KNU KYUNGPOOK NATIONAL UNIVERSITY

Fuzzing in serverless architecture

2021327158

Manish Pandey

20 June 2023

Motivation

- Serverless computing is becoming increasingly popular for building and deploying an application, however, it also introduces new security risks and vulnerabilities.
- The OWASP Serverless Top 10 list identifies common security risks in a serverless architecture.
- A fuzzing architecture can provide a comprehensive approach to testing and securing serverless functions.
- Goal: To design Fuzzer that can help prioritize security and ensure reliability in a serverless framework.

How to run the framework

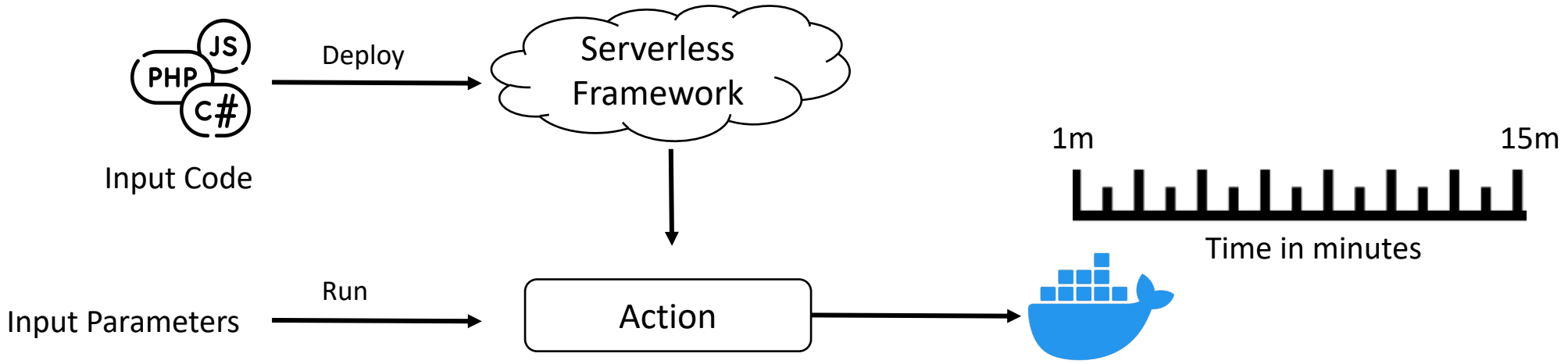


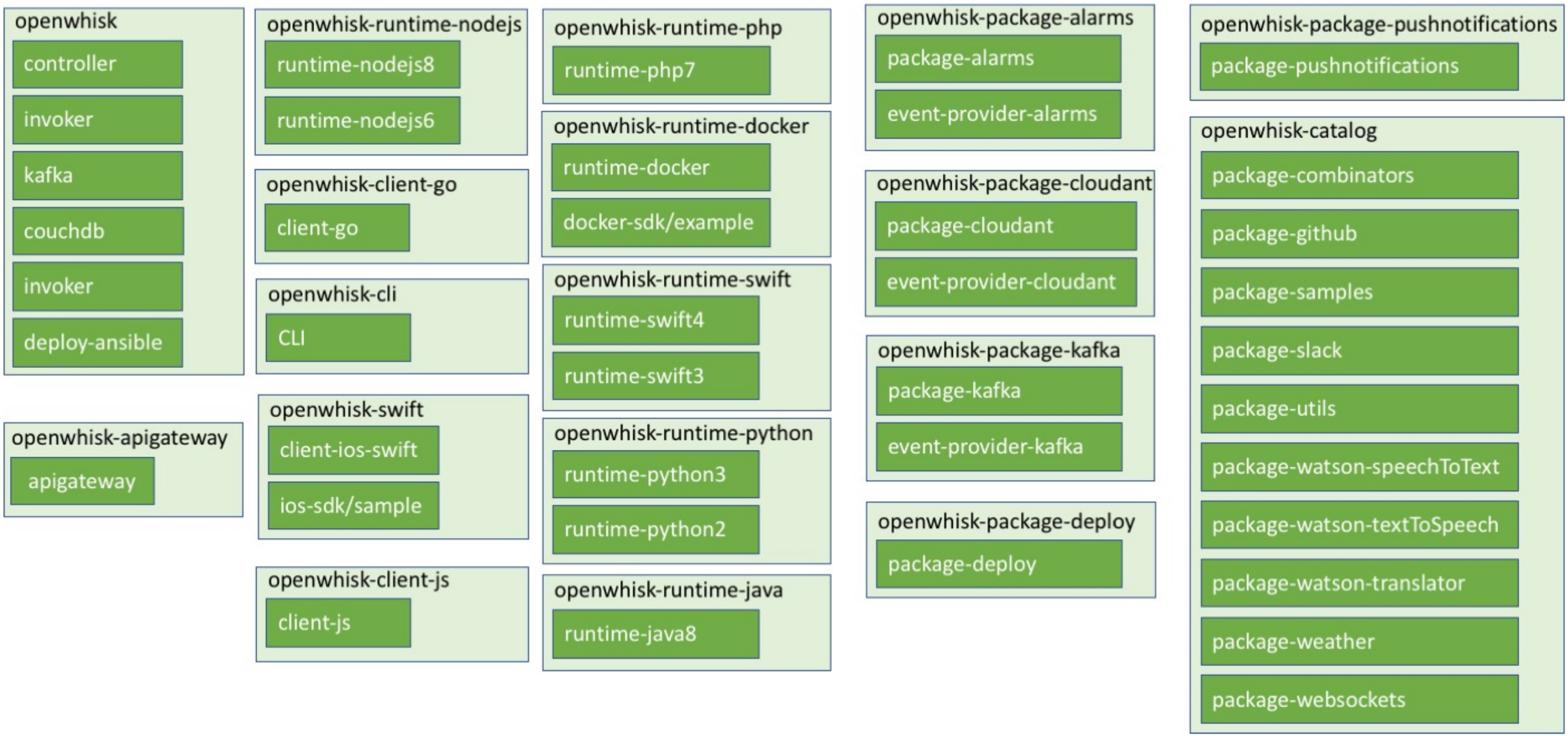
Fig: Serverless framework workflow

- User deploy the code to serverless functions such as Amazon lambda, Openwhisk
- When users run actions, functions are run in docker containers.
- Users can use either HTTP or Cli tool to pass input data
- These containers are run for short period of time.

