Web Api Fuzzing

The creation of Wuppiefuzz

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

TNO innovation for life

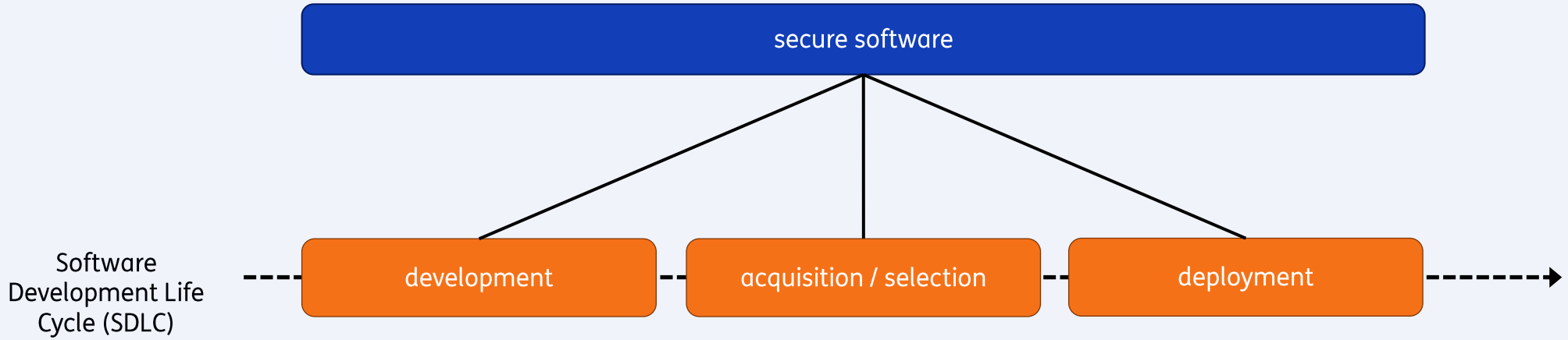
Agenda

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

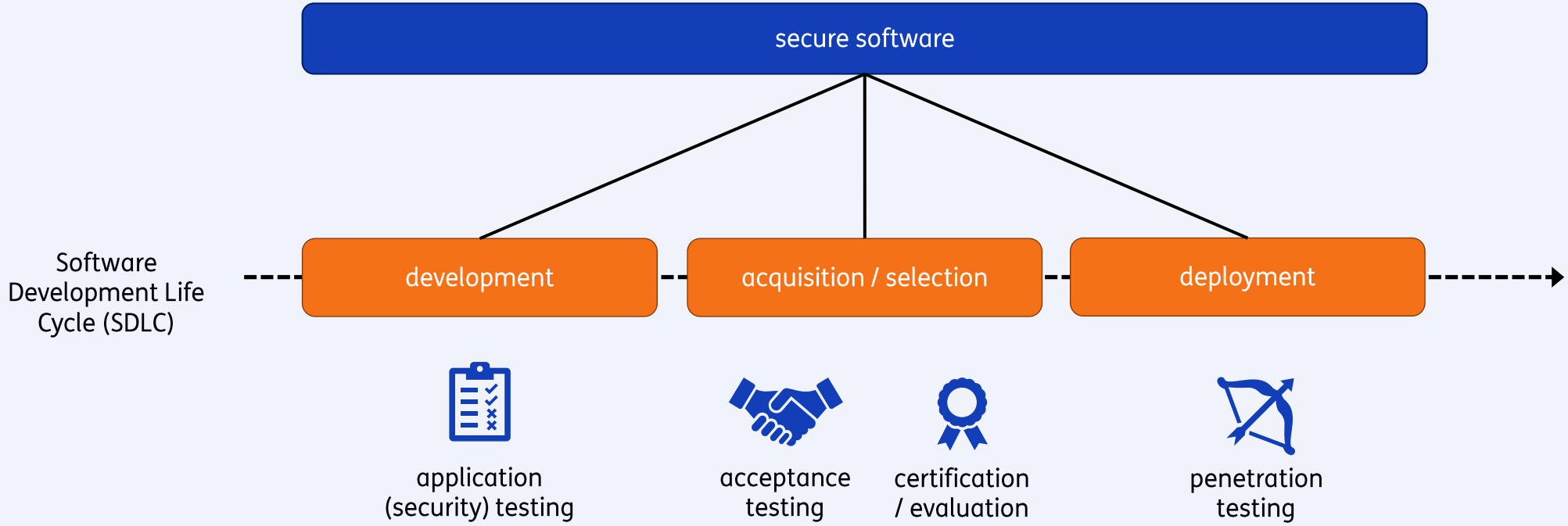
- Apply AVR technology throughout the entire software development life cycle with a strong focus on discovery



