

# Cloud Service API Fuzzing

---

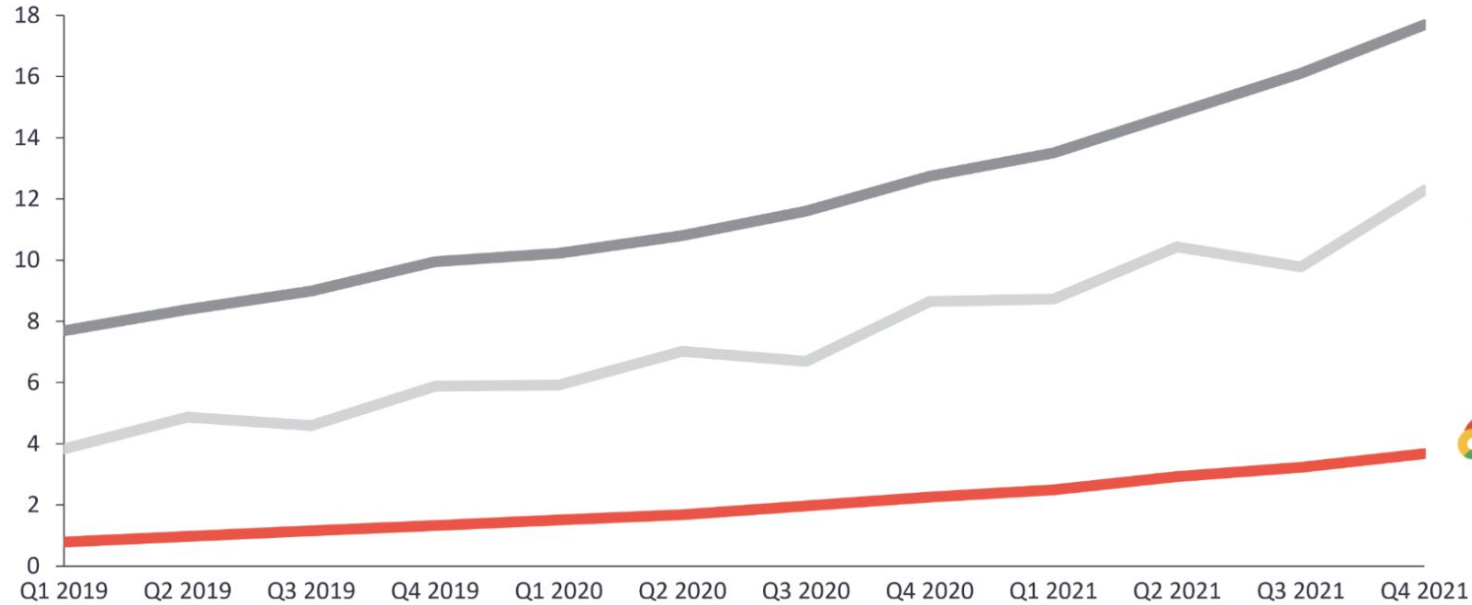
Vaggelis Atlidakis



# Structure and Feedback in Cloud Service API Fuzzing

## Revenue of leading cloud vendors (2019-21)

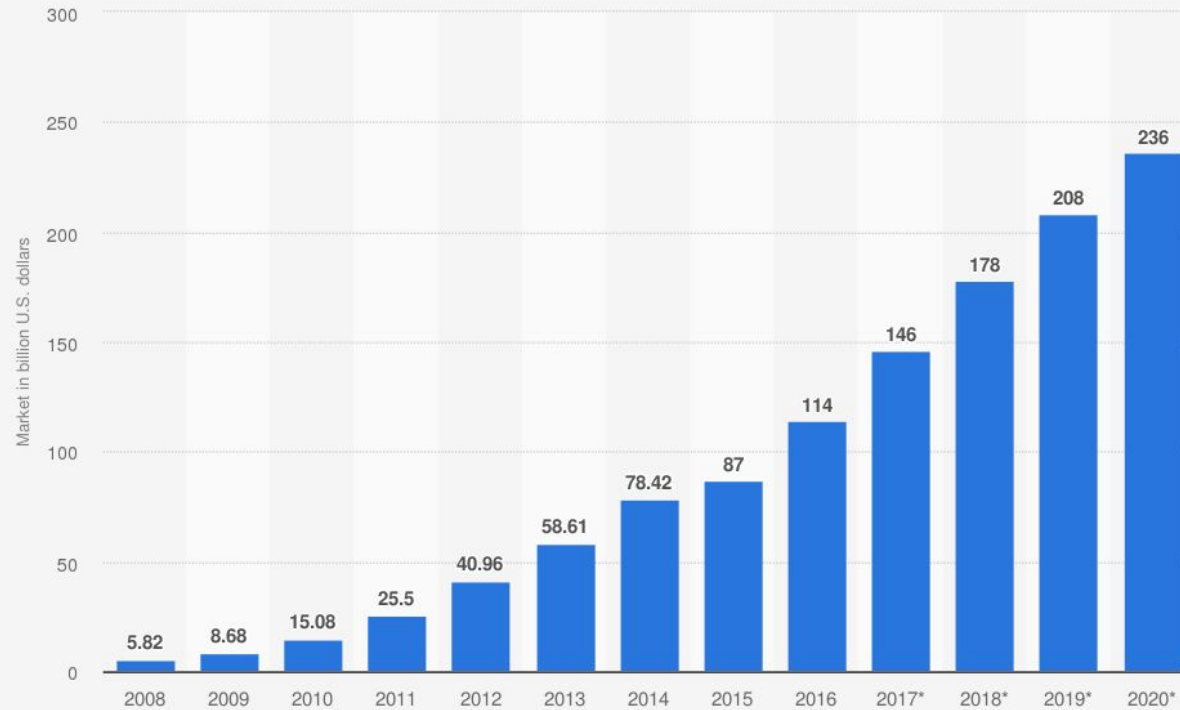
Quarterly cloud revenue in \$B (IaaS, PaaS, and Others)



