AI for Fuzz Testing Web APIs

Prof. Andrea Arcuri

Kristiania University College and OsloMet

In this talk

- About myself
- Importance of Software Testing
- Search-Based Software Testing
- Fuzzing Web APIs with EvoMaster
- Applications and Success Stories
- Demo: EvoMaster

- Prof. Andrea Arcuri
- Italian
- Work in Norway, Oslo
- Kristiania University College and OsloMet
- PhD in 2009 on AI applied to Software Testing, UK
- Worked in industry few years as Senior Engineer
- Main research interest: Search-Based Software Testing (**SBST**)
- Lead of Artificial Intelligence in Software Engineering (AISE) Lab

Importance of Software Testing

Software is Everywhere

Are software applications doing what are they supposed to do?

Ariane 5 – ESA

On June 4, 1996, the flight of the Ariane 5 launcher ended in a failure.

$500 millions in cost

Software bug

Fatal Therac-25 Radiation

1986, Texas, person died

Power Shutdown in 2003

Nearly 50 millions persons affected in Canada/US

2010, Toyota, bug in braking system, 200 000 cars recalled

Knight Capital Group 2012

$460 millions lost in 45 minutes of trading due to bug

March 2019, Boeing 737 Max **crashed** due to **software problems**; all **157 people on board died**.

2009-2018: Estimated 135-270 deaths in UK

450,000 Women Missed Breast Cancer Screenings Due to "Algorithm Failure" › A disclosure in the United Kingdom has sparked a heated debate about the health impacts of an errant algorithm

BY ROBERT N. CHARETTE | 11 MAY 2018 | 3 MIN READ

2024: CrowdStrike, 8.5M Machines Down

The 2024 Crowdstrike Update Fail and What it Means for Security

By Editorial Team | August 13, 2024

When Crowdstrike CEO George Kurtz said that the July 2024 update failure that caused a global IT shutdown "was not a security issue", he was rather downplaying this incident. After all, *availability* is part of the CIA triad; a fundamental concept in information security. With systems rendered inoperable, data inaccessible, and thousands of companies plunged into crisis mode, here's a brief overview of the Crowdstrike update fail and some takeaways from a security standpoint.

Crowdstrike 2024 Bug: What Happened?

The veritable tech meltdown caused by CrowdStrike's flawed update impacted 8.5 million Windows devices. Chaos ensued in industries like healthcare and travel—two German hospitals had to cancel elective operations. Cyber insurers estimate they'll have to pay out at least $1.5 billion while Fortune 500 companies, many of which use Falcon, could face financial hits of up to $5.4 billion.

The actual issue stemmed from an update error in Falcon, which is a cloud-based endpoint protection platform developed by CrowdStrike. Falcon protects against a wide range of cyber threats through a single lightweight agent installed on company endpoint devices (think PoS devices, workstations, laptops, etc). This specific update was for devices running Windows, and it caused blue screens of death on Windows devices running Falcon.

The tricky thing with platforms like Falcon is that they need certain updates to be rolled out as fast as possible to ensure proper protection against threats on endpoint devices. To slightly oversimplify, the issue was in an update to the product's antivirus signatures rather than the underlying software.

Still, though, the updates should've been staggered so that every CrowdStrike customer wasn't impacted simultaneously. Crowdstrike should've rolled it out and tested it on a computer running Windows in their own testing environment. And, companies that use Falcon should also have the option to select whether to apply updates to all of their systems at once or try it out first.

And I could go on the whole day…

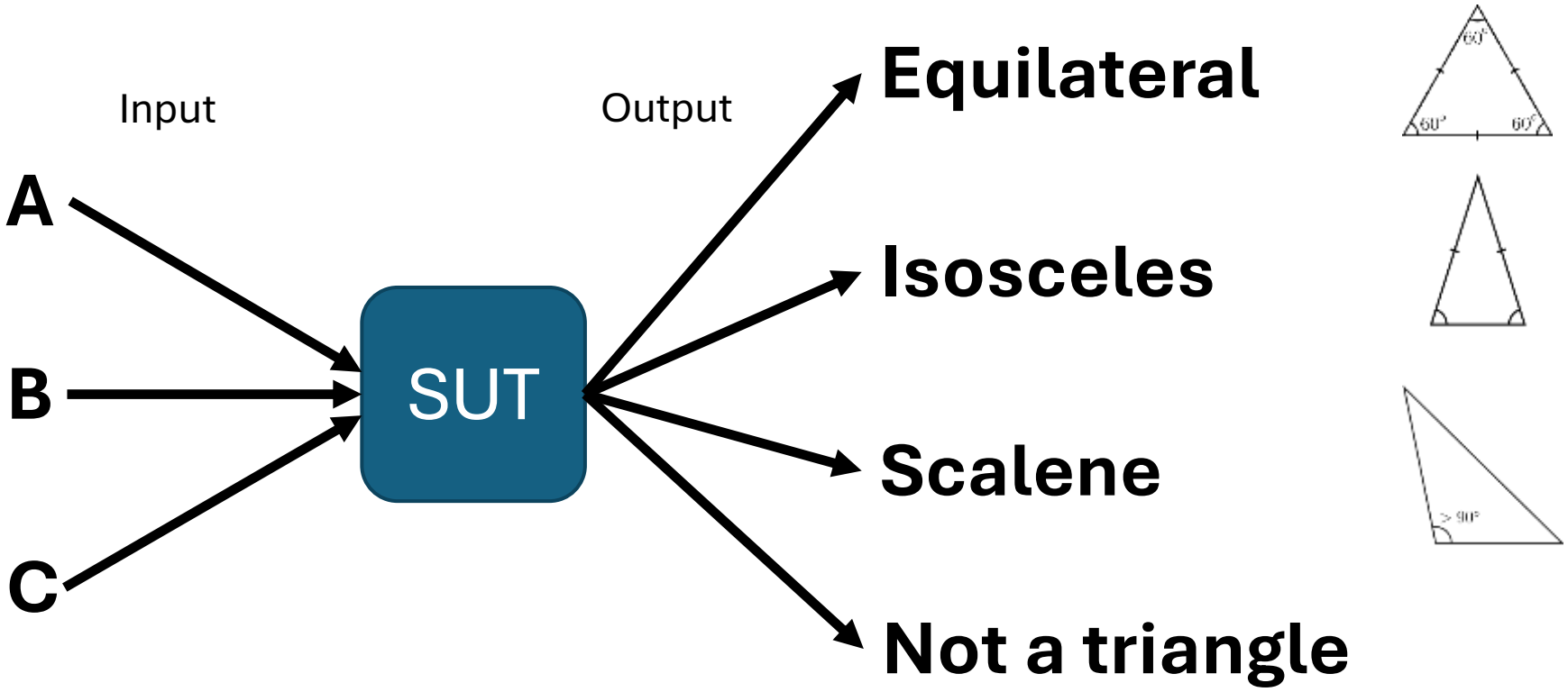
- As of 2013, estimated that software testing costing **$312 billions** worldwide
- In 2016, 548 recorded and documented software failures impacted **4.4 billion** people and **$1.1 trillion** in assets worldwide

What to do? **Test** the software

But how to test “*properly*”?

