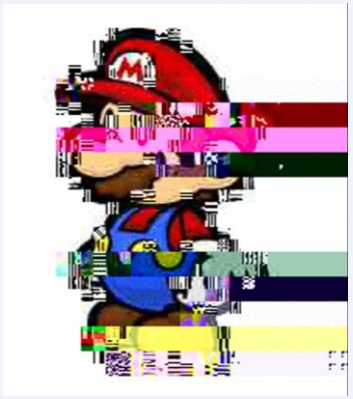


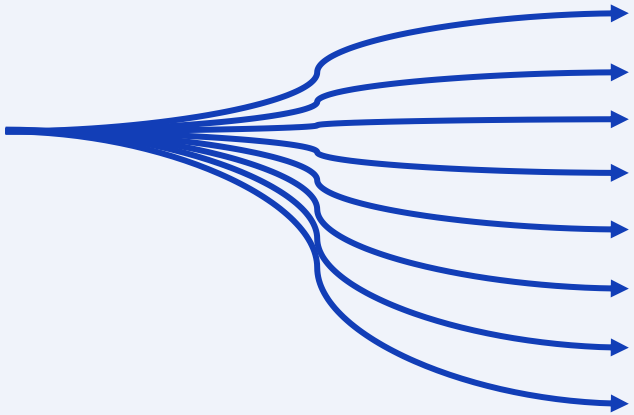
Harness

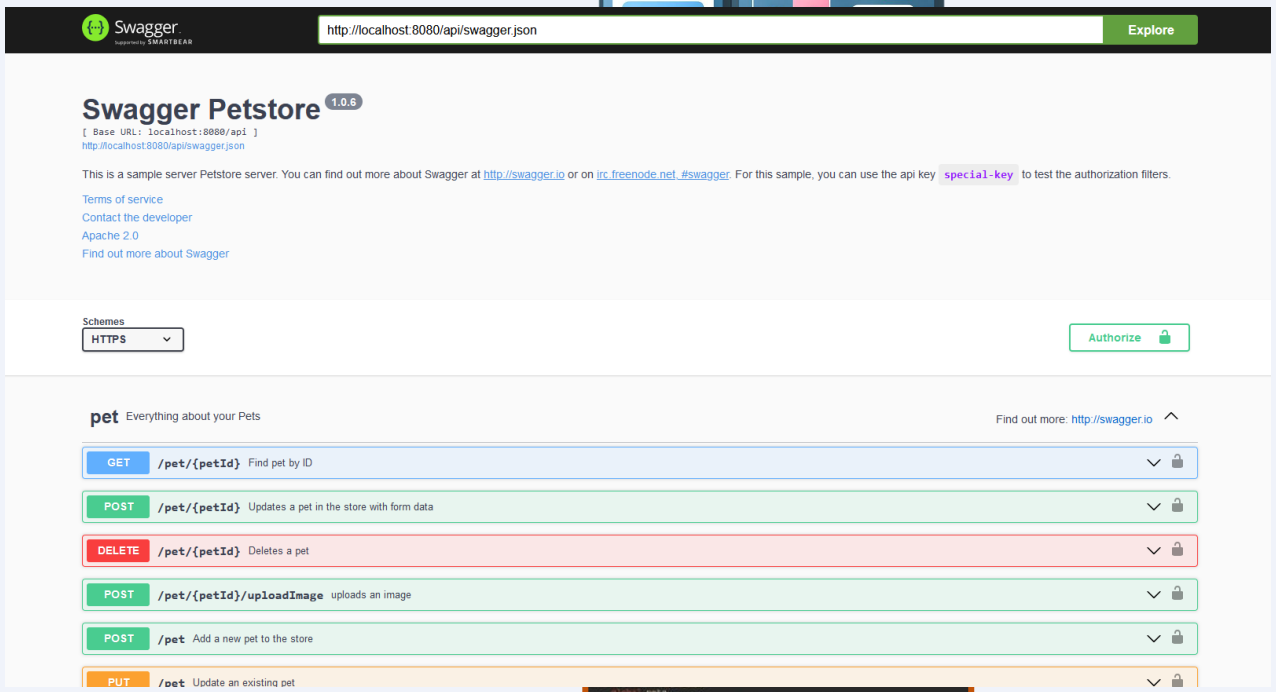
Fuzzing a REST API back-end

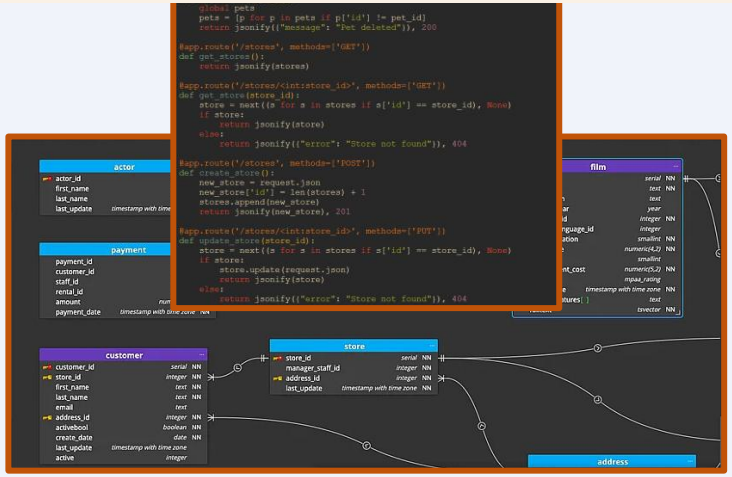


#1



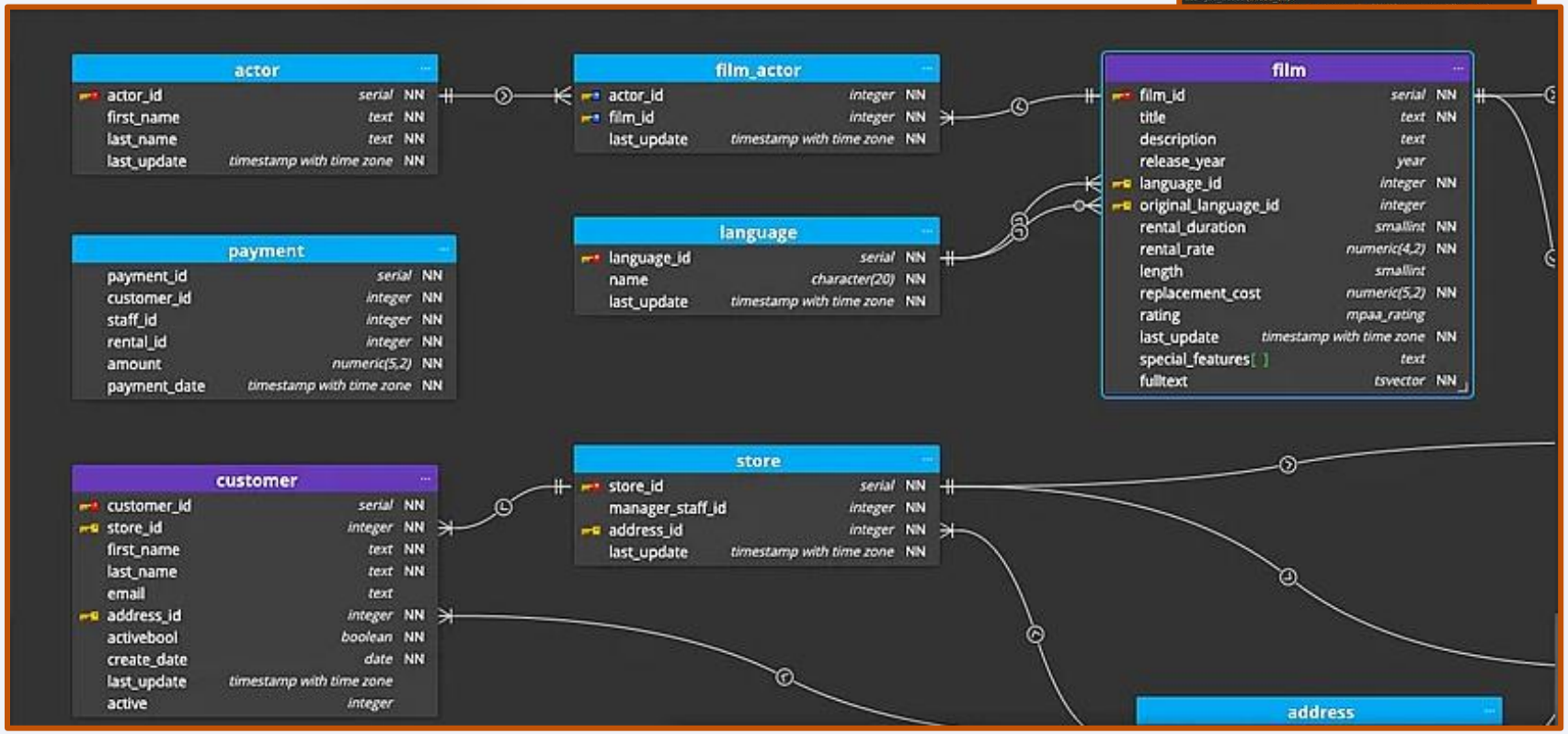






Statefulness

Fuzzing a REST API



#2

Fuzzing a REST API: fuzzer input and resources

Create Read Update Delete

Pet ID

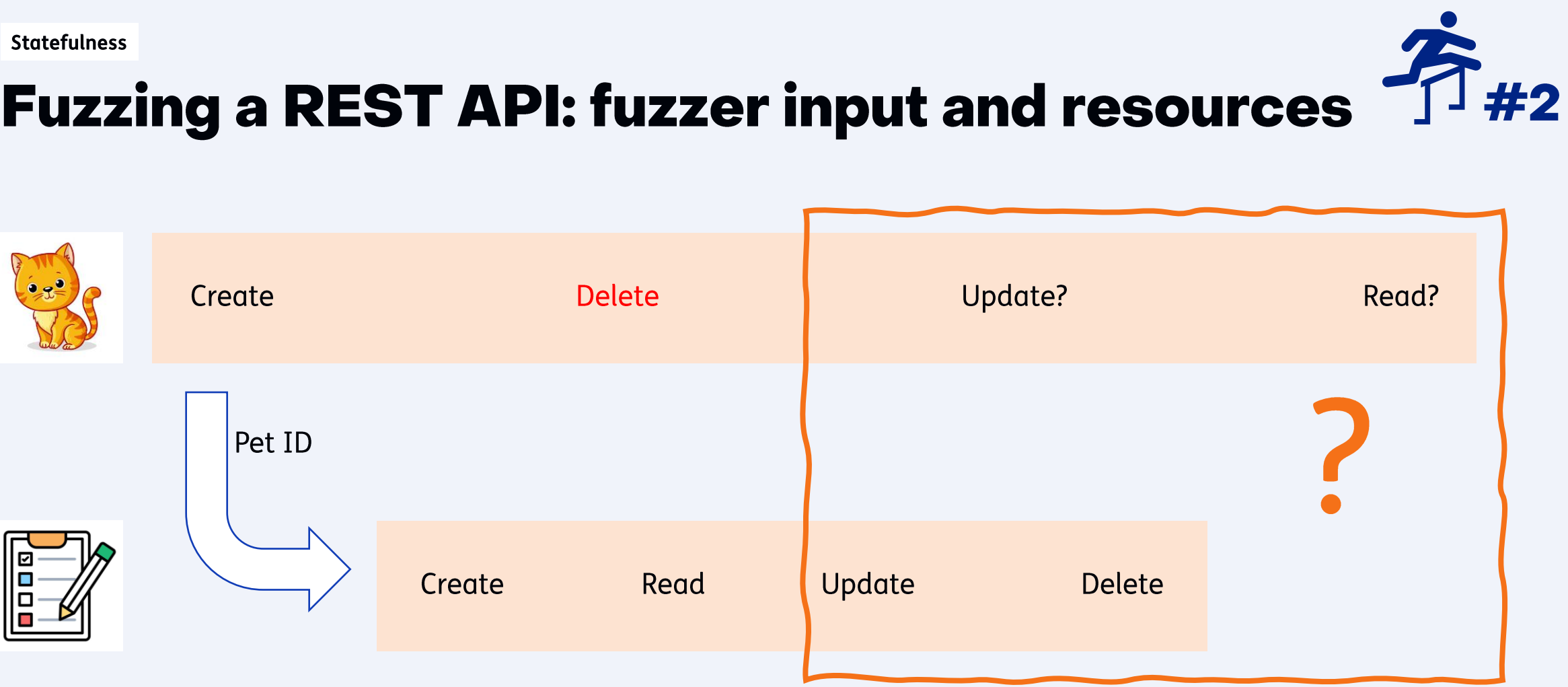
Create Read Update Delete

Statefulness

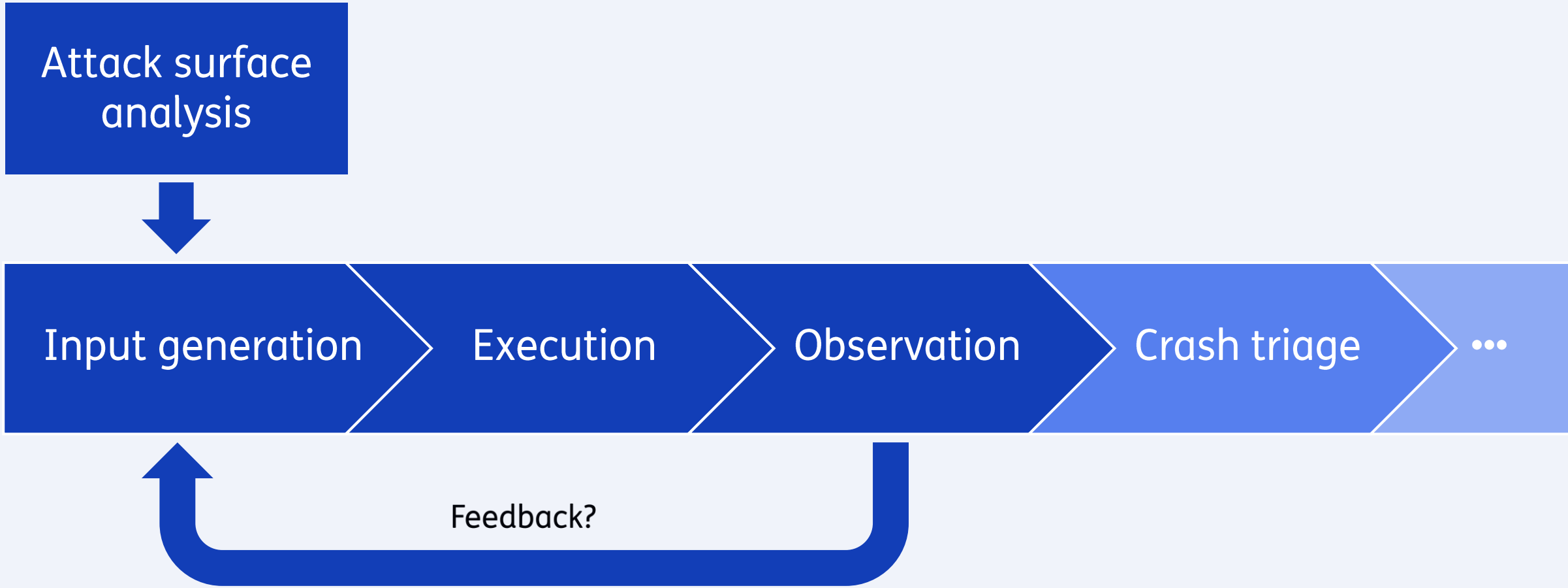
Corpus entry in action


```
