

Web Api Fuzzing

The creation of Wuppiefuzz

Ir. S.H.M. van den Berg |



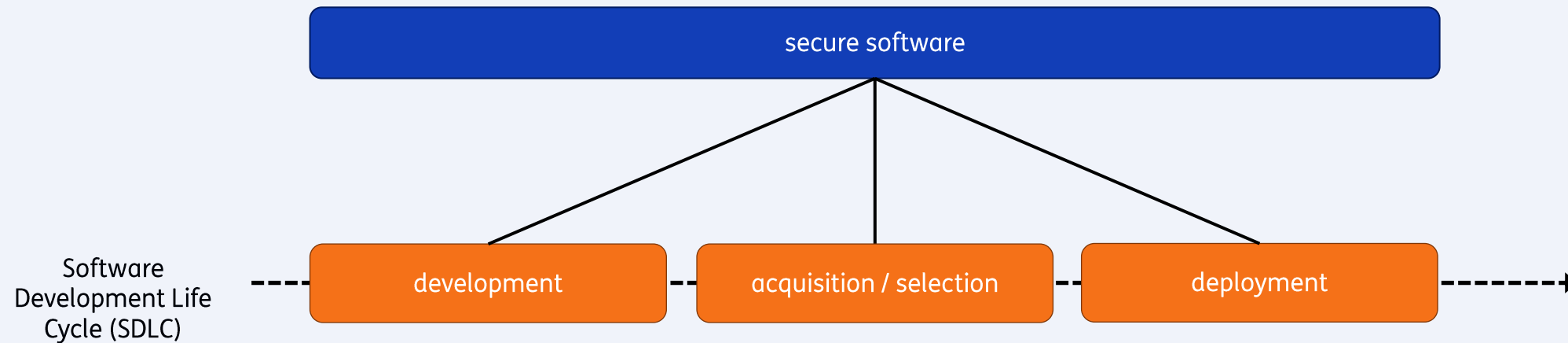
Agenda

1. Why are we doing this
2. Fuzzing
3. Why API-fuzzing
4. How does Wuppiefuzz work
5. Lessons learned

Automated Vulnerability Research

Key strategy

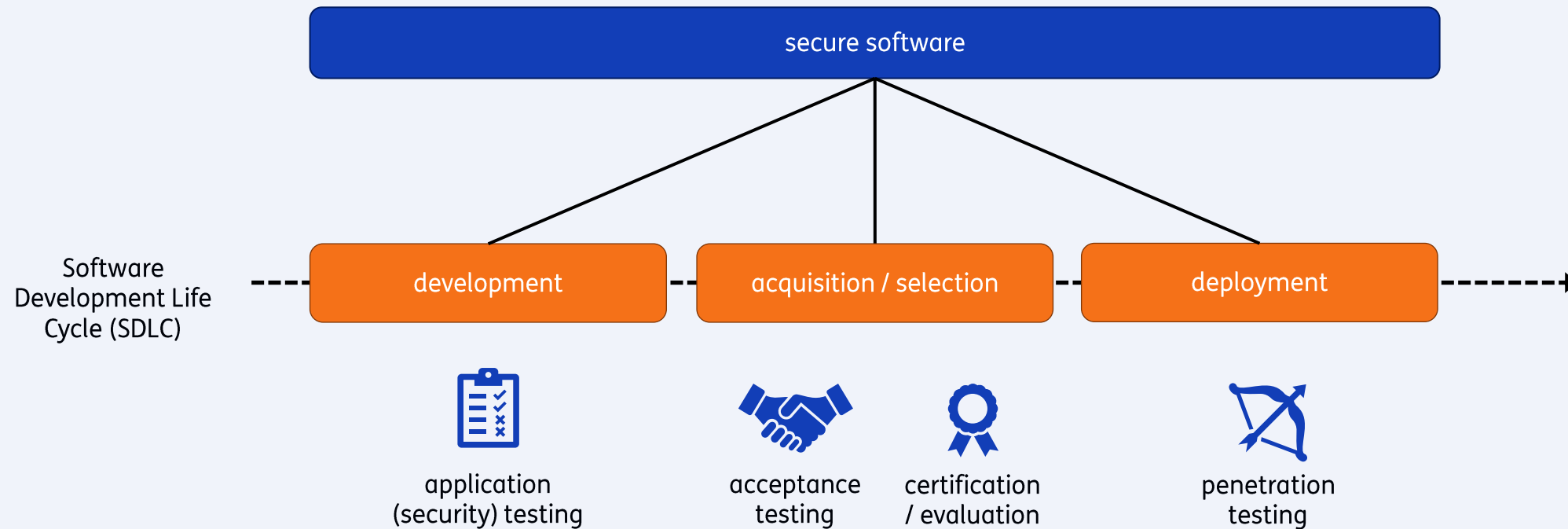
- Apply AVR technology throughout the entire software development life cycle with a strong focus on discovery



Automated Vulnerability Research

Key strategy

- Apply AVR technology throughout the entire software development life cycle with a strong focus on discovery