Automated Vulnerability Research

Key strategy

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

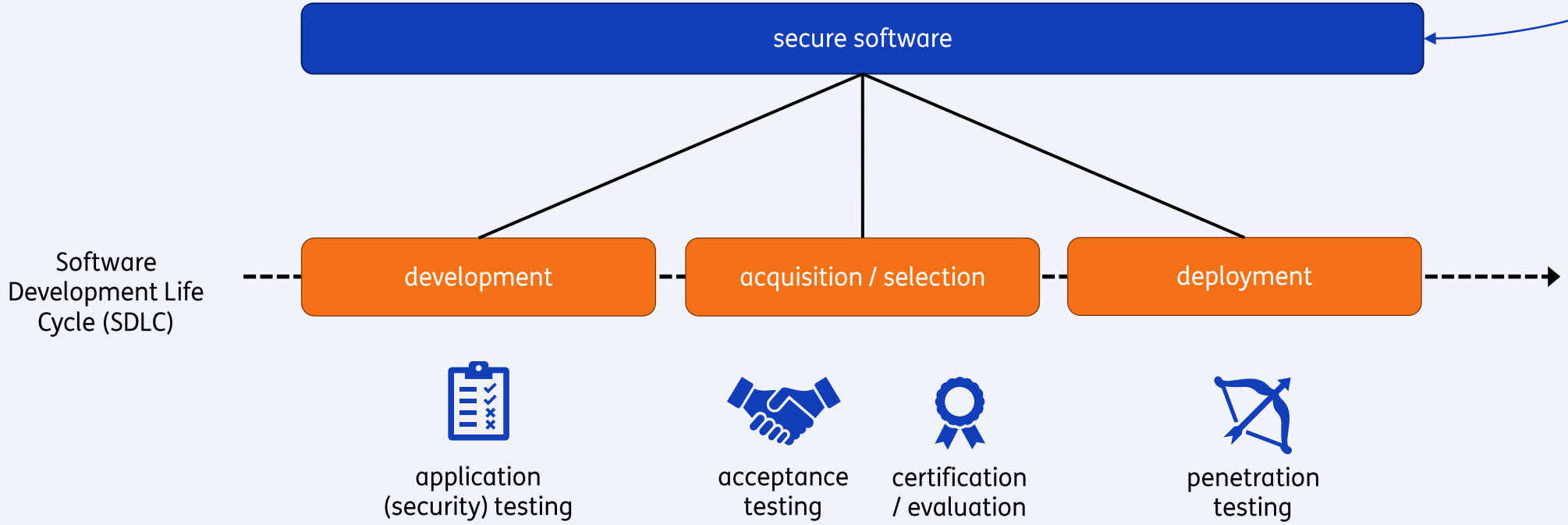


Automated Vulnerability Research

Ultimate goal

Key strategy