[1] <https://iot-analytics.com/cloud-market/>

# Structure and Feedback in Cloud Service API Fuzzing

Total size of the public cloud computing market from 2008 to 2020 (in billion U.S. dollars)



**Sources**

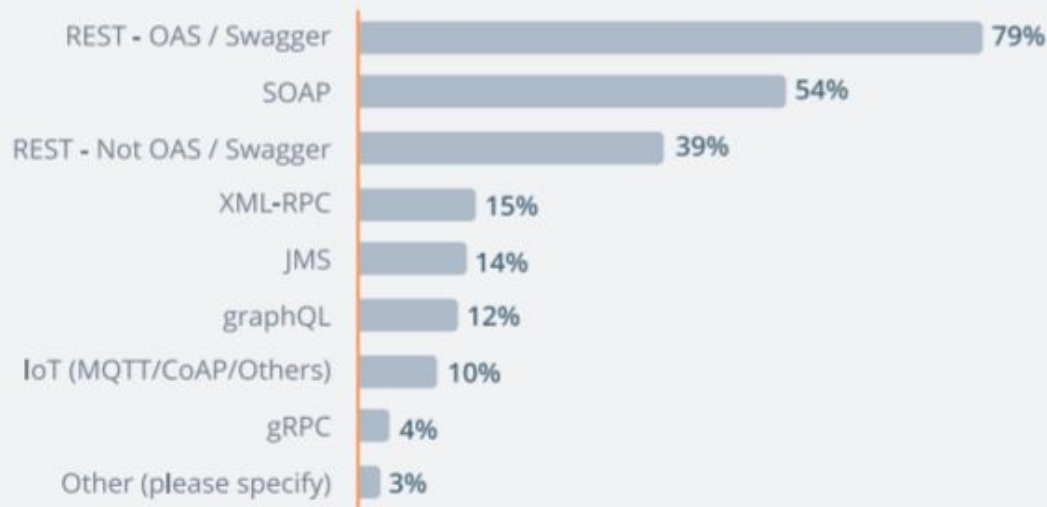
Forrester Research; Forbes; Tata Communications  
© Statista 2019