Manual testing is expensive, tedious and of limited effect

Example: Triangle Classification (TC)



- 3 integer numbers (A, B and C) as input representing the length of the *edges*
- 4 possible outcomes
- Does the *system under test* (SUT) give the right answer?

How to test TC?

- If numbers are 32 bit integers, there are $2^{32} * 2^{32} * 2^{32} = 2^{96} = 79{,}228{,}162{,}514{,}264{,}337{,}592{,}626{,}226{,}666$ possible combinations
  - ie, 79 Octillion possible combinations of edge lengths
- Cannot test all of them
- Need to define some *test criteria* to decide a good enough test suite which is:
  1. good at finding bugs
  2. small enough to be manageable

BlackBox Testing: eg, 1 Test per Output

- **t0:(A=42, B=42, C=42) => EQUILATERAL**
- **t1:(A=42, B=42, C=5) => ISOSCELES**
- **t2:(A=42, B=43, C=44) => SCALENE**
- **t3:(A=42, B=42, C=12345) = NOT A TRIANGLE**
- *Would such 4 test cases be enough?*
- What if the EQUILATERAL case is implemented with just something as naïve as "**if A==B and B==C then EQUILATERAL**"?
  - (**A=-3, B=-3, C=-3**) would wrongly return EQUILATERAL instead of NOT A TRIANGLE
  - Just checking basic scenarios is not enough

White-Box Testing

- Code can have bugs
- *To trigger a bug, the code must be executed*
- But code can have very complex control flow
- Some rare "paths" in the code might be executed only in very complex scenarios
- *Goal: in a test suite, have each single line and branch be executed at least once*

```java
public Classification classify(
                        int a, int b, int c){

    if(a <=0 || b<= 0 || c<= 0){
        return Classification.NOT_A_TRIANGLE;
    }

    if(a==b && b==c){
        return Classification.EQUILATERAL;
    }

    int max = Math.max(a, Math.max(b,c));

    if( (max == a && max -b -c >= 0 ) ||
            (max == b && max -a -c >= 0 ) ||
            (max == c && max -a -b >= 0 ) ){
        return Classification.NOT_A_TRIANGLE;
    }

    if(a==b || b==c || a==c){
        return Classification.ISOSCELES;
    } else {
        return Classification.SCALENE;
    }
}
```

Example

- **if((max == a && max -b -c >= 0) ||**
  (max == b && max -a -c >= 0) ||
  (max == c && max -a -b >= 0))
- In this disjunction of 3 different clauses, if in your test suite the first clause is always true, the other 2 would never be executed
  - so if wrong, you would not know
- This is a TRIVIAL example… real industrial software can be way more complex…
- Writing tests for each path is not only tedious, but can be quite hard as well

Oracle Problem

- Given **f(x)=y**, how do I know that **y** is the correct output for **x**???
- Need an *"oracle"* to determine the correctness of output
- Easiest oracle: *has the program crashed*?
  - In this case, **y** is not correct and we have a bug
  - But not all bugs lead to a program crash…
- We get an output, might not always be easy to tell if correct

Is this correct?

(A=42, B=42, C=12345) = NOT A TRIANGLE

What about this one?

(A=890321, B=1661466711, C=7711452) = NOT A TRIANGLE

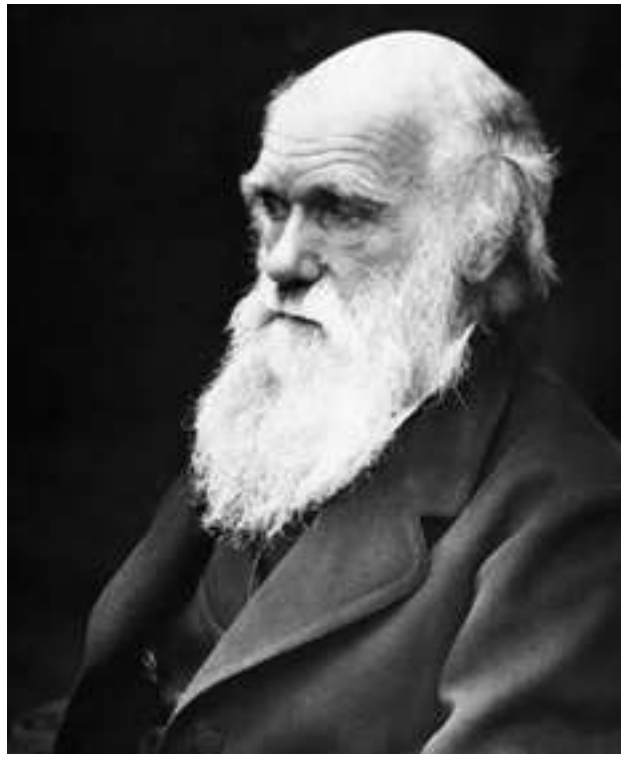
Automated Test Case Generation

- **Automatically generate test cases**
- Model software testing as an **optimization problem**
  - Maximize code coverage
  - Find bugs
  - Etc.
- Use optimization algorithms
- Benefits: *cheaper and more effective than manual testing*
- *Hard problem to automate*
  - given a non-linear constraint, there is no guaranteed algorithm that can solve it in polynomial time

Search-Based Software Testing

Search-Based Software Testing (SBST)

- Biology meets Software Engineering (SE)
- Casting SE problems into *Optimization Problems*
- *Genetic Algorithms*: one of most famous optimization algorithm, based on theory of evolution
- *Evolve* test cases



1976: First use of optimization in SE

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-2, NO. 3, SEPTEMBER 1976

Automatic Generation of Floating-Point Test Data

WEBB MILLER AND DAVID L. SPOONER

Abstract—For numerical programs, or more generally for programs with floating-point data, it may be that large savings of time and storage are made possible by using numerical maximization methods instead of symbolic execution to generate test data. Two examples, a matrix factorization subroutine and a sorting method, illustrate the types of data generation problems that can be successfully treated with such maximization techniques.

Index Terms—Automatic test data generation, branching, data constraints, execution path, software evaluation systems.

Introduction

RESEARCH in program evaluation and verification has only rarely (e.g., [1]) begun with the explicit requirement that the program deal with real numbers as opposed to integers. This may be an oversight since there are theoretical results which suggest the desirability of this assumption. Specifically, a general procedure of Tarski [2] shows that certain properties, undecidable (in the technical sense) for "integer" programs, are decidable for "numerical" "heuristic" in that it is not guaranteed to produce a set of test data executing a given path whenever such data exist. (On the other hand, we know of no guaranteed data generation scheme whose execution time does not, in the worst case, grow at least exponentially with the length of the execution path.)