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

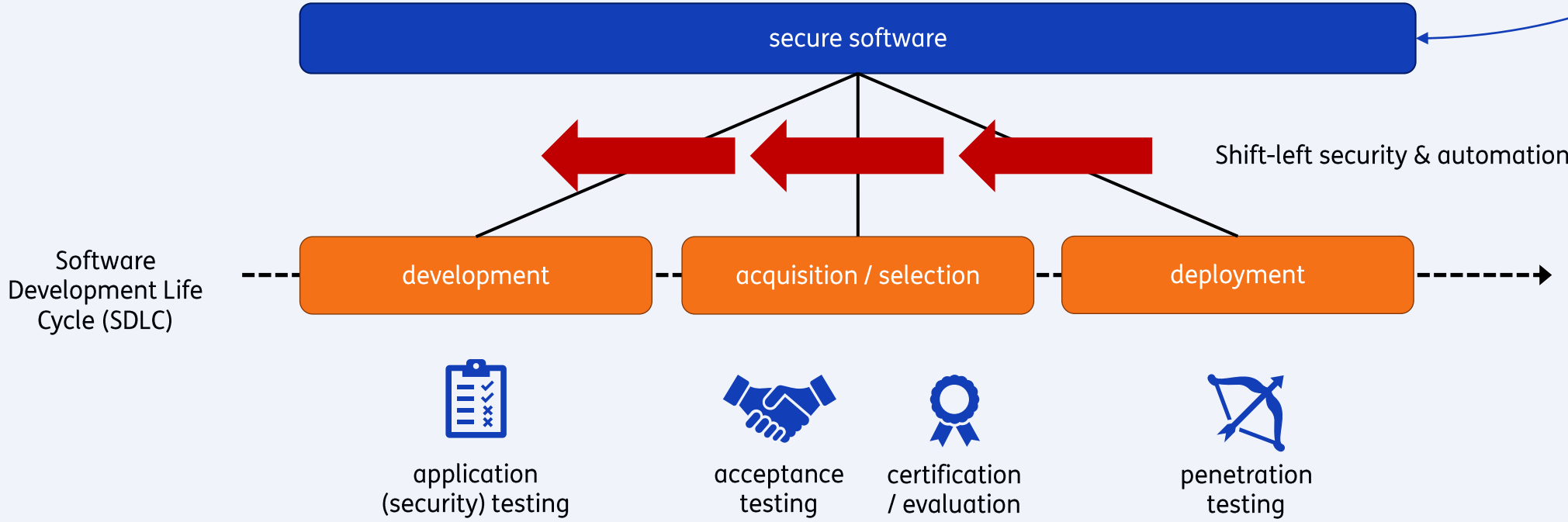


Automated Vulnerability Research

Ultimate goal

Key strategy

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

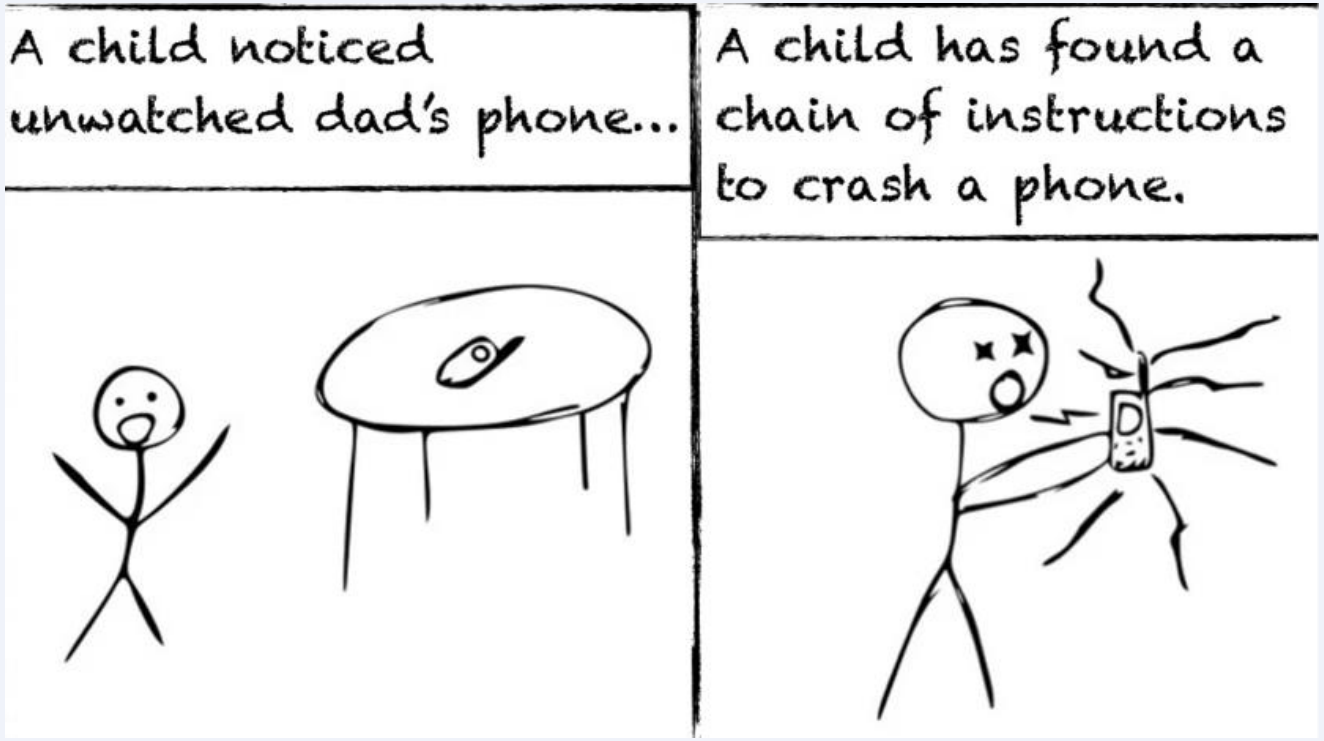