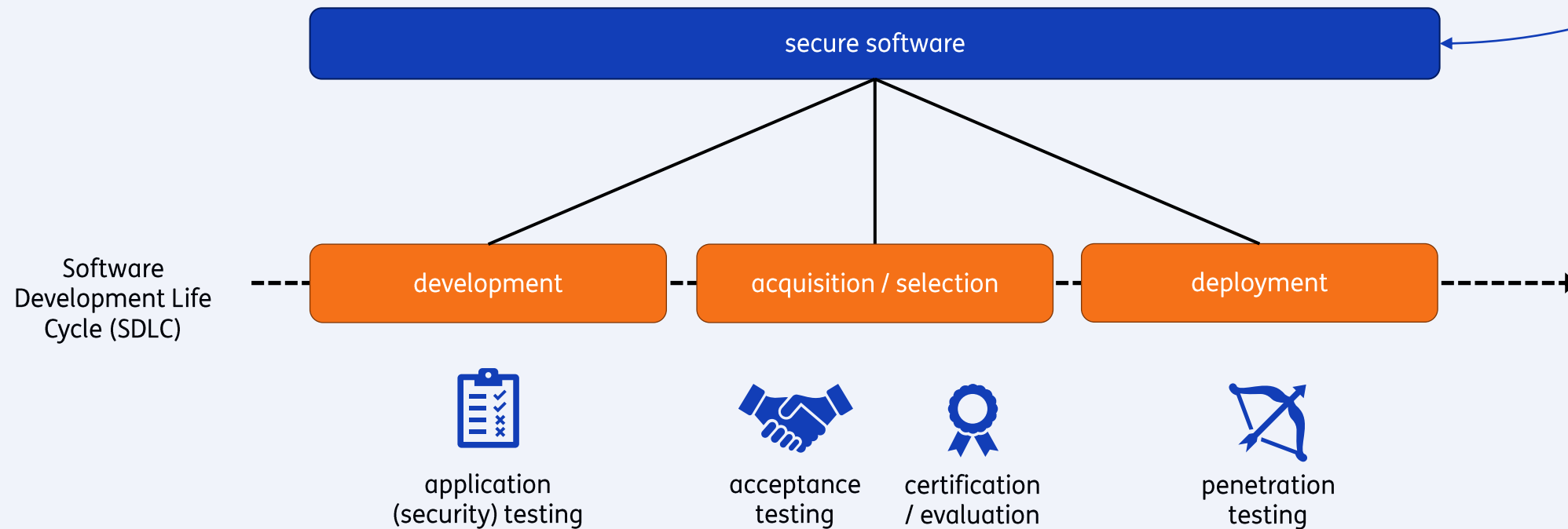
Automated Vulnerability Research



Ultimate goal

Key strategy

- Apply AVR technology throughout the entire software development life cycle with a strong focus on discovery



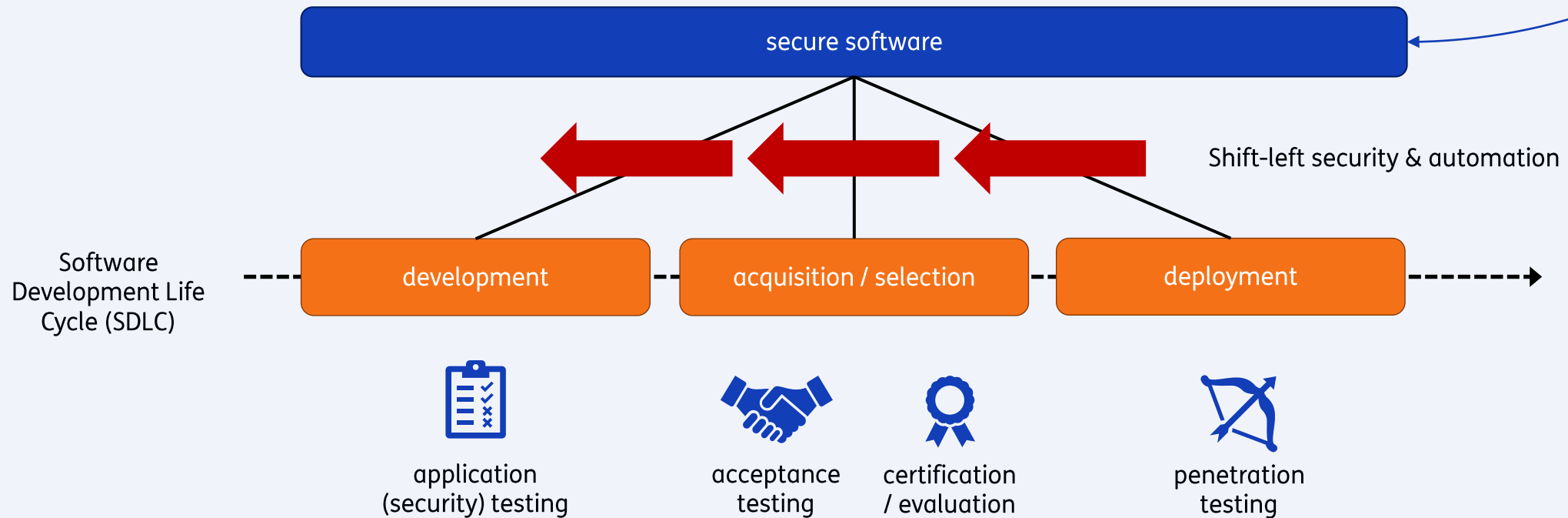
Automated Vulnerability Research



Ultimate goal

Key strategy

- Apply AVR technology throughout the entire software development life cycle with a strong focus on discovery