Numerical Maximization Methods for Generating Test Data

Given the problem of generating floating-point test data our approach begins by fixing all integer parameters of the given program (e.g., the dimensions of the data in a matrix program or the number of iterations in an iterative method) so that the only unresolved decisions controlling program flow are comparisons involving real values. Then, as will be seen, an execution path takes the form of a straight-line program of floating-point assignment statements interspersed with "path constraints" of the form $c_i = 0$, $c_i > 0$, or $c_i \geq 0$. Each c_i is a

Properties of Optimization Problems

- 2 main components: *Search Space* and *Fitness Function*
- **Goal**: find the best solution from the search space such that the fitness function is minimized/maximized

Search Space

- Set X of all possible solutions for the problem
- If a solution can be represented with 0/1 bit sequence of length N, then search space is all possible bit strings of size N
  - any data on computer can be represented with bitstrings
- Search space is usually huge, eg 2^N
  - Otherwise use brute force, and so would not be a problem

0	0	1	1	1	1	0	1	0	1

Fitness Function

- $f(x)=h$
- Given a solution x in X, calculate an heuristic h that specifies how good the solution is
- Problem dependent, to minimize or maximize:
  - Maximize code coverage
  - Maximize fault finding
  - Minimize test suite size
  - etc.

Optimization Algorithms

- Algorithm that explores the search space X
- Only a tiny sample of X can be evaluated
- Use fitness *f(x)* to guide the exploration to fitter areas of the search space with better solutions
- Stopping criterion: after evaluating K solutions (or K amount of time is passed), return best *x* among the evaluated solutions
- Many different kinds of optimization algorithms…
  - But as a user, still need to provide the representation and f(x)

Trivial Example

- Search space: ~4 billion values
- Only 1 value cover the *if* branch
- Covering *"OK"* at random is extremely unlikely
- Need some heuristics to driver the search

```
public String foo(int x) {
  if(x == 42)
    return "OK";
  return "NOPE";
}
```

SBST Heuristics: Branch Distance

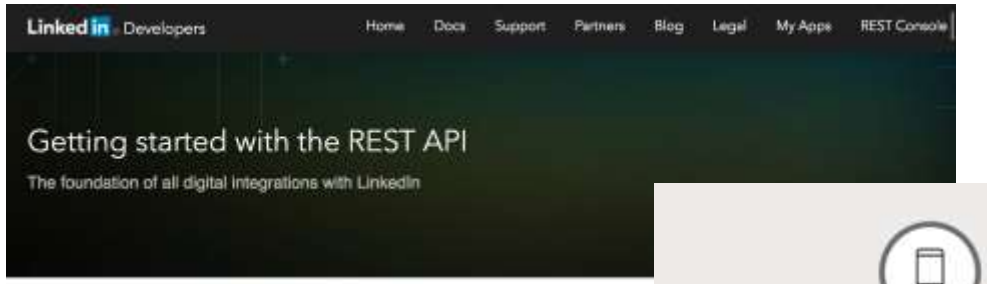
- Standard technique in the SBST literature
- Example: *if(x==42)*
- Both 5 and 900 do not solve the constraint, but 5 is *heuristically* closer
  - *d*(x==42)=|x-42|
  - *d* function to minimize
- Not just for integers, but also all other types, eg strings
- Need to *instrument* the code to calculate those branch distances
- **Trivial example, but there are many more sophisticated heuristics**