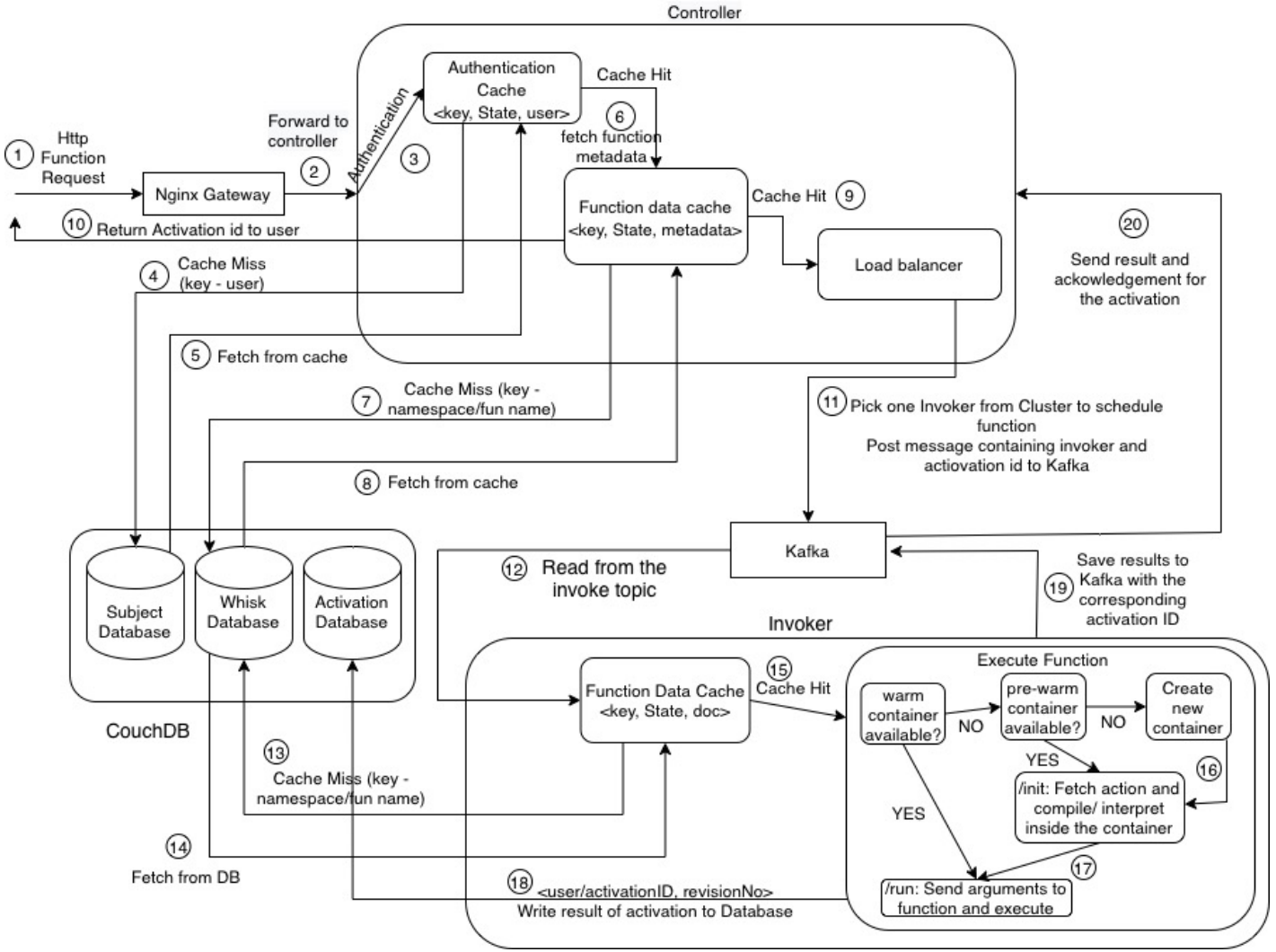
Target systems

- Domain
  - Serverless architecture
- System (Framework)
  - Openwhisk (an opensource serverless framework)
  - https://openwhisk.apache.org/
  - It is in production since December 2016
- Language
  - Scala

Apache OpenWhisk Components



What happens when a function is run?



Component

- Input
  - Code: python, Javascript
  - Docker Parameter
    - Memory allocation
    - Input parameters
- Component
  - Connector
    - Connects fuzzer with the framework
    - https://github.com/apache/openwhisk-cli
  - Runner
    - Responsible for feeding fuzzer with input
    - Collects the result of fuzzer in success and error file
  - Fuzzing
    - Mutation fuzzing: input parameter
    - Random fuzzing: memory and code
  - Api Server
    - Collects logs detail from the framework for each execution
    - Each log is maintained by transaction id

Scheduler

- Fuzzing techniques
  - Through log analysis, We identify the number of function that visited during each execution run.
  - Memory usages is identified from docker stats
  - Fuzzer guides with code coverage, success execution, memory usages and last function execution time

Results

- Reached target branch with in 3 seconds

```
docker_name,language,memory_used,execution_time,no_of_error,code_coverage
wsk0_519_guest_md5,py,44666880,3.6091318130493164,0,29
wsk0_520_guest_md5,py,46231552,3.4257099628448486,0,29
wsk0_521_guest_md5,py,44126208,3.355134963989258,0,29
wsk0_522_guest_md5,py,46301184,3.447938919067383,0,29
wsk0_523_guest_md5,js,94113792,3.7959909439086914,0,29
wsk0_524_guest_md5,js,84389888,3.574631929397583,0,29
wsk0_525_guest_md5,js,96120832,4.052873134613037,0,29