Fuzzing

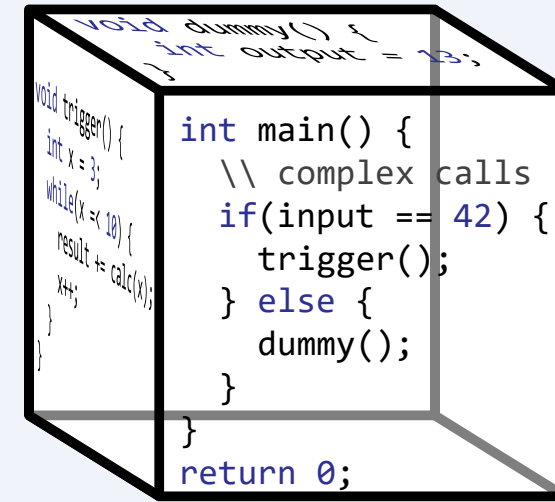
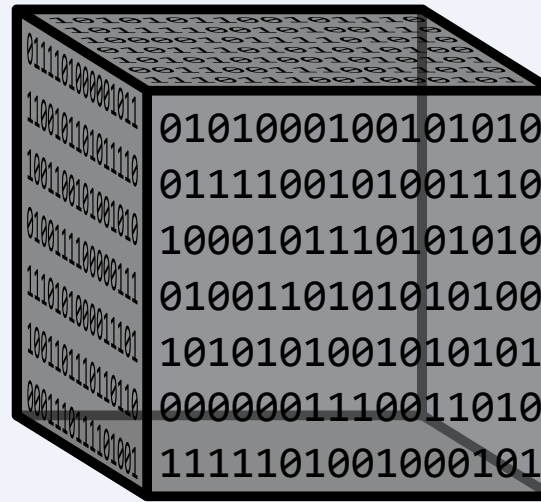
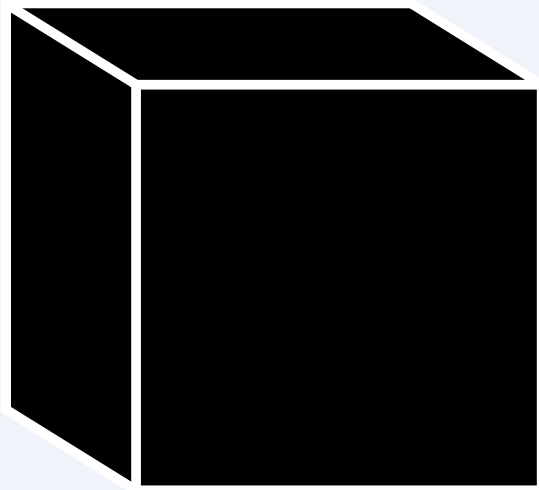
- In Fuzzing, unexpected, faulty and seemingly random inputs are fed to the software under test in an attempt to trigger unexpected, breaking, behaviour.
- Driven By
 - High volume (many different inputs)
 - Testing the unexpected
 - Automation
- Conceptually simple, complex in practice



Why - Fuzzing

- Detecting flaws in software increases
 - Reliability
 - Security
- Fuzzer results are repeatable
 - Simplifies debugging and patching
 - Aids regression testing
- After setup
 - Automation
 - Cheap
 - Fuzzers can run in the background (24/7)

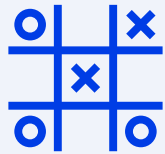
Types of fuzzers



← Less (internal) knowledge on software required

→ More information available for fuzzer guidance

Fuzzing research interests



Stateful
Fuzzing



API
Fuzzing



Protocol
Fuzzing



Directed
Fuzzing



Integration
in
CI/CD



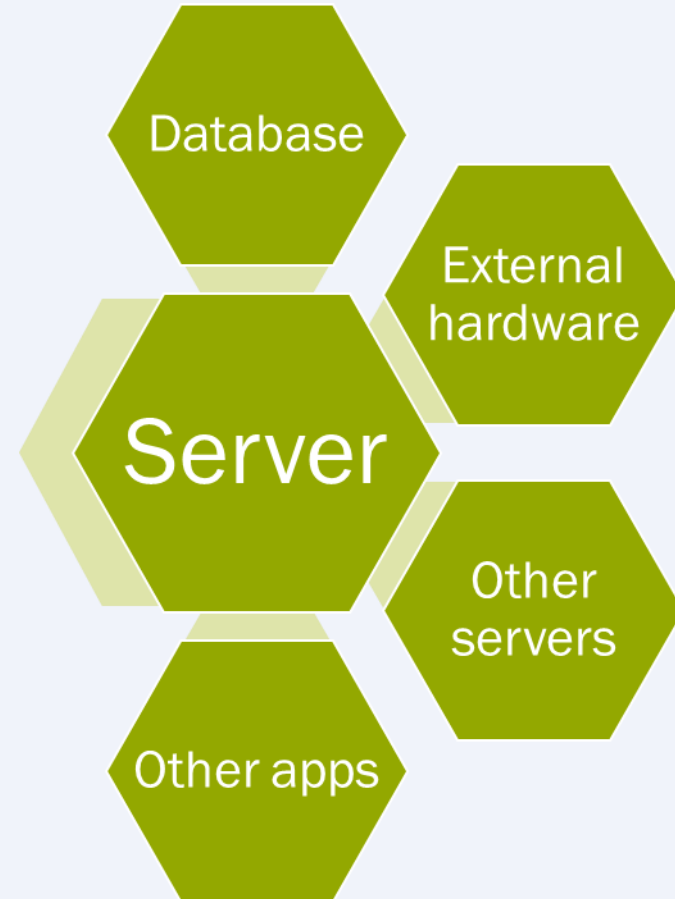
OWASP
vulnerability
detection



Corpus &
Harness
Generation

Why API fuzzing

- APIs are everywhere
 - Modern systems consist of many parts
 - Communication between systems through APIs
- APIs are more complex
 - 'Random data fuzzing' will create invalid commands
 - You are only testing the interpreter