PC-46128 > erieke > Targets/swagger-petstore > ./wuppiefuzz reproduce corpus/reference-corpus-entry --openapi-spec petstore_openapi.yml
Input file "corpus/reference-corpus-entry" contains 3 inputs
[2025-09-17T09:05:43Z INFO  wuppiefuzz::reproducer]
    ─────
    Sending request:
    POST /petContents in body: {"name": LeafValue(String("doggie")), "photoUrls": Array([LeafValue(String("🎵"))])}
[2025-09-17T09:05:43Z INFO  wuppiefuzz::reproducer] Converted to CURL command:
    echo eyJuYW1lIjoiZG9nZ2llIiwicGhvdG9VcmxzIjpbIvCfjrUiXX0= | \
    base64 --decode | \
    curl http://localhost:8080/api/pet? \
        --request POST \
        --header 'accept: application/json' \
        --header 'content-type: application/json' \
        --data @-
[2025-09-17T09:05:43Z INFO  wuppiefuzz::reproducer] Request successful (200 OK)
[2025-09-17T09:05:43Z INFO  wuppiefuzz::reproducer] Response matches specification
[2025-09-17T09:05:43Z INFO  wuppiefuzz::reproducer] Response contents printed below:
    {"id":11,"name":"doggie","photoUrls":["\uD83C\uDFB5"],"tags":[]}
[2025-09-17T09:05:43Z INFO  wuppiefuzz::reproducer]
    ─────
    Sending request:
    GET /pet/{petId}
      petId in Path: parameter id from request 0
[2025-09-17T09:05:43Z INFO  wuppiefuzz::reproducer] Converted to CURL command:
    curl http://localhost:8080/api/pet/11? \
        --request GET \
        --header 'accept: application/json'
[2025-09-17T09:05:43Z INFO  wuppiefuzz::reproducer] Request successful (200 OK)
[2025-09-17T09:05:43Z INFO  wuppiefuzz::reproducer] Response matches specification
[2025-09-17T09:05:43Z INFO  wuppiefuzz::reproducer] Response contents printed below:
    {"id":11,"name":"doggie","photoUrls":["\uD83C\uDFB5"],"tags":[]}
```