wsk0_526_guest_md5,py,57270272,3.5761187076568604,0,29
wsk0_527_guest_md5,py,49606656,3.3906590938568115,0,29
wsk0_528_guest_md5,js,100220928,3.6105830669403076,0,29
wsk0_529_guest_md5,py,42745856,3.276675224304199,0,29
wsk0_530_guest_md5,js,86179840,3.3915131092071533,0,29
wsk0_531_guest_md5,py,42291200,3.3185746669769287,0,29
wsk0_532_guest_md5,py,44576768,3.3450729846954346,0,29
wsk0_533_guest_md5,py,44281856,5.863972187042236,0,29
```

Fig: extracted metrics

Code

- https://github.com/Man1ish/fuzzerapplication

Fuzzing in serverless architecture

This project creates the fuzzer for serverless architecture

Requirement Installation

Download and run openwhisk framework

```
#download openwhisk
git clone https://github.com/apache/openwhisk.git
#running the framework
cd openwhisk
./gradlew core:standalone:bootRun

#download and setup the wsk tool
https://github.com/apache/openwhisk-cli

wsk property set \
  --apihost 'http://localhost:3233' \
  --auth '23bc46b1-71f6-4ed5-8c54-816aa4f8c502:123zO3xZCLrMN6v2BKK1dXYFpXlPkccOFqm12CdAsMgRU4VrNZ9ly

#clone the repo
git clone https://github.com/Man1ish/fuzzerapplication.git

#setup
python3 -m venv venv
venv/bin/python -m pip install --upgrade pip
venv/bin/python -m pip install -r requirements.txt

#run the api server
python api_server.py
```

Run project

To run the fuzzer

```
python main.py
```

Related work

- RESTler: Stateful REST API Fuzzing
  - Analyzes the API specification of a cloud service and generates sequences of requests that automatically test the service through its API.
  - A request B should be executed after request A. It is similar to the state function in serverless.

Thank you