**Additional Information:**

Worldwide; 2008 to 2016

# Structure and Feedback in Cloud Service API Fuzzing

Which of the following API / Web Services formats do you use? *(Select all that apply)*



## Preface:

This survey was designed to establish benchmarks for the API industry regarding the methodologies, practices, and tools used by software teams to plan, design, develop, test, document, and monitor APIs

## Methodology:

At SmartBear, we conducted a global online 52-question survey over the course of two months from November to December 2018 and collected a total of 3,372 responses. The primary audience for the survey were users of the open source, free, and commercial versions of the Swagger, SoapUI, and ReadyAPI

## Testing REST APIs

## Testing REST APIs

- ❖ Grammar-based fuzzing
  - Producing grammar requires manual effort
  - No coverage feedback (How much fuzzing is enough?)

## Testing REST APIs

- ❖ Grammar-based fuzzing
  - Producing grammar requires manual effort
  - No coverage feedback (How much fuzzing is enough?)
- ❖ HTTP fuzzers
  - Requires live traffic
  - Not Stateful (cannot reproduce sequences of events)

## Testing REST APIs

- ❖ Grammar-based fuzzing
  - Producing grammar requires manual effort
  - No coverage feedback (How much fuzzing is enough?)
- ❖ HTTP fuzzers
  - Requires live traffic
  - Not Stateful (cannot reproduce sequences of events)
- ❖ Custom tools for specific APIs
  - Labour intensive
  - High maintenance cost (APIs evolve over time)

## Challenge

- Testing individual API endpoints will miss errors that require sequences of events in order to be exposed

## Challenge

- Testing individual API endpoints will miss errors that require sequences of events in order to be exposed

## RESTler: Stateful REST API Fuzzing

- Dependency analysis between API requests
- Dynamic feedback loop that learns from past tests

## Challenge

- Testing individual API endpoints will miss errors that require sequences of events in order to be exposed

## RESTler: Stateful REST API Fuzzing

- Dependency analysis between API requests
- Dynamic feedback loop that learns from past tests

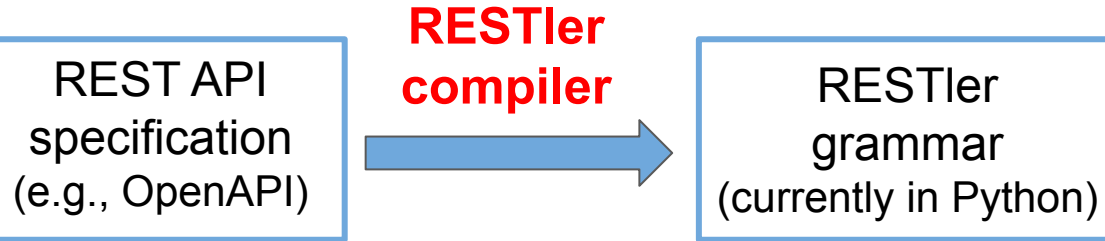
## Kinds of errors RESTler can find

- “500 Internal Server Error” (unhandled exceptions)

## System overview

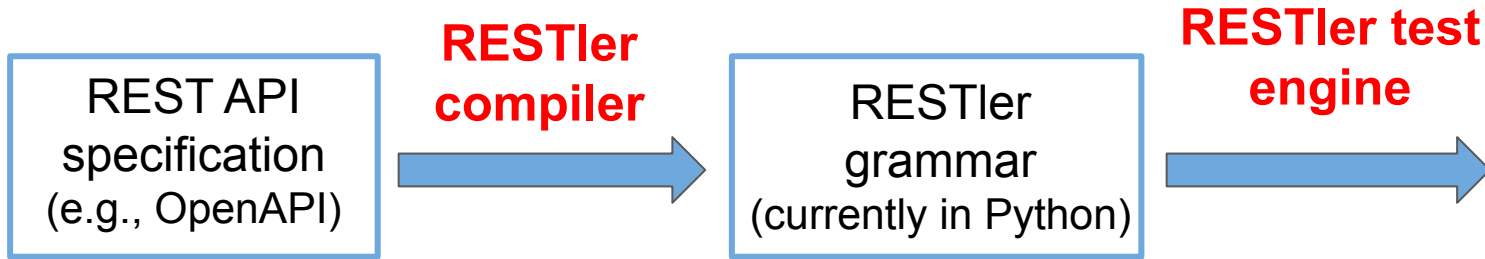
REST API  
specification  
(e.g., OpenAPI)