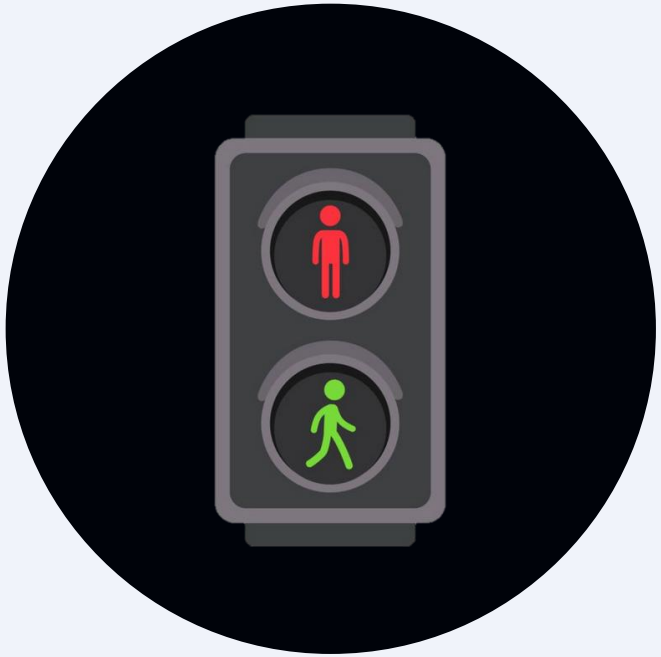
Fuzzing a REST API: fuzzer input and resources