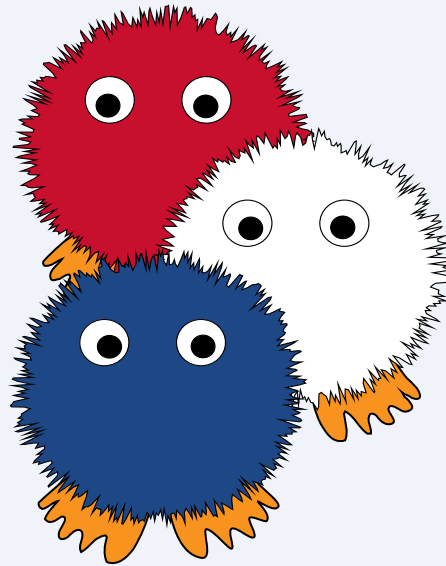


State of the art

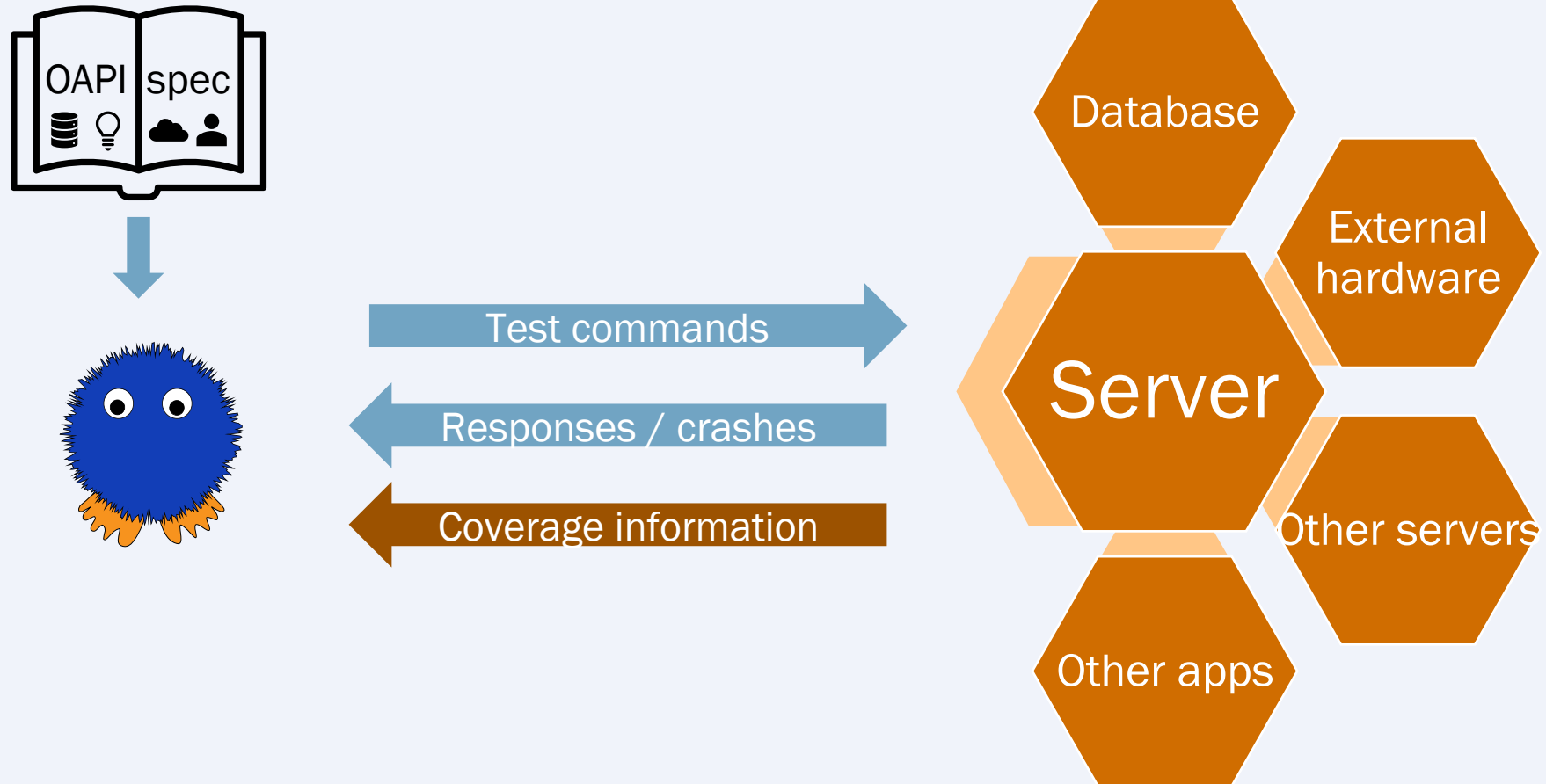
- Existing coverage guided fuzzers:
 - AFL, AFL++
 - Libfuzzer
 - Jazzer
- REST-API fuzzer
 - Restler
- No combination yet...