Fuzzing

- 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



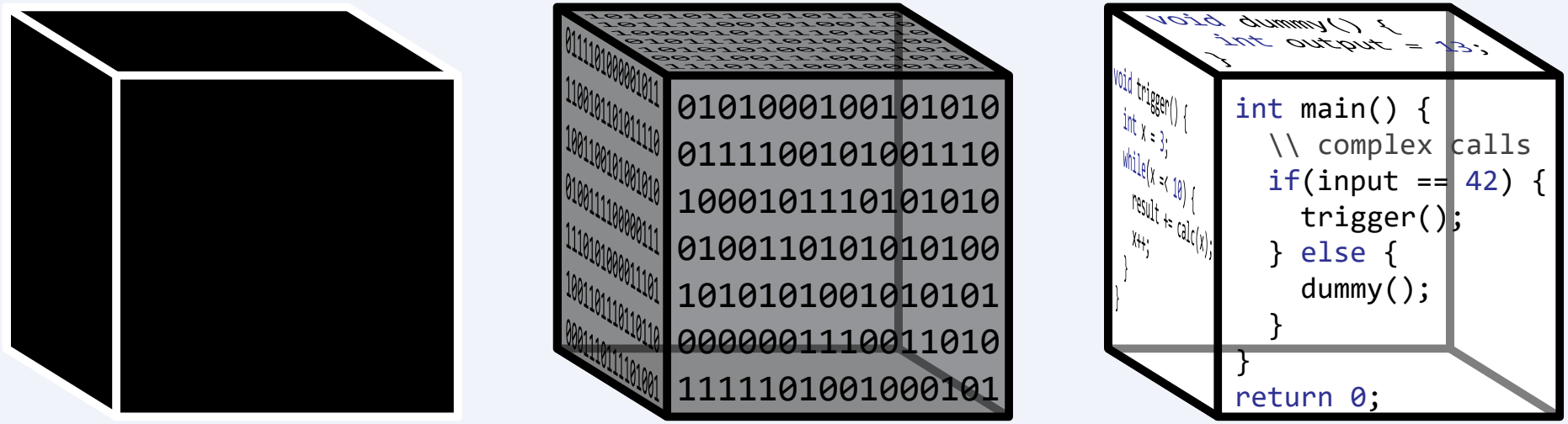
Why - Fuzzing

- 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

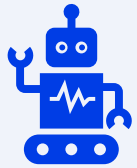
API Fuzzing