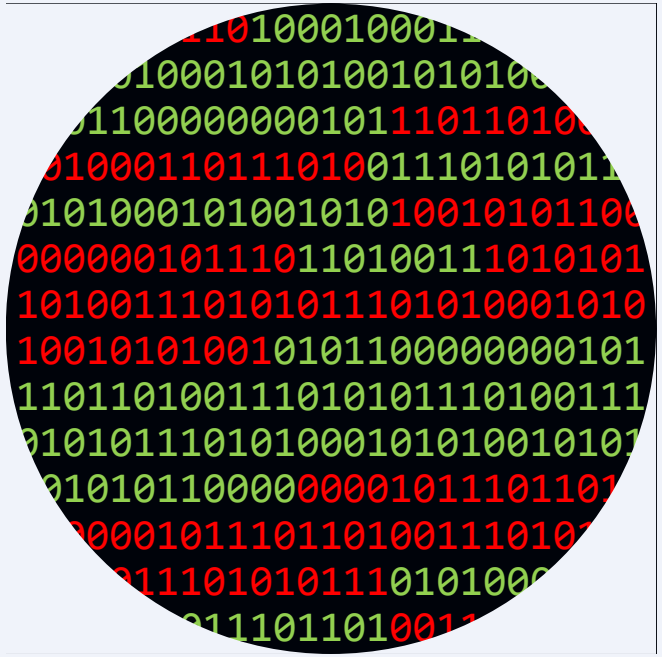


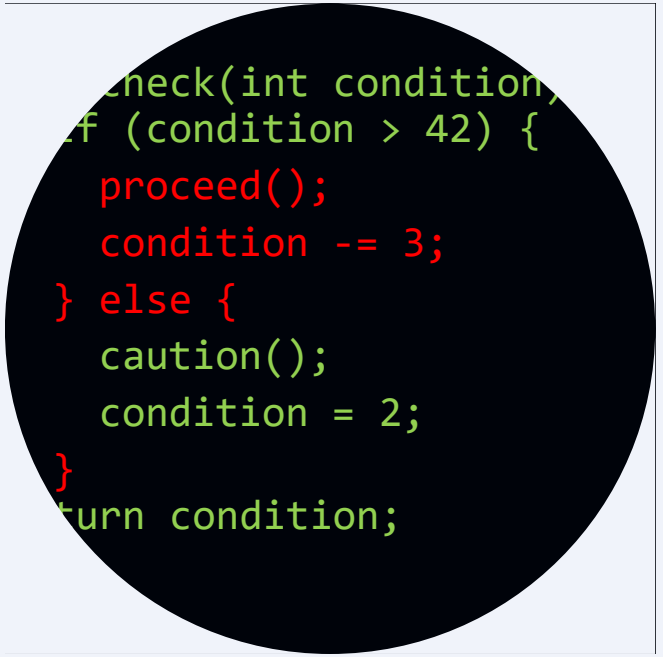
Fuzzing procedure



Coverage?







Less (internal) knowledge on software required

More information available for fuzzer guidance

Coverage: the minimal approach (black box)

- [x]
- [x]
- [x]
- [x]

PUT `/pet` Update an existing pet.

Responses

Code	Description	Links
200	Successful operation	*No links*
400	Invalid ID supplied	*No links*
404	Pet not found	*No links*
422	Validation exception	*No links*
default	Unexpected error	*No links*