## System overview



- ❖ Describe how to fuzz each request type
- ❖ Identify producer/consumer dependencies
- ❖ Generate code to parse responses

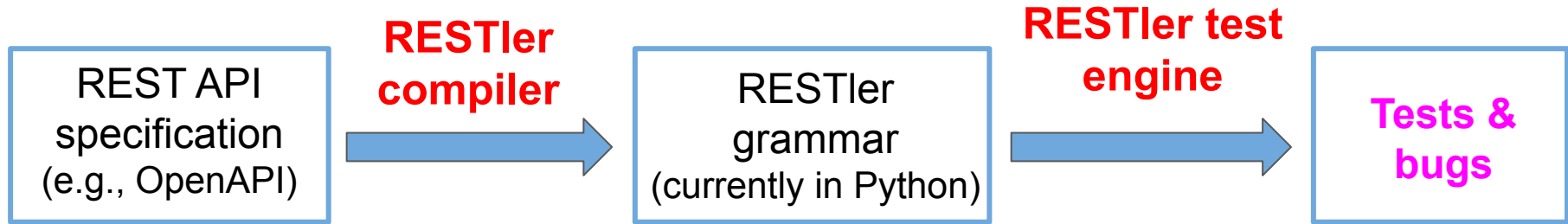
## System overview



- ❖ Describe how to fuzz each request type
- ❖ Identify producer/consumer dependencies
- ❖ Generate code to parse responses

- ❖ Generate and execute tests: sequences of requests
- ❖ Systematic state-space exploration (breadth first search and others)
- ❖ Analyze test results: Dynamic feedback loop learns from service responses of past tests`

## System overview



- ❖ Describe how to fuzz each request type
- ❖ Identify producer/consumer dependencies
- ❖ Generate code to parse responses

- ❖ Generate and execute tests: sequences of requests
- ❖ Systematic state-space exploration (breadth first search and others)
- ❖ Analyze test results: Dynamic feedback loop learns from service responses of past tests`

## Example

POST	/blog/posts	Creates a new blog post
Response Class (Status 200)		
Success		
Model	Model Schema	
<b>Blog post public {</b>		
<b>body</b> (string): Article content,		
}		
GET	/blog/posts	Returns a list of blog posts
DELETE	/blog/posts/{postId}	Deletes a blog post
GET	/blog/posts/{postId}	Returns a blog post
PUT	/blog/posts/{postId}	Updates a blog post

Sample Swagger specification

## Example

POST	/blog/posts	Creates a new blog post
Response Class (Status 200) Success		
Model	Model Schema	
<b>Blog post public {</b> <b>body</b> (string): Article content, }		
GET	/blog/posts	Returns a list of blog posts
DELETE	/blog/posts/{postId}	Deletes a blog post
GET	/blog/posts/{postId}	Returns a blog post
PUT	/blog/posts/{postId}	Updates a blog post

Sample Swagger specification

```
fuzzable["string"] = {"SampleString", "-"}}
```

```
def parse_posts(data):  
  post_id = data["id"]  
  dependencies.set_var(post_id)
```

```
request = requests.Request(  
  restler_static("POST"),  
  restler_static("/api/blog/posts/"),  
  restler_static("HTTP/1.1"),  
  restler_static("{}"),  
  restler_static("body:"),  
  restler_fuzzable("string"),  
  restler_static("{}"),  
  'post_send': {  
    'parser': parse_posts,  
    'dependencies': [  
      post_id.writer(),  
    ]  
  }  
}
```