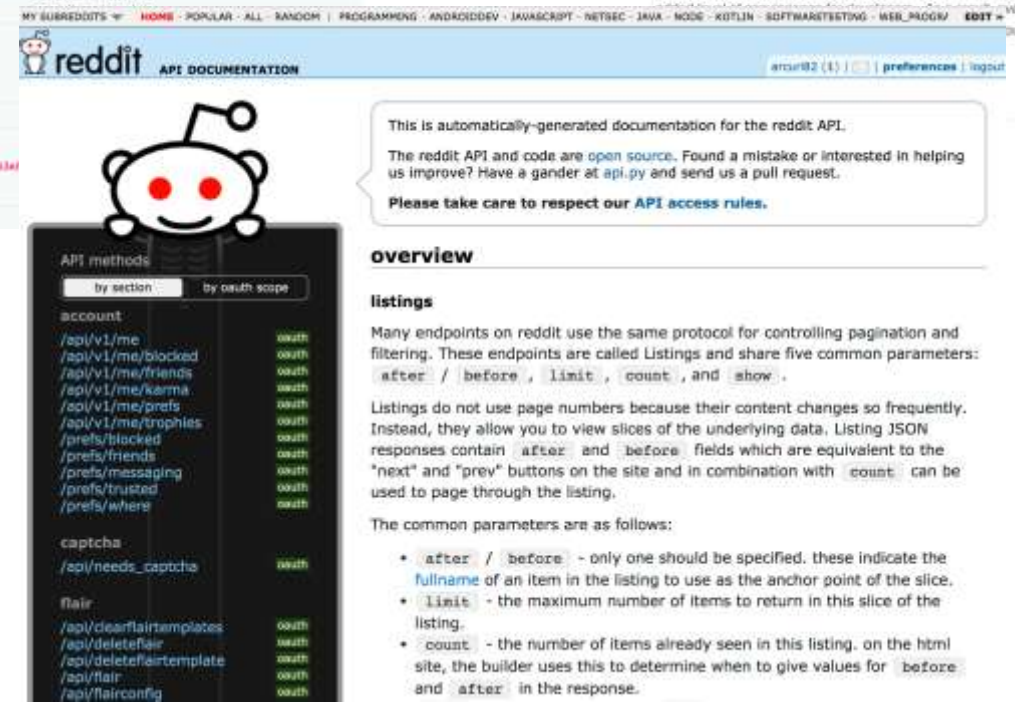
Fuzzing Web APIs with EvoMaster

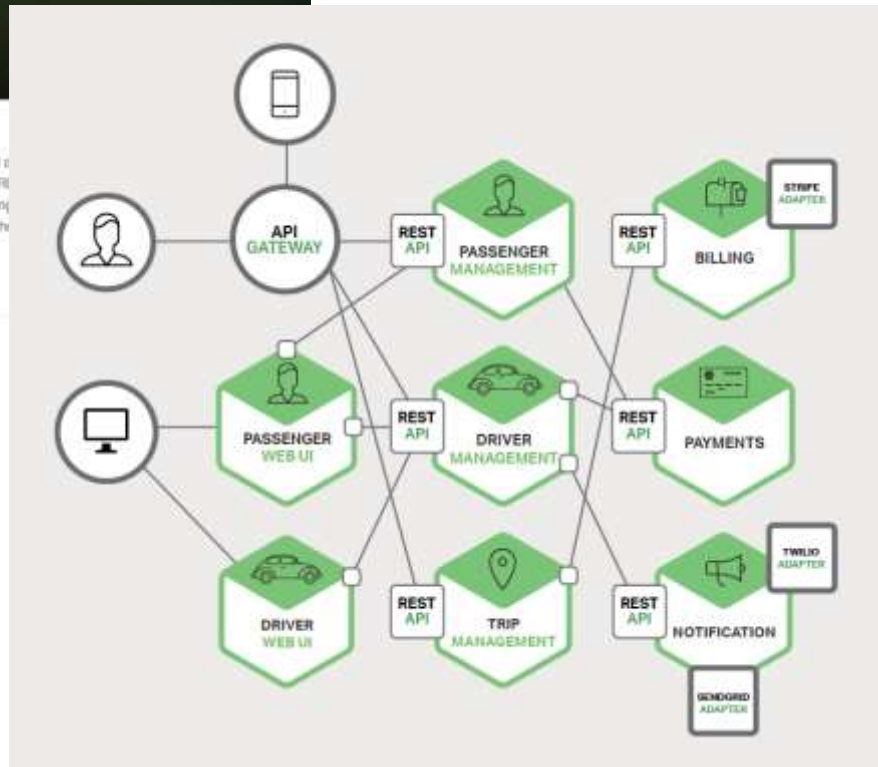
Web Services

- Providing APIs (Application Programming Interfaces) over network, remote servers
- Communications over UDP/TCP, with protocols like HTTP
- Different types of data transfer formats
  - JSON, XML, HTML, plain text, etc.
- Permanent storage:
  - eg, SQL/NoSQL databases
- **REST API**s most common type of web services
  - others are *SOAP, GraphQL and gRPC*

REST APIs are used everywhere...







REST Testing Challenges

- How to choose **query** and **path** parameters?
- How to prepare **body payloads** (e.g. JSON)?
- How to choose data to insert into **SQL** databases?
- Goals:
  - **Finding faults** (eg crashes)
  - **Maximize code coverage** (eg, regression tests)
- Writing high coverage tests *by hand* for every single endpoint is time consuming

What about **Automated Test Generation** for RESTful APIs?

- Automatically write all the test cases
- Not just execution, but choice of all the inputs
- Hard, complex problem

2 Uses of Generated Tests

- If automated oracles: **automatically detect faults**
  - e.g., HTTP response giving 500
- No oracles / faults: **regressing testing**
  - Tests can be added to Git, to capture current behavior of system
  - If in future introduce new bug that breaks functionality, regression tests will start to fail

Fuzzers

- Tools that automatically generate test inputs
- Different strategies: from **random** inputs to advanced **AI** techniques
  - eg, SBST, ML and LLM
- Can automatically create and evaluate **millions** of test cases
- Used in many different domains
  - eg, parser libraries and unit testing
- REST fuzzing is a more recent development
  - eg, Restler, Schemathesis, RESTest, Fuzz-Lightyear and EvoMaster

https://github.com/microsoft/restler-fuzzer

microsoft / restler-fuzzer

Type / to search

Code | Issues 251 | Pull requests 12 | Actions | Projects | Security | Insights

restler-fuzzer

Public

Watch 36 | Fork 289 | Star 2.5k

main | 53 Branches | 23 Tags | Go to file t | Add file | Code

marina-p Remove binary drop instructions from README. (#898) 694cc9e · 2 months ago 403 Commits

.github/ISSUE_TEMPLATE	Add RESTler version to the bug report template. (#595)	2 years ago
.vscode	Support replay mode with garbage collection (#845)	5 months ago
ci_build_pipelines	Update RESTler version to 9.2.4 (#868)	5 months ago
demo_server	Resolve security alerts for fastapi dependencies. (#893)	3 months ago
docker	Update dockerfile for .net upgrade (#604)	2 years ago
docs	Search mode default and doc updates (#887)	4 months ago
restler	Added a TERMINATING_CHUNK_DELIM constant for better h...	2 months ago
src	Fix Content-Type custom payload bugs (#888)	4 months ago
utilities	Extend spec coverage file with concrete requests and respon...	3 years ago
.gitignore	New experimental checker for invalid value fuzzing. (#589)	2 years ago
CODE_OF_CONDUCT.md	Restler v6.1 - initial commit	4 years ago
Dockerfile	update python path expectations (fixes #687) (#691)	last year

About

RESTler is the first stateful REST API fuzzing tool for automatically testing cloud services through their REST APIs and finding security and reliability bugs in these services.

- Readme
- MIT license
- Code of conduct
- Security policy
- Activity
- Custom properties
- 2.5k stars
- 36 watching
- 289 forks

Report repository

Releases

23 tags

https://github.com/schemathesis/schemathesis

schemathesis / schemathesis

Type / to search

Code | Issues 56 | Pull requests 2 | Discussions | Actions | Projects | Security | Insights

schemathesis

Public

Sponsor | Watch 20 | Fork 153 | Star 2.2k

master | 22 Branches | 279 Tags | Go to file t | Add file | Code

Stranger6667 chore: Release 3.34.1 33025ea · 1 hour ago 2,416 Commits

.github	build(deps): Bump github/codeql-action from 3.26.0 to 3.26.2	yesterday
benches	feat: Add a way to replace impl	last month
corpus	chore: Rework corpus	3 months ago
docs	chore: Release 3.34.1	1 hour ago
example	feat: Flexible operations filter	last month
img	docs: Update documentation	9 months ago
src/schemathesis	fix: Error in `response_header_conformance` if the header definit...	12 hours ago
test-corpus	chore: Sort imports	2 months ago
test	fix: Error in `response_header_conformance` if the header definit...	12 hours ago
.dockerignore	docs: Update README	last year
.gitignore	chore: Remove duplicate pattern from .gitignore	4 years ago
.gitmodules	test: Add tests on schemas from Open API catalog	4 years ago

About

Supercharge your API testing, catch bugs, and ensure compliance

schemathesis.readthedocs.io

testing | graphql | cli | swagger | openapi | pytest | property-based-testing | hypothesis | hacktoberfest | openapi3

Readme
MIT license
Code of conduct
Cite this repository
Activity
Custom properties
2.2k stars
20 watching
153 forks
Report repository

Releases 277

https://github.com/isa-group/RESTest

isa-group / RESTest

Type / to search

Code | Issues 4 | Pull requests 2 | Actions | Projects 2 | Wiki | Security | Insights

RESTest

Public

Watch 16 | Fork 34 | Starred 205

master | 45 Branches | 12 Tags | Go to file | Add file | Code

josgarmar31 Merge pull request #273 from isa-group/feature/#272 be6aab3 · 2 months ago 1,280 Commits