Protocol Fuzzing

Directed Fuzzing

Integration in CI/CD

OWASP vulnerability detection

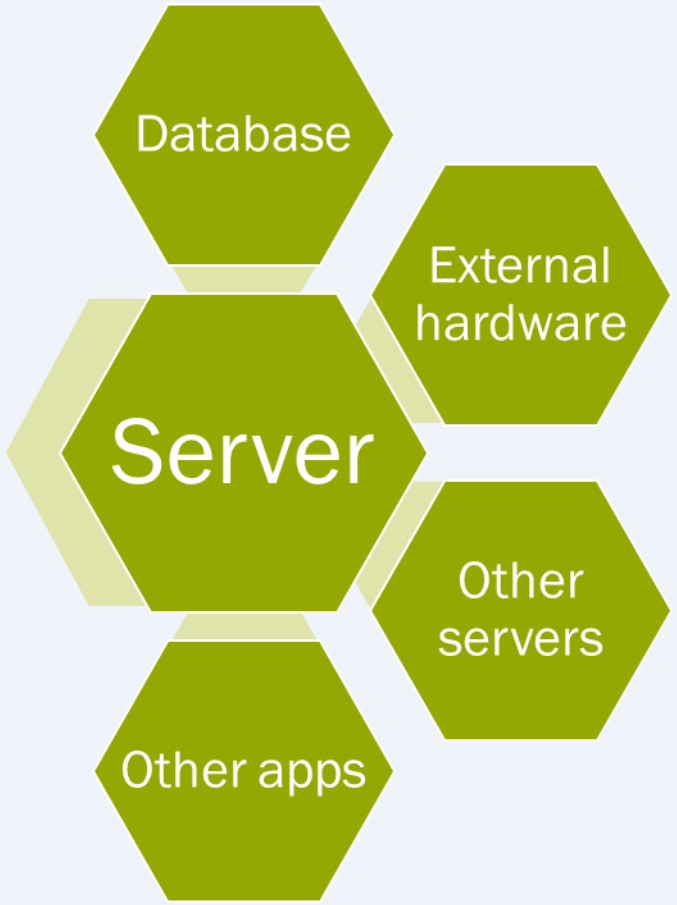


Corpus & Harness Generation

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



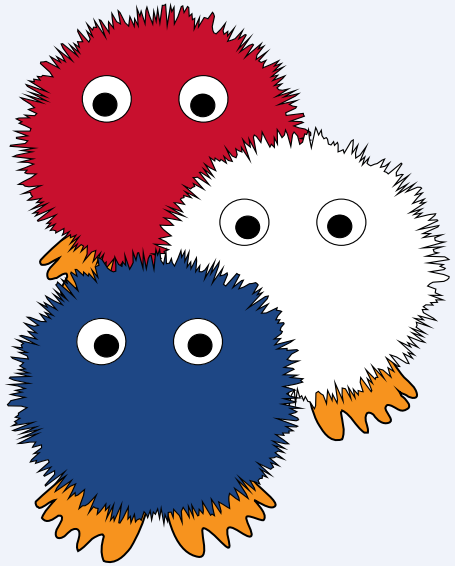