RESTler grammar fragment

# Structure and Feedback in Cloud Service API Fuzzing

## Example

POST	/blog/posts	Creates a new blog post
Response Class (Status 200) Success		
Model	Model Schema	
<b>Blog post public {</b> body (string): Article content, }		
GET	/blog/posts	Returns a list of blog posts
DELETE	/blog/posts/{postId}	Deletes a blog post
GET	/blog/posts/{postId}	Returns a blog post
PUT	/blog/posts/{postId}	Updates a blog post

Sample Swagger specification

```
fuzzable["string"] = {"SampleString", "_"}

def parse_posts(data):
    post_id = data["id"]
    dependencies.set_var(post_id)

request = requests.Request(
    restler_static("POST"),
    restler_static("/api/blog/posts/"),
    restler_static("HTTP/1.1"),
    restler_static("{}"),
    restler_static("body:"),
    restler_fuzzable("string"),
    restler_static("{}"),
    'post_send': {
        'parser': parse_posts,
        'dependencies': [
            post_id.writer(),
        ]
    }
}
```

RESTler grammar fragment

```
Send: POST /api/blog/posts/ HTTP/1.1
Accept: application/json
Content-Type: application/json
{"body": "sampleString"}

Recv: HTTP/1.1 201 CREATED
Content-Type: application/json
Server: Werkzeug/0.14.1 Python/2.7.12
{"body": "sampleString", "id": 5889}
```

```
Send: POST /api/blog/posts/ HTTP/1.1
Accept: application/json
Content-Type: application/json
{"body": ""}

Recv: HTTP/1.1 400 Bad Request
Content-Type: application/json
Server: Werkzeug/0.14.1 Python/2.7.12
{"error": "body is empty"}
```

Sample tests

## Test generation

POST	/blog/posts	Creates a new blog post
Response Class (Status 200)		
Success		
Model	Model Schema	
<b>Blog post public {</b>		
<b>body</b> (string): Article content,		
}		
GET	/blog/posts	Returns a list of blog posts
DELETE	/blog/posts/{postId}	Deletes a blog post
GET	/blog/posts/{postId}	Returns a blog post
PUT	/blog/posts/{postId}	Updates a blog post

## Sample Swagger specification

## Test generation

API Requests = {  $A_2^{\text{prod: post-id}}$ ,  $B_1^{\text{none}}$ ,  $C_1^{\text{cons: post-id}}$  }

POST	/blog/posts	Creates a new blog post
Response Class (Status 200)		
Success		
Model	Model Schema	
<b>Blog post public {</b>		
<b>body</b> (string): Article content,		
}		
GET	/blog/posts	Returns a list of blog posts
DELETE	/blog/posts/{postId}	Deletes a blog post
GET	/blog/posts/{postId}	Returns a blog post
PUT	/blog/posts/{postId}	Updates a blog post

## Sample Swagger specification

## Test generation

API Requests =  $\{ A_2^{\text{prod: post-id}}, B_1^{\text{none}}, C_1^{\text{cons: post-id}} \}$

Tests (Gen-0) =  $\{ \emptyset \}$

POST	/blog/posts	Creates a new blog post
Response Class (Status 200)		
Success		
Model	Model Schema	
<b>Blog post public {</b>		
<b>body</b> (string): Article content,		
}		
GET	/blog/posts	Returns a list of blog posts
DELETE	/blog/posts/{postId}	Deletes a blog post
GET	/blog/posts/{postId}	Returns a blog post
PUT	/blog/posts/{postId}	Updates a blog post

## Sample Swagger specification

## Test generation

POST	/blog/posts	Creates a new blog post
Response Class (Status 200)		
Success		
Model	Model Schema	
<b>Blog post public {</b>		
<b>body</b> (string): Article content,		
}		
GET	/blog/posts	Returns a list of blog posts
DELETE	/blog/posts/{postId}	Deletes a blog post
GET	/blog/posts/{postId}	Returns a blog post
PUT	/blog/posts/{postId}	Updates a blog post