.circleci	CIRCLECI	last year
allure	Fix permissions of Allure scripts	2 years ago
docs	Update README	10 months ago
src	ReadProperties Modification	2 months ago
.gitignore	Fix errors and refactor	10 months ago
LICENSE	LICENSE updated	4 years ago
README.md	Update README.md	5 months ago
pom.xml	[maven-release-plugin] prepare for next development iterati...	3 months ago

README | LGPL-3.0 license

RESTest

About

RESTest: Automated Black-Box Testing of RESTful Web APIs

java testing rest rest-api swagger openapi api-rest oas api-testing

Readme
LGPL-3.0 license
Activity
Custom properties
205 stars
16 watching
34 forks
Report repository

Releases 7

restest-1.5.0 Latest
on May 20

+ 6 releases

Contributors 9

https://github.com/Yelp/fuzz-lightyear

Yelp / fuzz-lightyear

Type / to search

Code | Issues 12 | Pull requests 6 | Actions | Projects | Security | Insights

fuzz-lightyear

Public

Watch 8 | Fork 25 | Star 205

master | 15 Branches | 11 Tags | Go to file t | Add file | Code

Chandra158 Merge pull request #94 from Yelp/update-changelog aeaae96 · 3 months ago 226 Commits

.github/workflows	Update Github action PyPi publish version	6 months ago
docs	adding logo	3 years ago
fuzz_lightyear	update version	3 months ago
scripts	version bump; adding uploader script	5 years ago
test_data	adding --ignore-non-vulnerable flag (#53)	4 years ago
testing	Update generator to consider 201 as success response code	last year
tests	Update generator to consider 201 as success response code	last year
.gitignore	adding database support, and apikey auth to testing app	5 years ago
.pre-commit-config.yaml	Add Python 3.8 support and drop 3.7	last year
CHANGELOG.md	update changelog: 0.0.11	3 months ago
CONTRIBUTING.md	adding documentation	5 years ago
LICENSE	fix LICENSE	5 years ago
Makefile	nicer cURL output	5 years ago

About

A pytest-inspired, DAST framework, capable of identifying vulnerabilities in a distributed, micro-service ecosystem through chaos engineering testing and stateful, Swagger fuzzing.

Readme
View license
Activity
Custom properties
205 stars
8 watching
25 forks
Report repository

Releases

11 tags

Packages

No packages published

Contributors 9

https://github.com/WebFuzzing/EvoMaster

WebFuzzing / EvoMaster

Type / to search

Code | Issues 25 | Pull requests 8 | Discussions | Actions | Projects | Wiki | Security 174 | Insights | Settings

EvoMaster

Public

Edit Pins | Unwatch 24 | Fork 77 | Starred 469

master | 128 Branches | 24 Tags | Go to file | Add file | Code

arcuri82 Merge pull request #1052 from WebFuzzing/fix/solver-tests — 084ce79 · 31 minutes ago — 10,172 Commits

.circleci	clarification	3 years ago
.github	3.1.1-SNAPSHOT	last week
.mvn	trying workaround for Kotlin compiler issue	last year
client-java	3.1.1-SNAPSHOT	last week
core-driver-it	3.1.1-SNAPSHOT	last week
core-graphql-it	3.1.1-SNAPSHOT	last week
core-it	integration test for link support	last week
core	fix for #989	yesterday
dbconstraint	3.1.1-SNAPSHOT	last week
docs	library documentation for JS and PY	last week
e2e-tests	fix for #989	yesterday
outdated-client-dotnet	outdating no longer supported white-box testing of backen...	last month
outdated-client-js	outdating no longer supported white-box testing of backen...	last month

About

The first open-source AI-driven tool for automatically generating system-level test cases (also known as fuzzing) for web/enterprise applications. Currently targeting whitebox and blackbox testing of Web APIs, like REST, GraphQL and RPC (e.g., gRPC and Thrift).

kotlin | java | testing | graphql | rest | grpc | thrift | evolutionary-algorithms | fuzzing | api-rest | fuzzer | api-testing | rpc-api | test-case-generation | search-based-software-testing

Readme

LGPL-3.0 license

Activity

Custom properties

469 stars

24 watching

77 forks

Report repository

Releases 10

Input: OpenAPI/Swagger Schema

- Need to know what endpoints are available, and their parameters
- Schema defining the APIs
- OpenAPI is the most popular one
- Defined as JSON file, or YAML

Example: PetStore

- Online schema at https://petstore3.swagger.io/api/v3/openapi.json

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON


```
openapi:                "3.0.2"
info:
  title:                "Swagger Petstore - OpenAPI 3.0"
  description:          "This is a sample Pet Store Server based on the OpenAPI 3.0 specification.  You can find out more
                        about\nSwagger at [http://swagger.io](http://swagger.io). In the third iteration of the pet store, we've
                        switched to the design first approach!\nYou can now help us improve the API whether it's by making changes to
                        the definition itself or to the code.\nThat way, with time, we can improve the API in general, and expose
                        some of the new features in OAS3.\n\nSome useful links:\n- [The Pet Store repository](https://github.com/swag
                        ger-api/swagger-petstore)\n- [The source API definition for the Pet Store](https://github.com/swagger-api/swa
                        gger-petstore/blob/master/src/main/resources/openapi.yaml)"
  termsOfService:       "http://swagger.io/terms/"
  contact:
    email:              "apiteam@swagger.io"
  license:
    name:               "Apache 2.0"
    url:                "http://www.apache.org/licenses/LICENSE-2.0.html"
  version:              "1.0.19"
externalDocs:
  description:          "Find out more about Swagger"
  url:                  "http://swagger.io"
servers:
  0:
    url:                "/api/v3"
```

BlackBox: What Can Expect?