State of the art

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

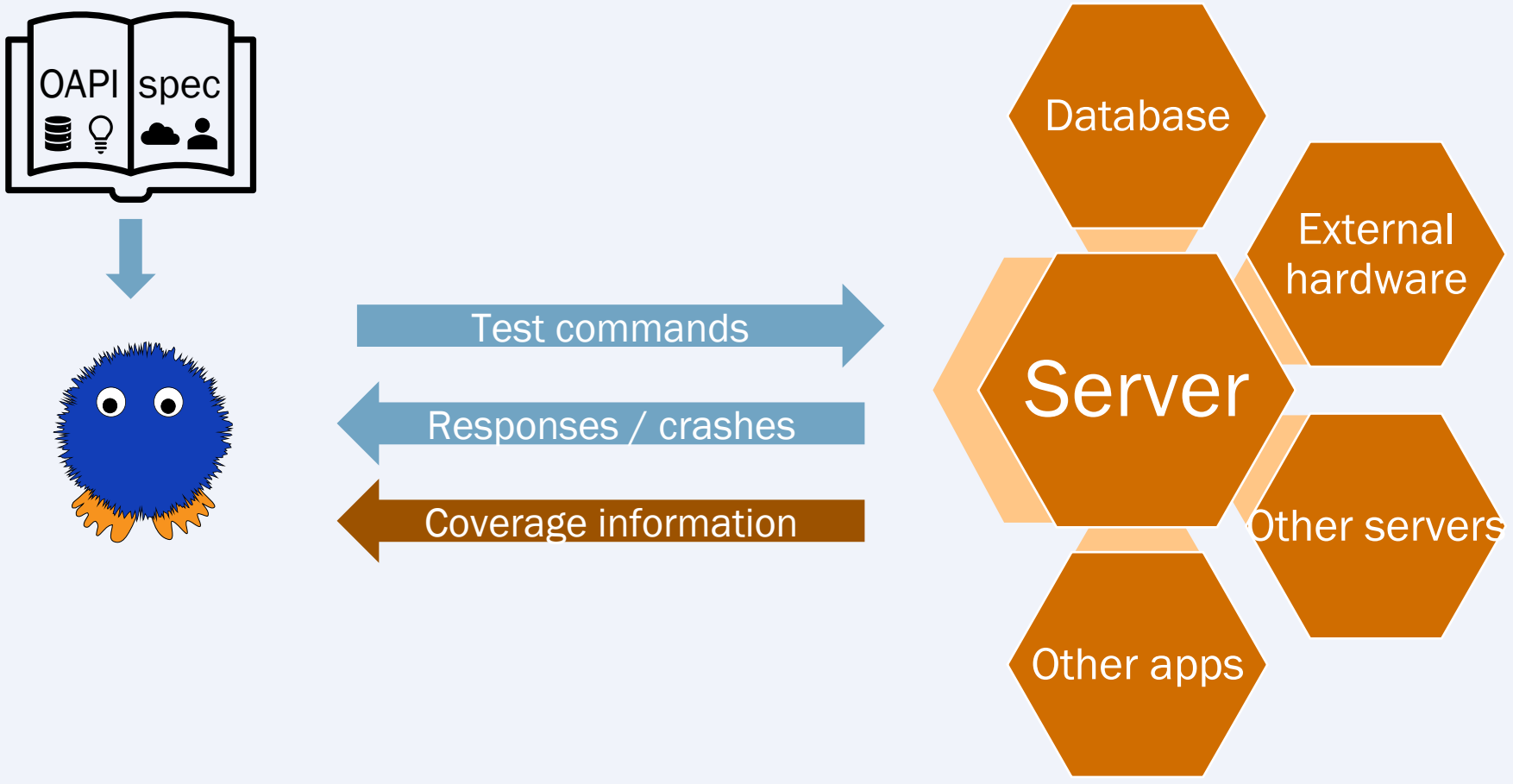
Introducing Wuppiefuzz

- Written in Rust using libafl

- Coverage guided REST-API fuzzing