Introducing Wuppiefuzz

- Written in Rust using libafl
- Coverage guided REST-API fuzzing



How does Wuppiefuzz work



RTFM: Reading the f..... manual

RTFM: Reading the fuzzing manual

The image shows the Swagger Editor interface. On the left, the Swagger JSON definition is displayed in a code editor. On the right, the rendered API documentation is shown, including the title 'Swagger Petstore 1.0.0', a description, a list of tags, and a list of API endpoints with their methods and descriptions.

```
1 swagger: "2.0"
2 info:
3   description: "This is a sample server Petstore server. You can find out
4     more about Swagger at [http://swagger.io](http://swagger.io) or on
5     [irc.freenode.net, #swagger](http://swagger.io/irc/). For this sample
6     , you can use the api key `special-key` to test the authorization
7     filters."
8   version: "1.0.0"
9   title: "Swagger Petstore"
10  termsOfService: "http://swagger.io/terms/"
11  contact:
12    email: "apiteam@swagger.io"
13  license:
14    name: "Apache 2.0"
15    url: "http://www.apache.org/licenses/LICENSE-2.0.html"
16 host: "petstore.swagger.io"
17 basePath: "/v2"
18 tags:
19 - name: "pet"
20   description: "Everything about your Pets"
21   externalDocs:
22     description: "Find out more"
23     url: "http://swagger.io"
24 - name: "store"
25   description: "Access to Petstore orders"
26 - name: "user"
27   description: "Operations about user"
28   externalDocs:
29     description: "Find out more about our store"
30     url: "http://swagger.io"
31 schemes:
32 - "https"
33 - "http"
34 paths:
35 /pet:
36   post:
37     tags:
38     - "pet"
39     summary: "Add a new pet to the store"
40     description: ""
41     operationId: "addPet"
42     consumes:
43     - "application/json"
44     - "application/xml"
```

Swagger Petstore 1.0.0

[Base URL: petstore.swagger.io/v2]

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on irc.freenode.net, #swagger. For this sample, you can use the api key `special-key` to test the authorization filters.

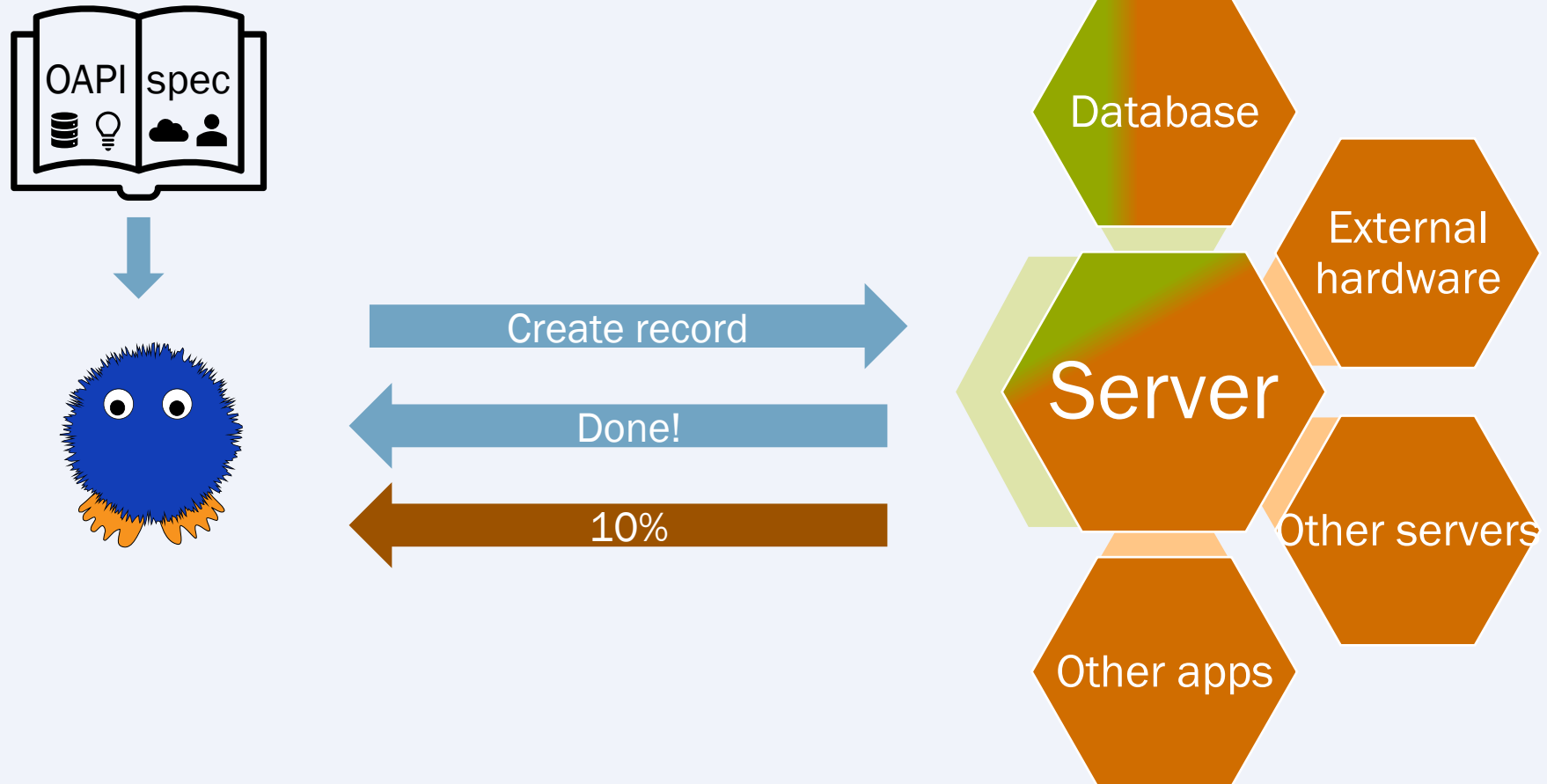
[Terms of service](#)
[Contact the developer](#)
[Apache 2.0](#)
[Find out more about Swagger](#)