- All these tools will analyze the schema
- Send requests with many different strategies
  - there is lot of research in academia on this
- Check if any error in the API can be identified
- Output executable test cases
  - in different formats, eg Java and Python

```
$ evomaster.exe --blackBox true --bbSwaggerUrl https://petstore3.swagger.io/api/v3/openapi.json --bbTargetUrl https://petstore3.swagger
.io --maxTime 30s --ratePerMinute 60 --outputFormat JAVA_JUNIT_5
*
```

```
* EvoMaster version: 3.1.0
* Loading configuration file from: C:\Users\arcur\WORK\code\EvoMaster\em.yaml
* WARNING: You are doing Black-Box testing, but you did not specify the 'problemType'. The system will default to RESTful API test
ing.
* Initializing...
10:31:47.901 [main] WARN  o.e.c.problem.rest.param.BodyParam - Not supported data type: application/octet-stream
* There are 19 usable RESTful API endpoints defined in the schema configuration
10:31:47.955 [main] WARN  o.e.c.s.gene.optional.ChoiceGene - cannot bind ChoiceGene with StringGene
* Starting to generate test cases
* Consumed search budget: 123.950%
* Covered targets: 44; time per test: 7427.2ms (7.6 actions); since last improvement: 8s
* Starting to apply minimization phase
* Recomputing full coverage for 5 tests
* Analyzing 5 tests with size greater than 1
* Minimization progress: 5/5
* Minimization phase took 76 seconds
* Evaluated tests: 5
* Evaluated actions: 38
* Needed budget: 100%
* Passed time (seconds): 114
* Execution time per test (ms): Avg=7427.20 , min=2995.00 , max=9006.00
* Execution time per action (ms): Avg=980.59 , min=904.11 , max=1000.67
* Computation overhead between tests (ms): Avg=15223.60 , min=4.00 , max=76086.00
* Going to save 21 tests to generated_tests
10:33:42.319 [main] WARN  o.e.c.o.service.HttpWsTestCaseWriter - Currently no assertions are generated for response type: application/x
ml
10:33:42.323 [main] WARN  o.e.c.o.service.HttpWsTestCaseWriter - Unhandled type for body payload: application/xml
* Potential faults: 13
* Successfully executed (HTTP code 2xx) 7 endpoints out of 20 (35%)
* EvoMaster process has completed successfully
* Use --help and visit http://www.evomaster.org to learn more about available options
```

```
@Test @Timeout(60)
public void test_1() throws Exception {

  given().accept("application/xml")
        .contentType("application/json")
        .body(" { " +
          " \"id\": 940, " +
          " \"name\": \"doggie\", " +
          " \"photoUrls\": [ " +
          " \"yHQXry\", " +
          " \"AZOgWb5y\", " +
          " \"GROBCmON\" " +
          " ], " +
          " \"tags\": [ " +
          " {}, " +
          " { " +
          " \"name\": \"nosupgc\" " +
          " } " +
          " ], " +
          " \"status\": \"pending\" " +
          " } ")
        .post(baseUrlOfSut + "/api/v3/pet")
        .then()
        .statusCode(200)
        .assertThat()
        .contentType("application/xml");
}
```

Success Calls: Random but Valid Data

Crashing with 500

```
@Test @Timeout(60)
public void test_4_with500() throws Exception {
   ExpectationHandler expectationHandler = expectationHandler();

   ValidatableResponse res_0 = given().accept("application/xml")
         .get(baseUrlOfSut + "/api/v3/user/8WlY1")
         .then()
         .statusCode(500)
         .assertThat()
         .contentType("application/xml");

   expectationHandler.expect(ems)
      .that(sco, Arrays.asList(200, 400, 404).contains(res_0.extract().statusCode()));
}
```

Invalid response (eg status code not declared in schema)

```
@Test @Timeout(60)
public void test_8() throws Exception {
   ExpectationHandler expectationHandler = expectationHandler();

   ValidatableResponse res_0 = given().accept("application/json")
         .contentType("application/json")
         .body(" null ")
         .post(baseUrlOfSut + "/api/v3/store/order")
         .then()
         .statusCode(400)
         .assertThat()
         .contentType("application/json")
         .body(containsString("No Order provided. Try again?"));

   expectationHandler.expect(ems)
      .that(sco, Arrays.asList(200, 405).contains(res_0.extract().statusCode()));
}
```

What about some more advanced cases with SBST?

Dealing With SQL Databases

- Bytecode instrumentation to intercept all JDBC calls
- Find all SQL SELECT queries that return no data
  - eg due to WHERE clauses that are not satisfied
- Insert data directly into DB as part of the test case
  - Not always possible to create data with REST endpoints (eg POST/PUT)
  - using a JDBC connection
  - need to analyze DB's schema
- *Goal*: insert data such that SELECT are not empty
- *Challenges*: WHERE clauses might have complex constraints. Need search
- *Why?* Can have impact on code execution flow

Java Example Using Spring

```
@RequestMapping(
        path = "/{x}/{y}",
        method = RequestMethod.GET,
        produces = MediaType.APPLICATION_JSON
)
public ResponseEntity get(@PathVariable("x") int x, @PathVariable("y") int y) {

    List<DbDirectIntEntity> list = repository.findByXIsAndYIs(x, y);
    if (list.isEmpty()) {
        return ResponseEntity.status(400).build();
    } else {
        return ResponseEntity.status(200).build();
    }
}
```

Generated Test

```
@Test @Timeout(60)
fun test_1()  {
   val insertions = sql().insertInto("DB_DIRECT_INT_ENTITY", 14L)
         .d("ID", "-65536")
         .d("X", "-67108182")
         .d("Y", "0")
      .dtos()
   val insertionsresult = controller.execInsertionsIntoDatabase(insertions)

   given().accept("*/*")
         .get("${baseUrlOfSut}/api/db/directint/-67108182/0")
         .then()
         .statusCode(200)
         .assertThat()
         .body(isEmptyOrNullString())

}
```

- Arcuri et al. "*Handling SQL Databases in Automated System Test Generation*". TOSEM'20

Taint Analysis

- Inputs can have constraint checks
  - eg, strings matching a regex, numbers in a certain range and strings representing dates
- Constraints might be in code and NOT in the OpenAPI schema
- Can evolve inputs till satisfy constraints… eg using SBST heuristics
- … but what if inputs are not modified and used as they are? Can we do better?

Java Example Using Spring

```
@GetMapping(
    path = "/{date:\\d{4}-\\d{1,2}-\\d{1,2}}/{number}/{setting}",
    produces = MediaType.APPLICATION_JSON_VALUE)
public String getSeparated(
    @PathVariable("date") String date,
    @PathVariable("number") String number,
    @PathVariable("setting") String setting
){

  LocalDate d = LocalDate.parse(date);
  int n = Integer.parseInt(number);
  List<String> list = Arrays.asList("Foo", "Bar");

  if(d.getYear() == 2019 && n == 42 && list.contains(setting)){
    return "OK";
  }

  return "ERROR";
}
```

Solution

- Using bytecode instrumentation, check all JDK API usages
- Checking if input from HTTP is used without modification in a JDK call
- If yes, tell the search how input should be evolved
  - eg strings only representing valid dates, like for *LocalDate.parse(date)*
  - eg strings evolved always matching a particular regex
- Still need search to evolve the inputs
  - eg to handle constraints like *d.getYear() == 2019*
- Can dramatically boost the search efforts

Generated Test

```
@Test @Timeout(60)
fun test_4()  {

   given().accept("application/json")
         .get("${baseUrlOfSut}/api/testability/2019-12-10/42/Bar")
         .then()
         .statusCode(200)
         .assertThat()
         .contentType("application/json")
         .body(containsString("OK"))

}
```

Arcuri et al. *“Enhancing Search-Based Testing With Testability Transformations For Existing APIs”*. TOSEM’21

Applications and Success Stories

Experience With EvoMaster

- Author’s of EvoMaster
- Academic tool, started in 2016
  - Around 30M NOK in funding from ERC and NFR
- Applied on many open-source APIs
  - found thousands of bugs
- Only tool supporting *white-box* testing
  - but only for JVM
- Academic collaborations with industry

Open-Source Projects

- Found hundreds of faults in open-source projects
- Many APIs out there are not robust to receive invalid inputs, and so crashes
- https://github.com/WebFuzzing/EMB
- Marculescu et al. *"On the faults found in REST APIs by Automated Test Generation"*. TOSEM'22
- A. Arcuri et al. *"EMB: A Curated Corpus of Web/Enterprise Applications And Library Support for Software Testing Research"*. ICST'23.