API Requests =  $\{ A_2^{\text{prod: post-id}}, B_1^{\text{none}}, C_1^{\text{cons: post-id}} \}$

Tests (Gen-0) =  $\{ \emptyset \}$

Tests (Gen-1) =  $\{ A_{1/2}^{\text{prod: post-id}}, A_{2/2}^{\text{prod: post-id}}, B_{1/1}^{\text{none}} \}$

## Sample Swagger specification

## Test generation

POST	/blog/posts	Creates a new blog post
Response Class (Status 200)		
Success		
Model	Model Schema	
<b>Blog post public {</b>		
body (string): Article content,		
}		
GET	/blog/posts	Returns a list of blog posts
DELETE	/blog/posts/{postId}	Deletes a blog post
GET	/blog/posts/{postId}	Returns a blog post
PUT	/blog/posts/{postId}	Updates a blog post

## Sample Swagger specification

API Requests =  $\{ A_2^{\text{prod: post-id}}, B_1^{\text{none}}, C_1^{\text{cons: post-id}} \}$

Tests (Gen-0) =  $\{ \emptyset \}$

Tests (Gen-1) =  $\{ A_{1/2}^{\text{prod: post-id}}, A_{2/2}^{\text{prod: post-id}}, B_{1/1}^{\text{none}} \}$

Tests (Gen-2) =  $\{ A_{1/2}^{\text{prod: post-id}}, A_{1/2}^{\text{prod: post-id}},$

$A_{1/2}^{\text{prod: post-id}}, A_{2/2}^{\text{prod: post-id}}, A_{1/2}^{\text{prod: post-id}}, B_{1/1}^{\text{none}},$

$A_{1/2}^{\text{prod: post-id}}, C_{1/1}^{\text{cons: post-id}}, \dots, B_{1/1}^{\text{none}}, B_{1/1}^{\text{none}} \}$

## Test generation

POST	/blog/posts	Creates a new blog post
Response Class (Status 200)		
Success		
Model	Model Schema	
<b>Blog post public {</b>		
body (string): Article content,		
}		
GET	/blog/posts	Returns a list of blog posts
DELETE	/blog/posts/{postId}	Deletes a blog post
GET	/blog/posts/{postId}	Returns a blog post
PUT	/blog/posts/{postId}	Updates a blog post

## Sample Swagger specification

API Requests =  $\{ A_2^{\text{prod: post-id}}, B_1^{\text{none}}, C_1^{\text{cons: post-id}} \}$

Tests (Gen-0) =  $\{ \emptyset \}$

Tests (Gen-1) =  $\{ A_{1/2}^{\text{prod: post-id}}, A_{2/2}^{\text{prod: post-id}}, B_{1/1}^{\text{none}} \}$

Tests (Gen-2) =  $\{ A_{1/2}^{\text{prod: post-id}}, A_{1/2}^{\text{prod: post-id}},$

$A_{1/2}^{\text{prod: post-id}}, A_{2/2}^{\text{prod: post-id}}, A_{1/2}^{\text{prod: post-id}}, B_{1/1}^{\text{none}},$

$A_{1/2}^{\text{prod: post-id}}, C_{1/1}^{\text{cons: post-id}}, \dots, B_{1/1}^{\text{none}}, B_{1/1}^{\text{none}} \}$

➤ BFS total tests: **13**

## Selected evaluation results

- Q1: Are tests generated by RESTler exercising deeper service-side code over time?
- Q2: Can RESTler find bugs in production-scale cloud services?

## Selected evaluation results

- Q1: Are tests generated by RESTler exercising deeper service-side code over time?
- Q2: Can RESTler find bugs in production-scale cloud services?