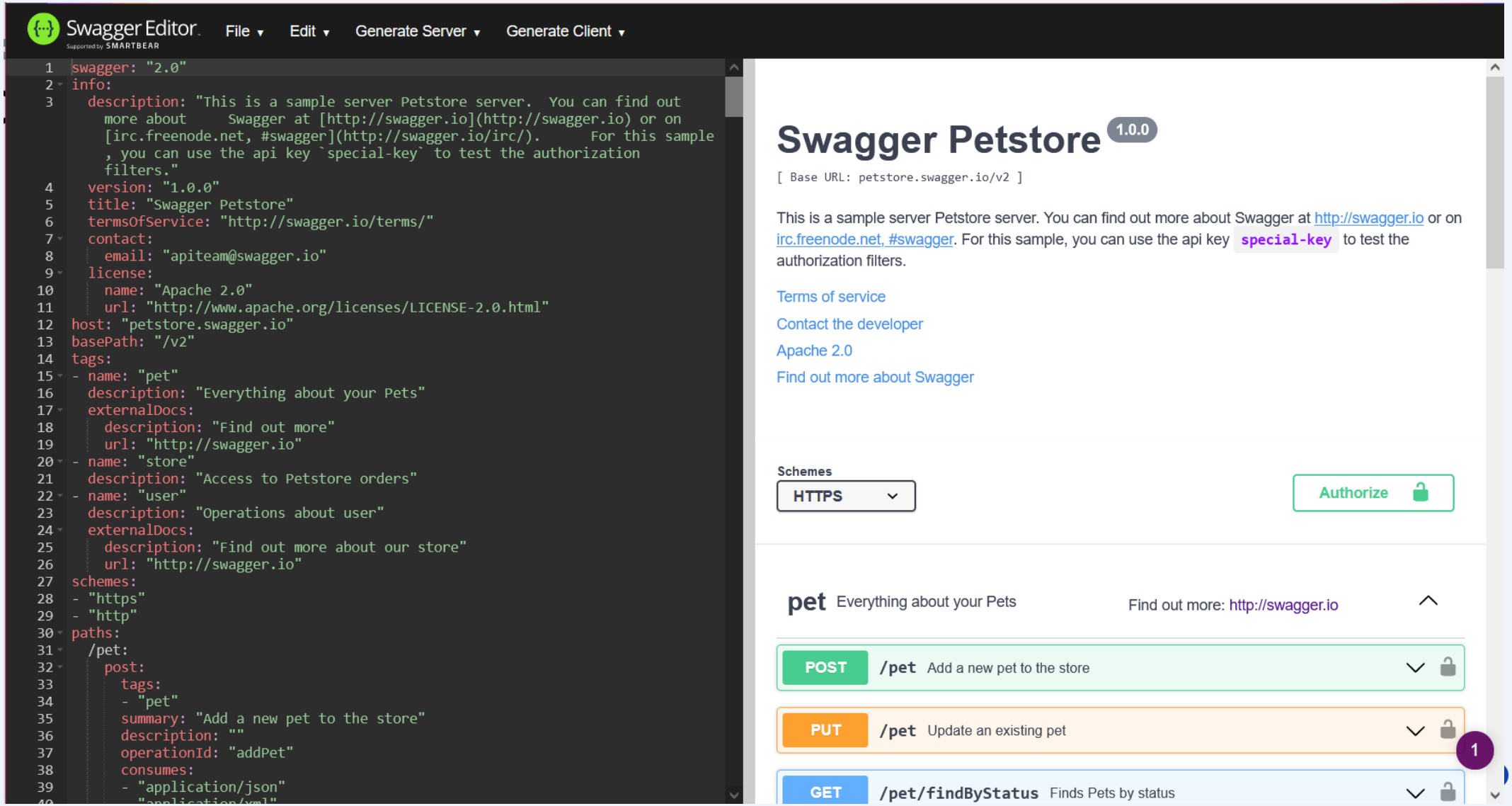


How does Wuppiefuzz work

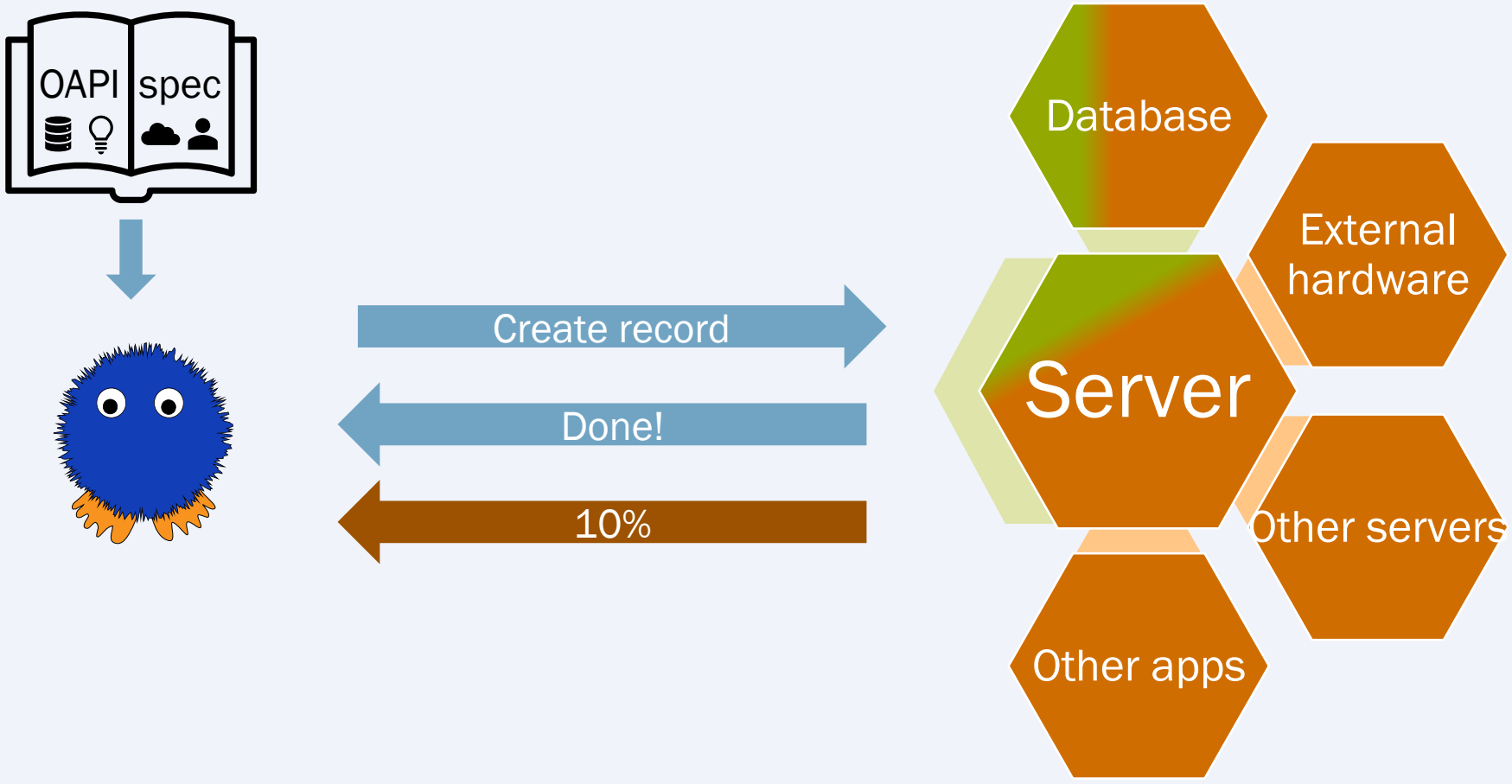


RTFM: Reading the f..... manual

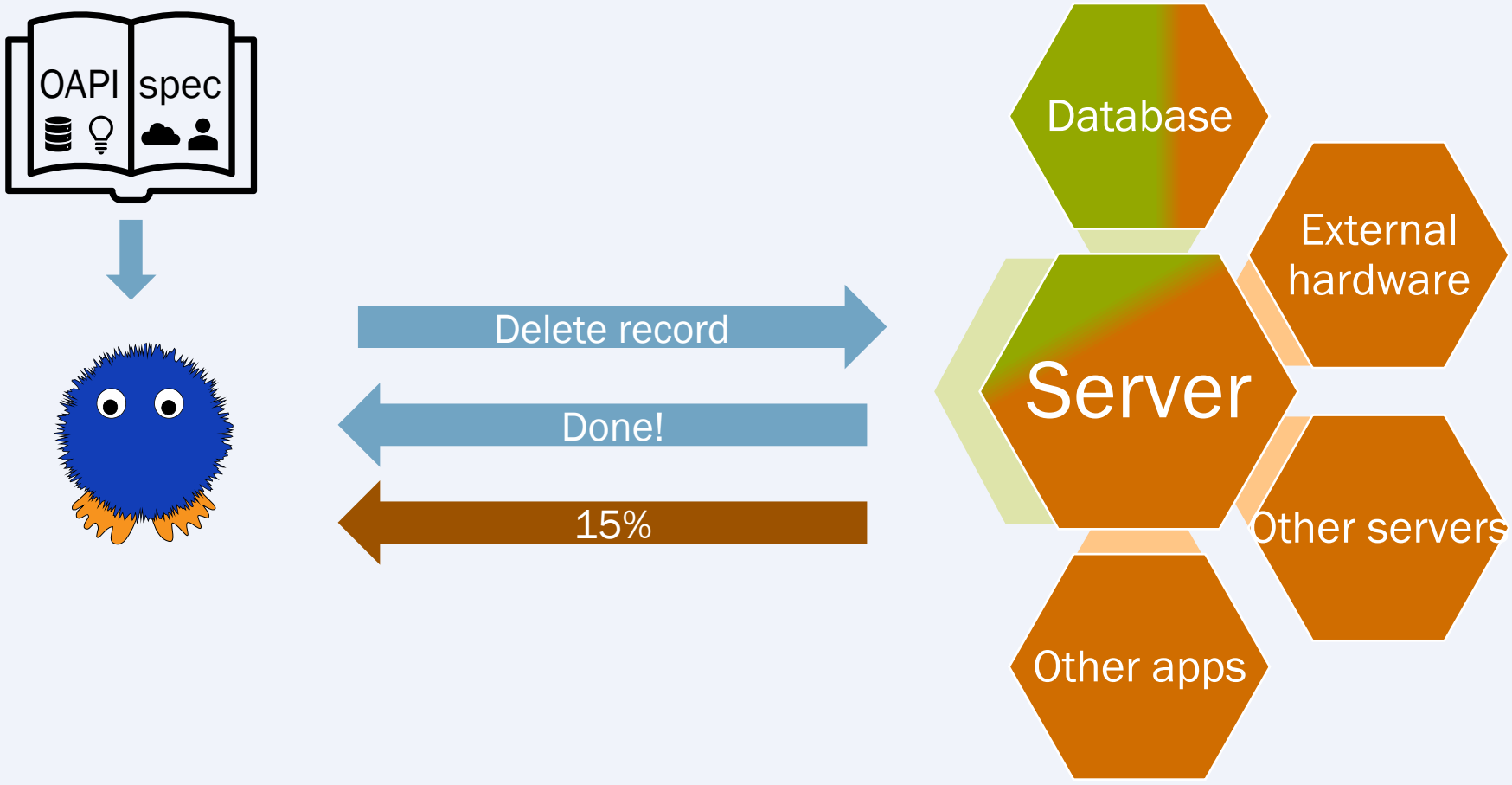
RTFM: Reading the fuzzing manual



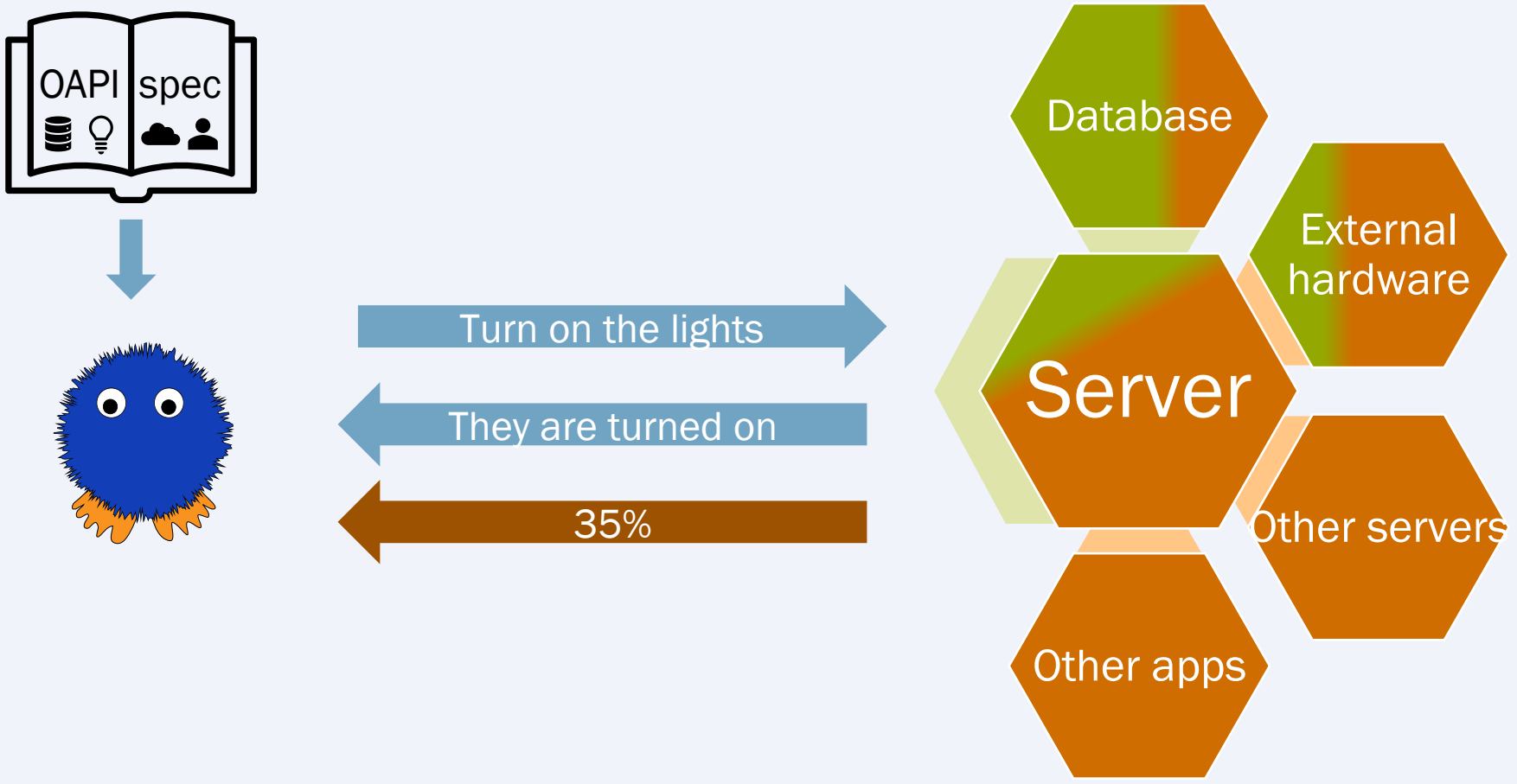