Schemes: HTTPS Authorize

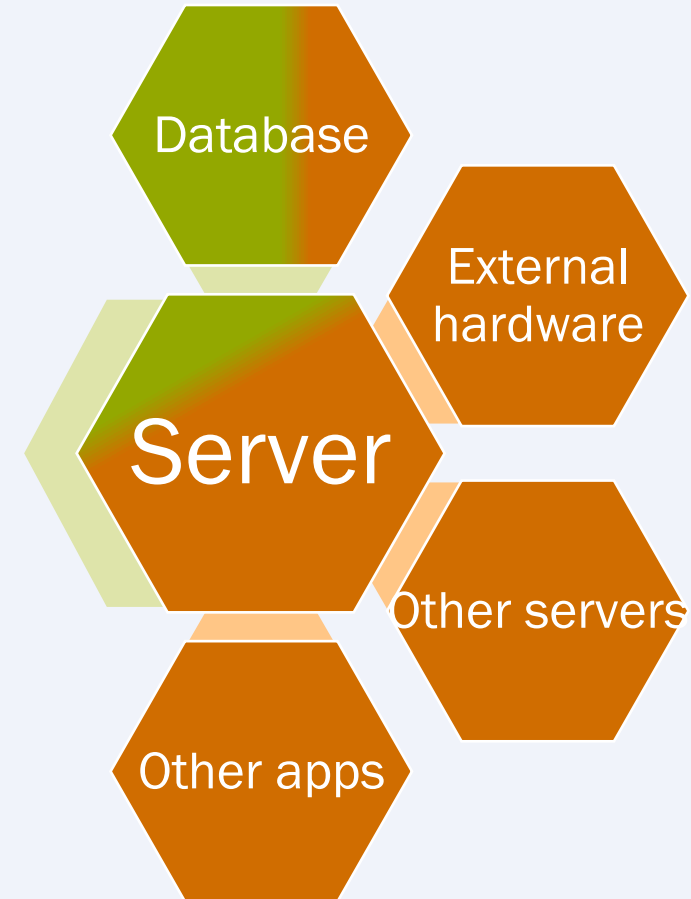
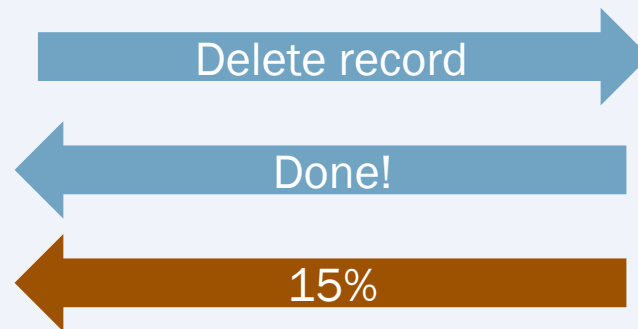
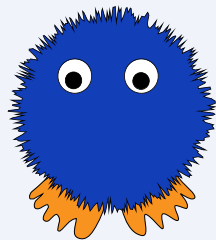
pet Everything about your Pets Find out more: <http://swagger.io>

- POST** /pet Add a new pet to the store
- PUT** /pet Update an existing pet
- GET** /pet/findByStatus Finds Pets by status