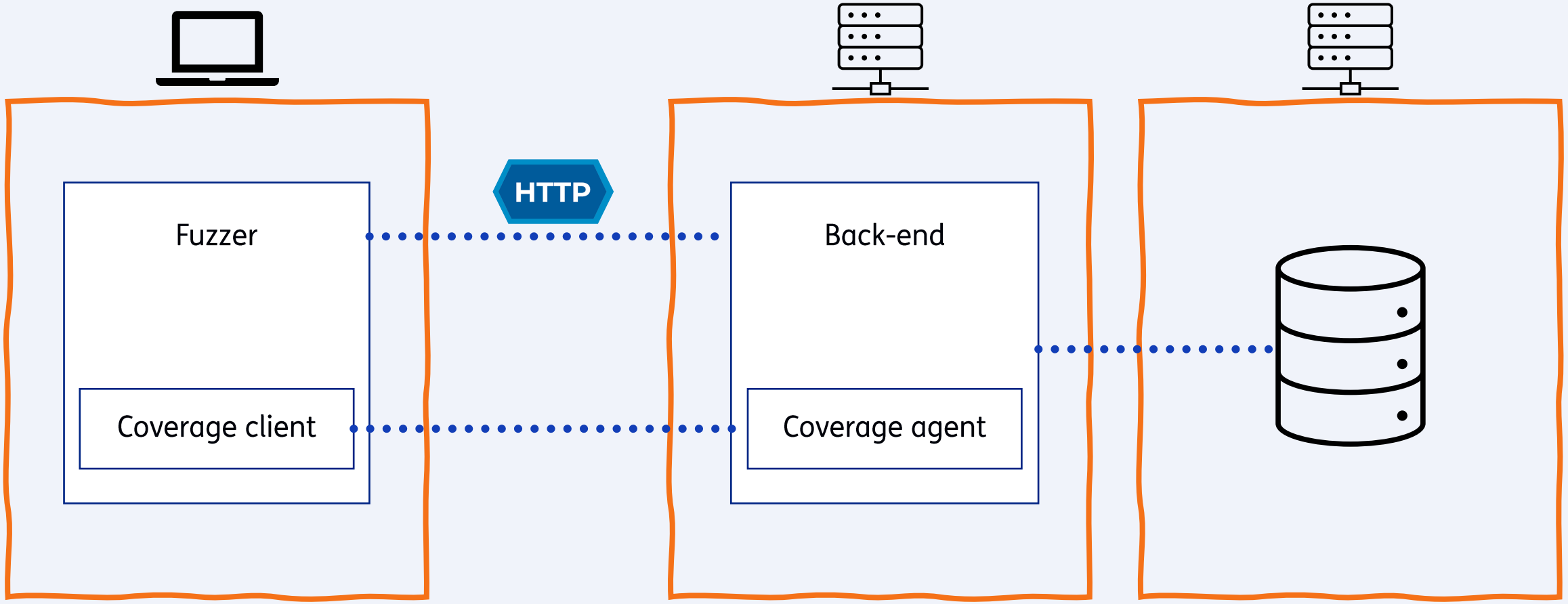
- [x]
- [x]
- [x]

POST `/pet` Add a new pet to the store.