Tool Comparisons

- Several new approaches have been developed for fuzzing Web APIs in recent years
- EvoMaster provided **best results** in tool comparisons
- Only tool doing white-box testing (all others support only black-box testing)
- Kim et al. *"Automated Test Generation for REST APIs: No Time to Rest Yet"*. ISSTA'22
- Zhang et al. *"Open Problems in Fuzzing RESTful APIs: A Comparison of Tools"*. TOSEM'23

EvoMaster at Meituan

- Large Chinese e-commerce enterprise
- EvoMaster used daily on hundreds of microservices, for millions of lines of code
- White-box testing Thrift RPC APIs

Table 5: Mean of Line%, Critical Line%, and #Detected Faults achieved by tests generated by *SM* EvoMaster with 1-hour time budget for 10 repetitions, and results of comparing with *Base* 1-hour using *Relative%* and Vargha-Delaney $\hat{A}_{12}$

SUT	Budgets by seeds	Line% Mean	Line% Relative%($\hat{A}_{12}$)	Critical Line% Mean	Critical Line% Relative%($\hat{A}_{12}$)	#Detected Faults Mean	#Detected Faults Relative%($\hat{A}_{12}$)
cs01	6.6%	16.6	**+26.7 (0.78)**	15.4	**+36.6 (1.00)**	17.2	+2.1 (0.42)
cs02	0.2%	19.3	+0.0 (0.53)	45.4	-1.5 (0.42)	26.4	-4.8 (0.38)
cs03	3.8%	7.1	+1.4 (0.40)	2.4	+1.6 (0.57)	7.0	+0.0 (0.50)
cs04	0.3%	13.0	+1.0 (0.57)	10.5	+1.7 (0.59)	43.8	+1.3 (0.64)
cs05	1.8%	22.6	**+6.5 (0.77)**	18.1	+7.1 (0.71)	63.2	-7.1 (0.27)
cs06	0.5%	8.5	**+6.1 (0.97)**	4.8	**+9.6 (0.96)**	13.0	+0.0 (0.50)
cs07	1.5%	17.6	**+44.9 (0.99)**	16.3	**+44.7 (0.99)**	79.7	+6.5 (0.72)
cs08	59.2%	10.1	-28.8 (0.05)	10.8	-23.5 (0.06)	38.8	-36.8 (0.04)
cs09	1.4%	15.3	+2.8 (0.65)	11.2	**+9.4 (0.73)**	35.1	**+3.4 (0.75)**
cs10	7.0%	9.2	**+39.1 (1.00)**	4.6	**+83.6 (1.00)**	5.0	+0.0 (0.50)
cs11	9.7%	16.0	**+76.1 (1.00)**	13.2	**+101.4 (1.00)**	46.9	+4.0 (0.66)
cs12	31.3%	12.4	**+24.6 (1.00)**	9.0	**+50.8 (1.00)**	56.1	**+6.9 (0.77)**
cs13	12.9%	24.6	**+25.8 (1.00)**	24.0	**+23.8 (1.00)**	36.6	-1.1 (0.33)
cs14	0.2%	14.6	+1.8 (0.52)	24.1	+5.9 (0.62)	67.6	+3.8 (0.59)
cs15	11.6%	15.9	**+118.1 (1.00)**	15.9	**+153.5 (1.00)**	33.7	+1.0 (0.55)
cs16	1.6%	16.7	**+17.1 (0.94)**	15.1	**+19.4 (0.92)**	45.4	+1.6 (0.59)
cs17	22.0%	15.8	**+7.4 (0.71)**	11.3	**+15.8 (0.82)**	60.9	-6.3 (0.23)
cs18	0.6%	6.9	**+18.5 (0.88)**	6.4	**+89.3 (0.97)**	70.6	**+17.2 (0.83)**
cs19	2.1%	10.8	**+11.6 (0.84)**	6.9	**+16.3 (0.83)**	63.9	+1.1 (0.54)
cs20	28.8%	21.8	**+14.8 (0.85)**	19.0	**+14.9 (0.82)**	35.8	-17.8 (0.30)
cs21	1.7%	10.3	**+25.7 (0.99)**	8.8	**+40.3 (0.99)**	47.8	+0.5 (0.50)
cs22	9.9%	16.3	**+18.1 (0.97)**	8.4	**+103.5 (1.00)**	56.8	-2.7 (0.39)
cs23	36.6%	10.2	**+134.3 (1.00)**	6.7	**+135.6 (0.93)**	110.7	-16.8 (0.31)
cs24	8.8%	10.4	**+11.1 (0.77)**	11.2	**+24.9 (0.94)**	93.2	+3.1 (0.60)
cs25	6.3%	10.0	+2.1 (0.69)	9.9	+4.3 (0.69)	61.2	-2.6 (0.31)
cs26	6.1%	12.8	**+42.7 (0.96)**	12.9	**+56.0 (0.92)**	106.8	+6.5 (0.60)
cs27	1.0%	22.1	**+17.3 (0.84)**	14.3	**+15.8 (0.83)**	69.0	**+10.2 (0.76)**
cs28	83.8%	23.1	**+92.7 (1.00)**	21.6	**+119.0 (1.00)**	48.6	-28.2 (0.07)
cs29	0.2%	2.4	+9.5 (0.59)	3.9	-8.0 (0.38)	42.6	-3.4 (0.24)
cs30	6.5%	8.7	**+66.5 (1.00)**	12.5	**+136.7 (1.00)**	69.0	**+4.4 (0.88)**
cs31	18.1%	9.6	**+11.5 (0.77)**	8.2	**+10.3 (0.72)**	72.4	-16.0 (0.12)
cs32	4.0%	9.4	**+24.9 (0.91)**	7.4	**+32.2 (0.94)**	84.9	+6.0 (0.59)
cs33	47.8%	9.7	+4.7 (0.61)	6.2	**+16.9 (0.77)**	105.1	-14.2 (0.34)
cs34	0.9%	9.8	**+14.9 (0.99)**	6.4	**+17.4 (0.98)**	77.4	+1.1 (0.57)
cs35	31.5%	10.1	**+16.5 (0.85)**	7.0	**+28.6 (0.93)**	111.3	+4.4 (0.59)
cs36	29.2%	11.8	**+58.1 (0.95)**	18.5	**+98.1 (1.00)**	129.1	**+12.4 (0.80)**
cs37	34.6%	9.1	**+35.6 (0.93)**	6.7	**+47.0 (0.96)**	104.7	-33.9 (0.11)
cs38	5.1%	18.7	**+30.1 (0.99)**	14.9	**+36.0 (1.00)**	101.0	+1.7 (0.58)
cs39	16.5%	8.0	+9.0 (0.62)	3.6	+13.2 (0.68)	96.4	+9.7 (0.61)
cs40	4.3%	12.6	**+34.4 (0.91)**	7.9	**+47.1 (0.91)**	100.0	**+13.8 (0.68)**
Mean		13.2	+26.9 (0.8)	12.0	+40.9 (0.8)	63.4	-1.7 (0.5)
#Relative > 0			39		37		23
#SM > *Base*			30		31		7
#*Base* > SM			1		1		8