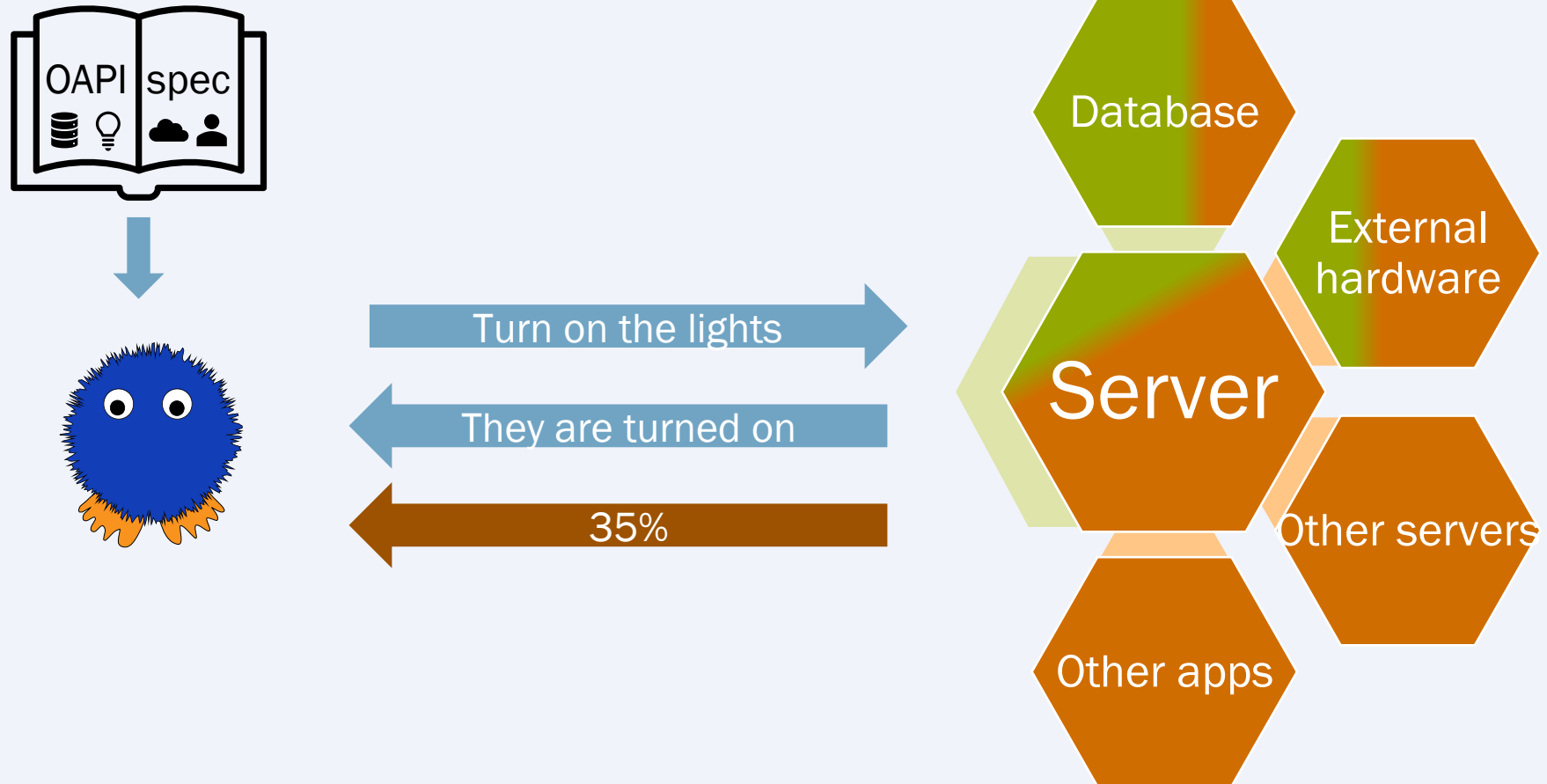
How does Wuppiefuzz work



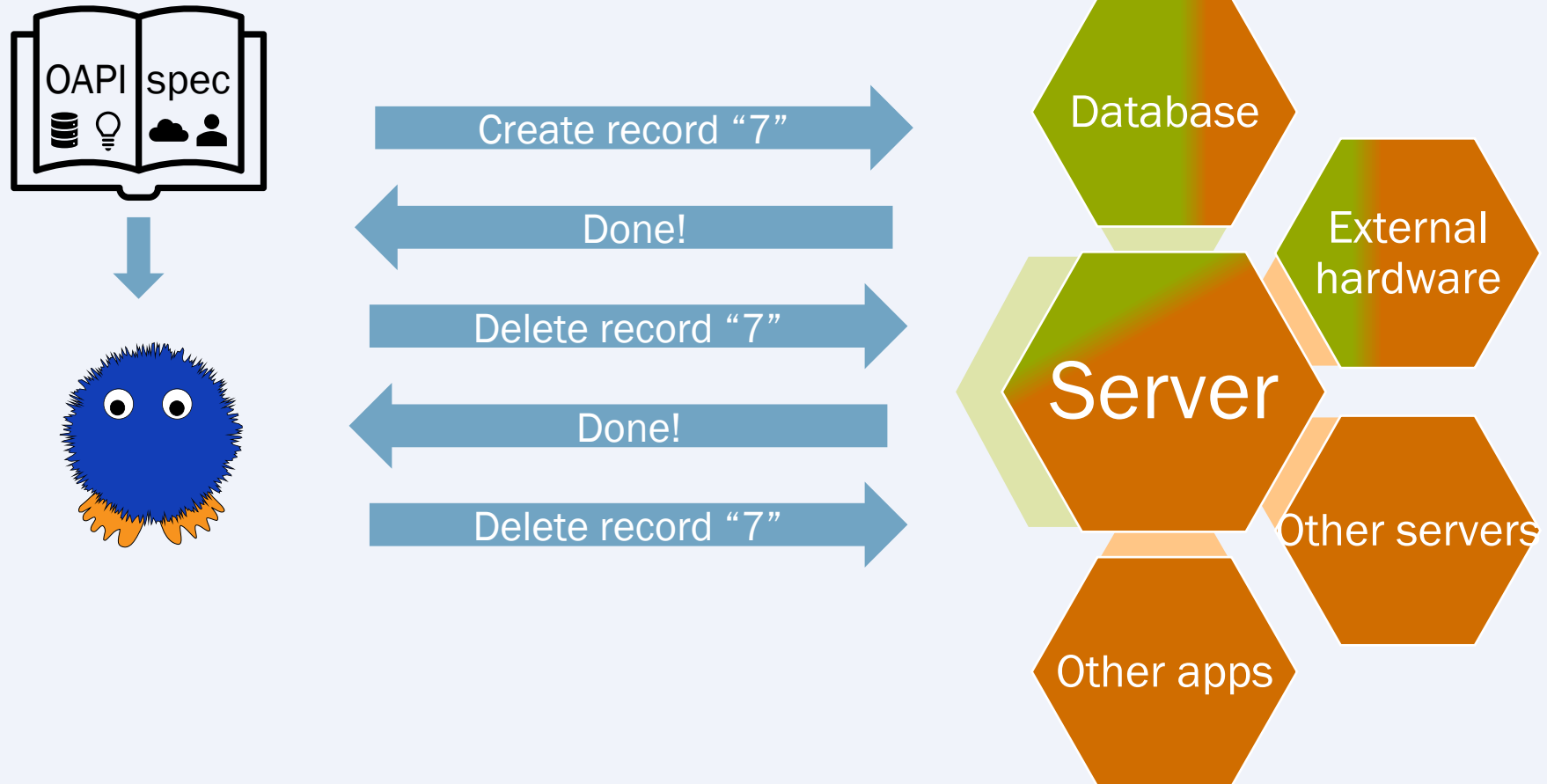
How does Wuppiefuzz work



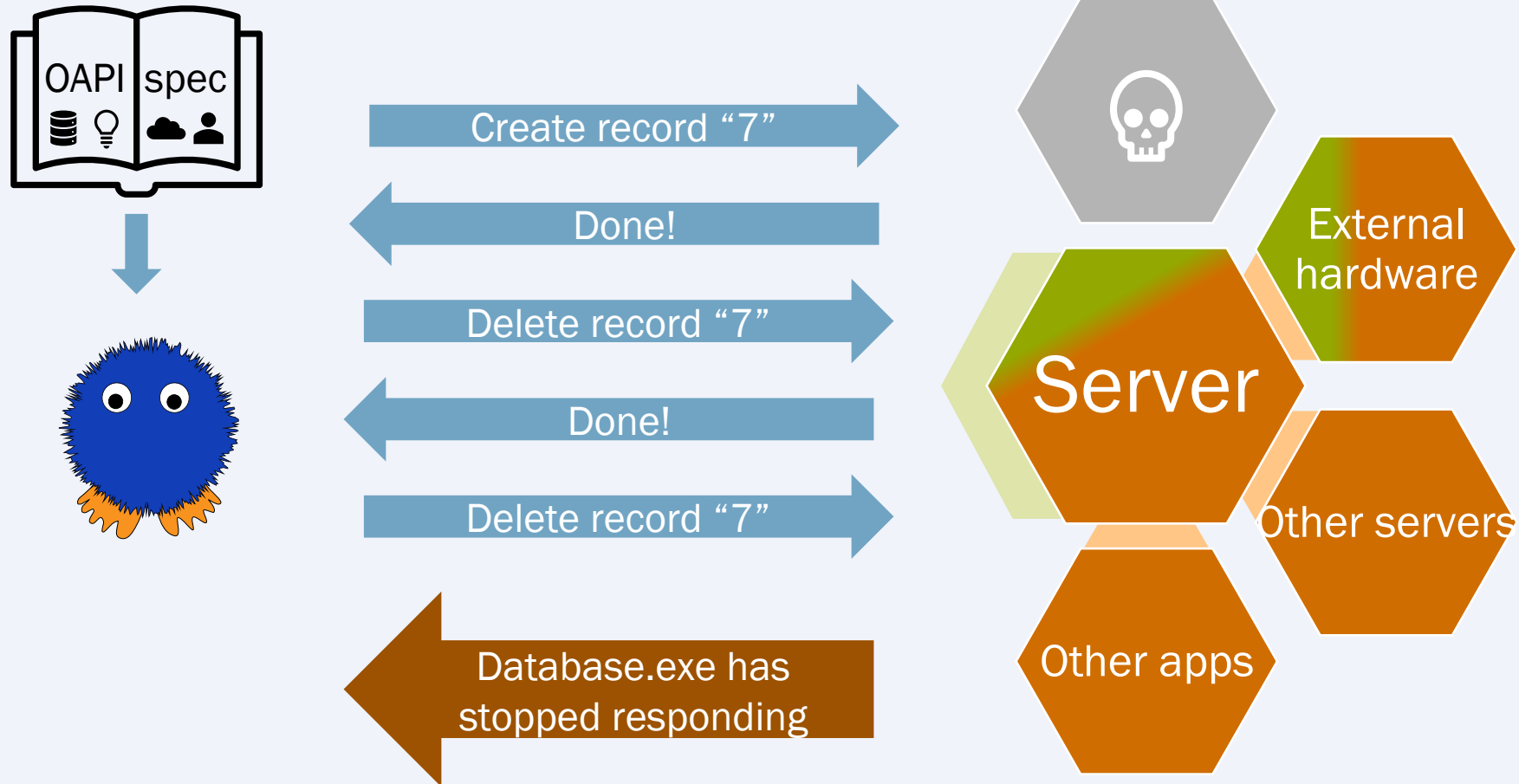
How does Wuppiefuzz work



How does Wuppiefuzz work



How does Wuppiefuzz work



Lessons learned

- Automatically creating dependencies is very difficult
 - Naming can be very different even though the same is meant
- Resetting the target API takes a lot of time
 - With fuzzing the target state is usually reset
 - Without reset you accumulate state
 - Harder to create reproduction steps
- Start on time with validation tooling



TNO innovation
for life