How does Wuppiefuzz work



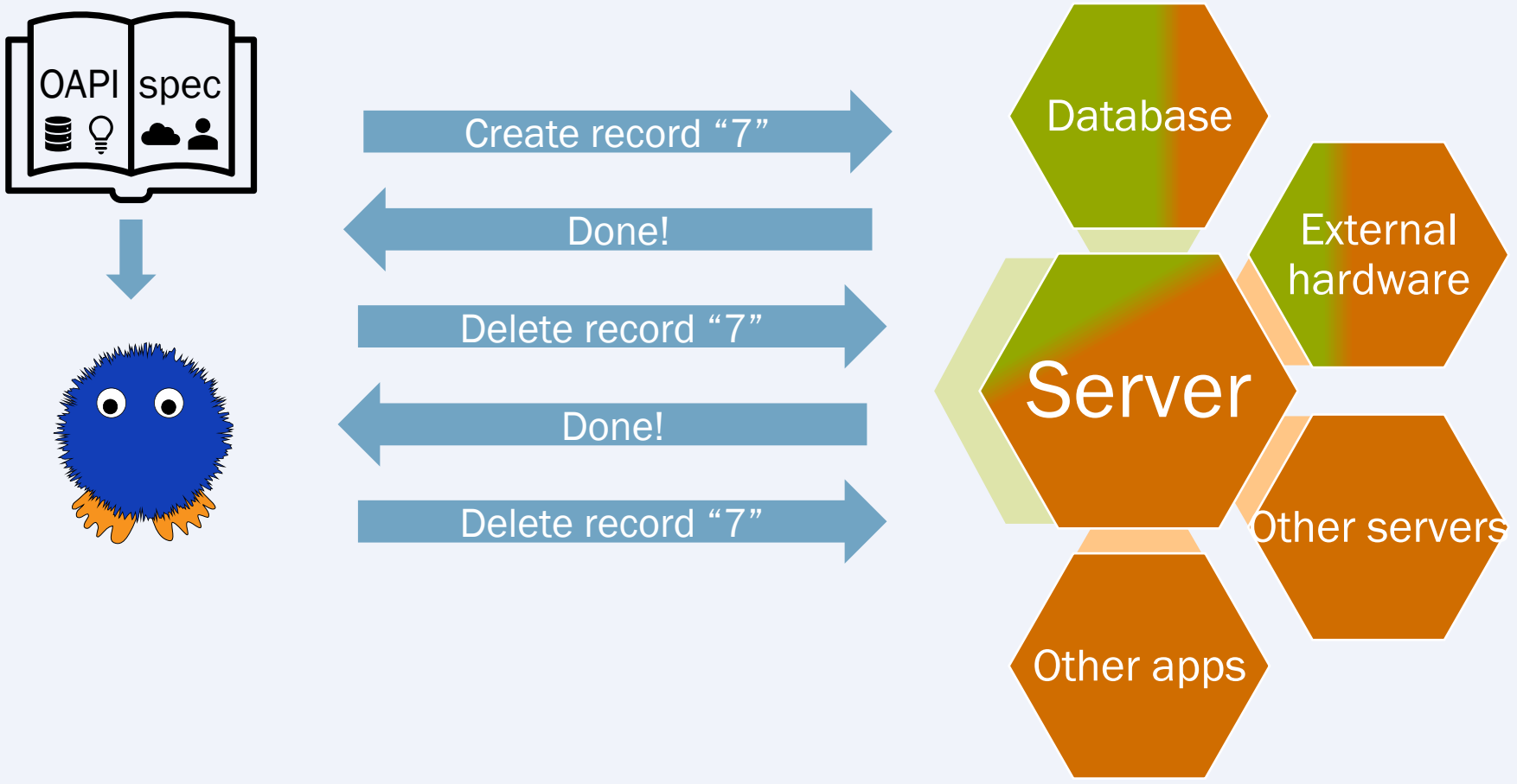
How does Wuppiefuzz work



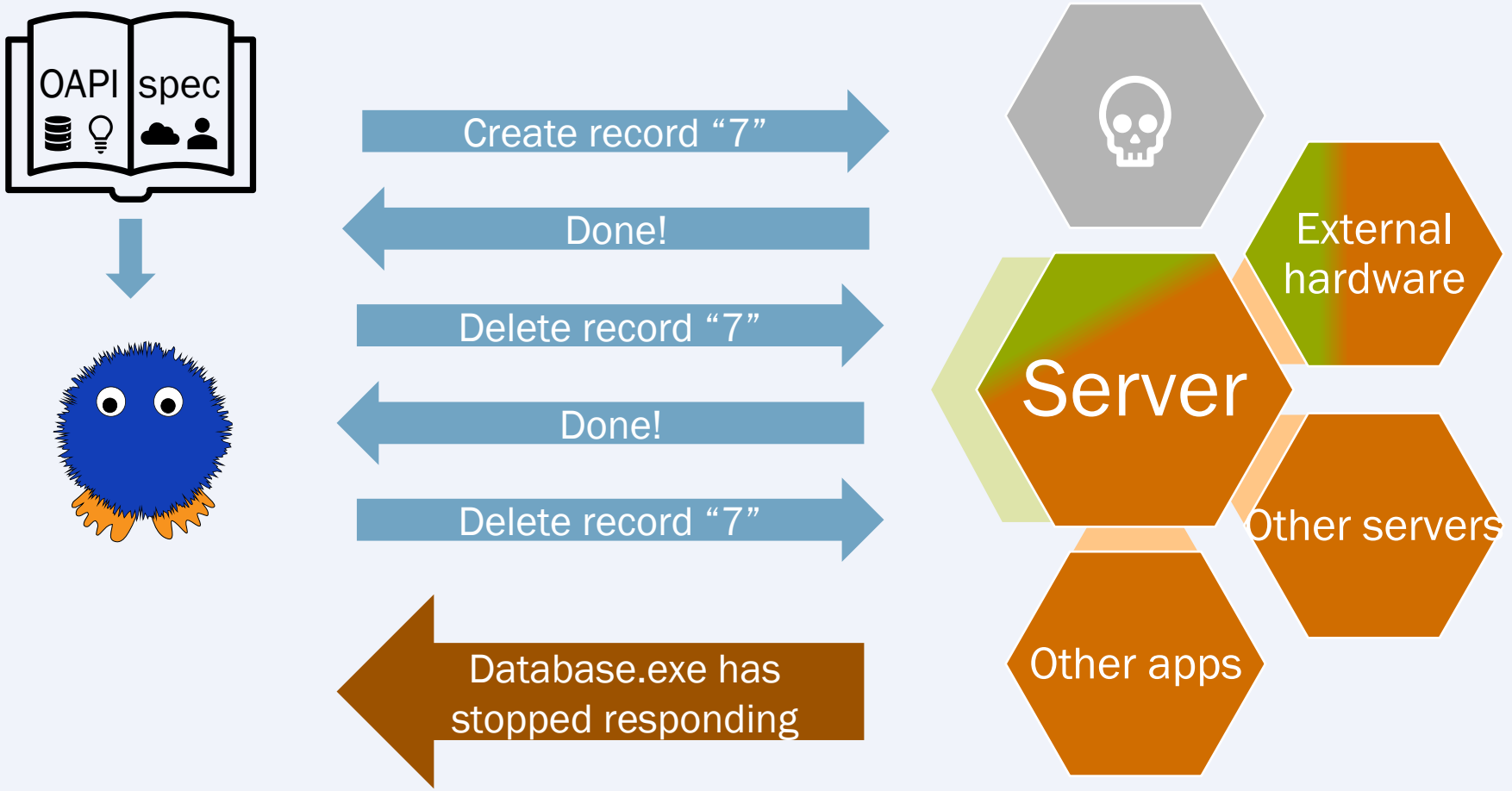
How does Wuppiefuzz work



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