- ASE'24: "*Seeding and Mocking in White-Box Fuzzing Enterprise RPC APIs: An Industrial Case Study*"
- 40 APIs at Meituan
- More than 5M LOC
- Automatically found hundreds of faults

EvoMaster at Volkswagen

- Large German car manufacturer
- EvoMaster used for black-box testing of REST APIs
- Recent collaborations
  - research articles under review, eg SBST vs LLM

Research Challenges

- Lot of open research challenges for better test generation strategies
- Cover larger parts of API code
- Find more faults (and fault types)
  - not all faults have same severity
- Test readability
  - testers still need to look at generated tests

https://dl.acm.org/doi/full/10.1145/3617175

ACM DIGITAL LIBRARY | Association for Computing Machinery

Browse About Si

Journals Magazines Proceedings Books SIGs Conferences People

Search ACM Digital Library

Journal Home Just Accepted Latest Issue Archive Authors Editors Reviewers About Contact Us

Home > ACM Journals > ACM Transactions on Software Engineering and Methodology > Vol. 33, No. 1 > Testing RESTful APIs: A Survey

SURVEY | OPEN ACCESS

Testing RESTful APIs: A Survey

Authors: Amid Golmohammadi, Man Zhang, Andrea Arcuri Authors Info & Claims

ACM Transactions on Software Engineering and Methodology, Volume 33, Issue 1 • Article No.: 27, Pages 1 - 41
https://doi.org/10.1145/3617175

Published: 24 November 2023 Publication History

Check for updates

6 6,013

PDF eReader

ACM Transactions on Software Engineering...
Volume 33, Issue 1

← Previous Next →

Abstract
1 Introduction
2 Background
3 Related Work
4 Research Method
5 Status of Publications in REST API Testing
6 Approaches to Testing REST API
7 Tools for Testing RESTful APIs

Abstract

In industry, RESTful APIs are widely used to build modern Cloud Applications. Testing them is challenging, because not only do they rely on network communications, but also they deal with external services like databases. Therefore, there has been a large amount of research sprout in recent years on how to automatically verify this kind of web services. In this article, we present a comprehensive review of the current state-of-the-art in testing RESTful APIs based on the analysis of 92 scientific articles. These articles were gathered by utilizing search queries formulated around the concept of RESTful API testing on seven popular databases. We eliminated irrelevant articles based on our predefined criteria and conducted a snowballing phase to minimize the possibility of missing any relevant paper. This survey categorizes and summarizes the existing scientific work on testing RESTful APIs and discusses the current challenges in the verification of RESTful APIs. This survey clearly shows an increasing interest among researchers in this field, from 2017 onward. However, there are still a lot of open research challenges to overcome.

https://github.com/WebFuzzing/EvoMaster/blob/master/docs/publications.md

Publications

The development of *EvoMaster* is rooted in academia. Here, you can find the PDFs of all the academic publications based on *EvoMaster*. Furthermore, slides of presentations can be found here. These can be useful if you want to know more on how *EvoMaster* works internally, e.g., details on the Many Independent Objective (MIO) algorithm.

To help to replicate previous studies, for most of these papers we also provide the scripts used to setup the experiments. This explained in more details here. Also, some of these papers provides full replication packages, which are linked directly in the papers (and not stored in this repository).

Recent arXiv Technical Reports, not Peer-Reviewed (Yet)

- M. Zhang, A. Arcuri, Y. Li, K Xue, Z Wang, J. Huo, W Huang. *Fuzzing Microservices In Industry: Experience of Applying EvoMaster at Meituan*. [arXiv]

Peer-Reviewed Publications

2024

- S. Seran. *Search-based Security Testing of Enterprise Microservices*. IEEE International Conference on Software Testing, Validation and Verification (ICST), Doctoral Symposium. [PDF]
- A. Arcuri, M. Zhang, J.P. Galeotti. *Advanced White-Box Heuristics for Search-Based Fuzzing of REST APIs*. ACM Transactions on Software Engineering and Methodology (TOSEM). [PDF][Scripts]

2023

- S. Seran, M. Zhang, A. Arcuri. *Search-Based Mock Generation of External Web Service Interactions*. Symposium on Search-based Software Engineering (SSBSE). [PDF]
- A. Golmohammadi, M. Zhang, A. Arcuri. *On the Impact of Tool Evolution and Case Study Size on SBSE Experiments: A Replicated Study with EvoMaster*. Symposium on Search-based Software Engineering (SSBSE). [PDF]
- A. Golmohammadi. *Enhancing White-Box Search-Based Testing of RESTful APIs*. IEEE International Symposium on

- If you want to go into the low-level details of how these techniques work

Future Work

- Improve code/**bytecode analysis** for SBST
  - increase code coverage
- Integrating other AI techniques: **ML** and **LLM**
- Handling whole **Microservice Architectures**
  - ie., not just testing services in isolation
- Support for **Frontend Web GUIs** (eg, actions on browser)

Challenge: Building Usable Research Tools

- Major challenge
- More than 200 000 LOCs
- More than 8 years (since 2016)
- Several people worked on same code-base
- **Many needed engineering tasks lead to no scientific output** (ie. no publications)
  - Difficult to do this kind of work in academia
- A. Arcuri et al. *"Building An Open-Source System Test Generation Tool: Lessons Learned And Empirical Analyses with EvoMaster"*. SQJ'23

Concluding Remarks Before Demo

- Many success stories about fuzzing / test-generation
- REST fuzzing (and partially GraphQL and RPC) is getting momentum
- **For practitioners**:
  - several open-source tools are available, to try out, already today
- **For researchers**:
  - plenty of research challenges to address
  - test generators can be used in other SE research topics

Demo: EvoMaster