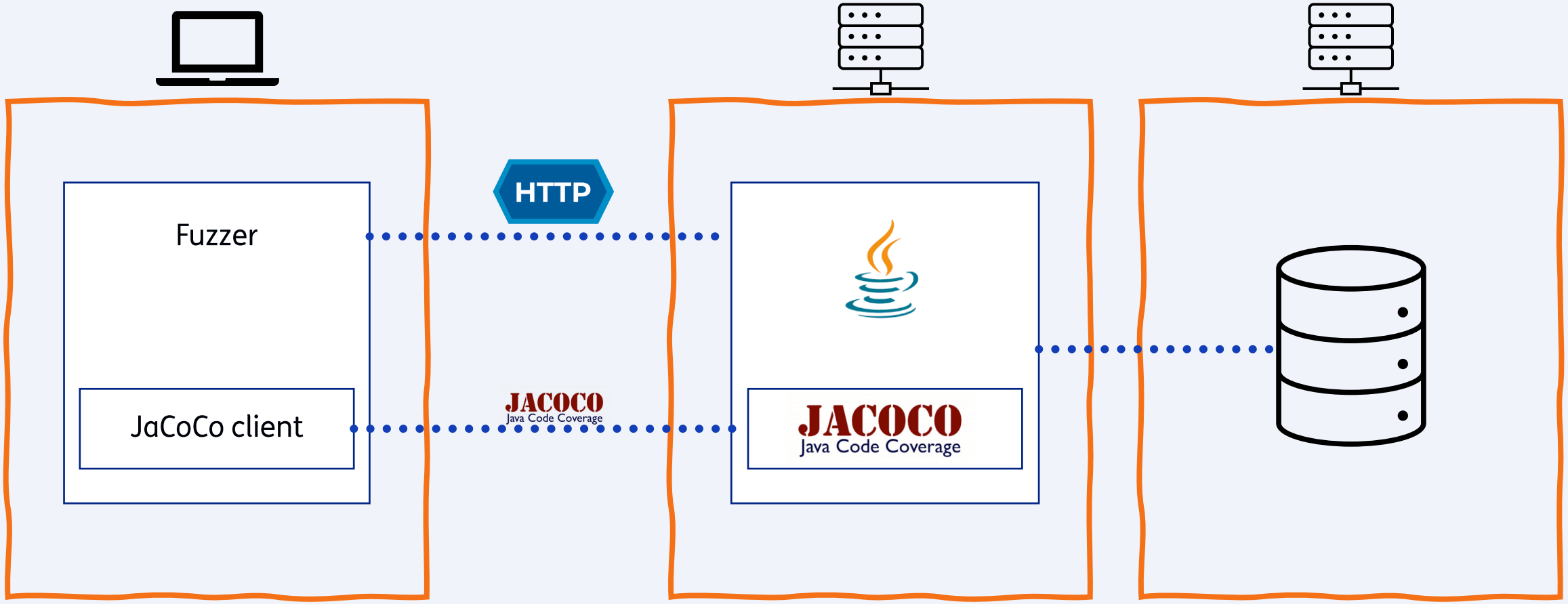
Responses

Code	Description	Links
200	Successful operation	*No links*
400	Invalid input	*No links*
422	Validation exception	*No links*
default	Unexpected error	*No links*

Coverage: for real this time



Coverage: for real this time

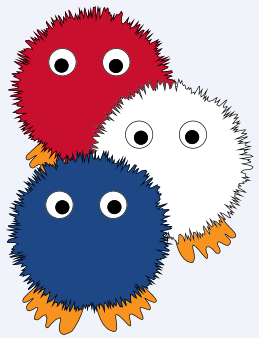


What can WuppieFuzz do?

- Find vulnerable points and issues in your codebase

Symptoms:

- 5xx HTTP status codes (internal server error and friends)
- Deviation from the specification
  - e.g., the response contains "extra" fields
  - e.g., response is an array (all resources) instead of one entry
- Distribution of responses in the dashboard is not as expected
  - e.g., information leak via 401 "unauthorized" / 404 "not found"



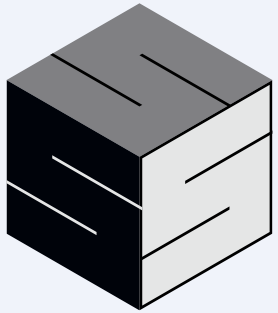
Wuppie Fuzz

open source initiative Approved License

https://tno.to/wuppiefuzz

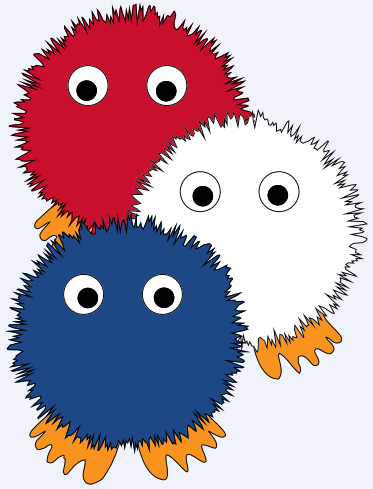
Launch

🚀 WuppieFuzz released 🎉 – September 2024



Coverage-guided REST API fuzzing

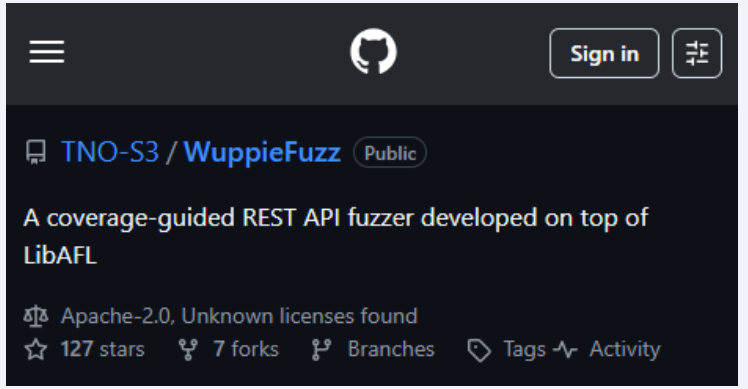
- Test your (public-facing) interfaces at ease
- Supports black box, grey box and white box testing
- Code coverage as well as endpoint coverage



Wuppie Fuzz

Key focus 🔍

- Ease-of-use
- Interpretation of results
- Modularity and extensibility
- Open source



open source initiative
Approved License

https://tno.to/wuppiefuzz