Study subjects: Gitlab, Spree, and Mastodon

- ❖ Open-source production-scale services
- ❖ Self-hosted git, e-commerce, and social networking
- ❖ Complex REST APIs & millions of users

## Deeper service exploration (Q1)

API Family	Total API Requests	Seq. Len.	Code Coverage Increase (lines of code)	Tests
Gitlab Commits	15 (*11)	1	598	1
		2	1108	7
		3	1196	250
		4	1760	2220
		5	1760	3667
Spree Cart	8 (*11)	1	10	1
		2	208	2
		3	1473	47
		4	1943	6380
Mastodon Statuses	18 (*19)	1	4	60
		2	333	5908
		3	631	28926

❖ Longer sequences increase service-side code coverage

## Deeper service exploration (Q1)

API Family	Total API Requests	Seq. Len.	Code Coverage Increase (lines of code)	Tests
Gitlab Commits	15 (*11)	1	598	1
		2	1108	7
		3	1196	250
		4	1760	2220
		5	1760	3667
Spree Cart	8 (*11)	1	10	1
		2	208	2
		3	1473	47
		4	1943	6380
Mastodon Statuses	18 (*19)	1	4	60
		2	333	5908
		3	631	28926

- ❖ Longer sequences increase service-side code coverage
- ❖ Sequences of at least three requests
- ❖ Progress in large search space

## Deeper service exploration (Q1)

API Family	Total API Requests	Seq. Len.	Code Coverage Increase (lines of code)	Tests
Gitlab Commits	15 (*11)	1	598	1
		2	1108	7
		3	1196	250
		4	1760	2220
		5	1760	3667
Spree Cart	8 (*11)	1	10	1
		2	208	2
		3	1473	47
		4	1943	6380
Mastodon Statuses	18 (*19)	1	4	60
		2	333	5908
		3	631	28926

❖ Longer sequences increase service-side code coverage

❖ Sequences of at least three requests

❖ Progress in large search space

Example: Testing Commits API (5 hours)

➤ **Brute-force**: 15 requests / 11 different payloads; that is:  $(15*11)^3 = 4.4$  million sequences of length three

## Deeper service exploration (Q1)

API Family	Total API Requests	Seq. Len.	Code Coverage Increase (lines of code)	Tests
Gitlab Commits	15 (*11)	1	598	1
		2	1108	7
		3	1196	250
		4	1760	2220
		5	1760	3667
Spree Cart	8 (*11)	1	10	1
		2	208	2
		3	1473	47
		4	1943	6380
Mastodon Statuses	18 (*19)	1	4	60
		2	333	5908
		3	631	28926

- ❖ Longer sequences increase service-side code coverage

- ❖ Sequences of at least three requests

- ❖ Progress in large search space

Example: Testing Commits API (5 hours)

- **Brute-force:** 15 requests / 11 different payloads; that is:  $(15*11)^3 = 4.4$  million sequences of length three

- **RESTler:** Sequence length three: 250 tests generated (request dependencies + service feedback)

## New bugs found with RESTler (Q2)

API	BFS	BFS-Fast	RandomWalk	$\cap$	U
Commits	5	1	5	1	5
Cart	1	1	1	1	1
Statuses	1	1	0	0	1
...	...	...	...	...	...
<b>Total</b>	<b>18</b>	<b>15</b>	<b>22</b>	<b>10</b>	<b>24</b>

Bug buckets per search strategy

- ❖ BFS-Fast finds least bugs
- ❖ RandomWalk finds most bugs
- ❖ All 24 bugs are easily reproducible, confirmed, and fixed

## New bugs found with RESTler (Q2)

API	BFS	BFS-Fast	RandomWalk	$\cap$	U
Commits	5	1	5	1	5
Cart	1	1	1	1	1
Statuses	1	1	0	0	1
...	...	...	...	...	...
<b>Total</b>	<b>18</b>	<b>15</b>	<b>22</b>	<b>10</b>	<b>24</b>

Bug buckets per search strategy

- ❖ BFS-Fast finds least bugs
- ❖ RandomWalk finds most bugs
- ❖ All 24 bugs are easily reproducible, confirmed, and fixed

### Example Bug [#50268]

1. Create a gitlab project
  2. Create a file with a proper commit message
  3. Delete the file with an empty commit message
- “500 Internal Server Error”

## Main take-aways (so far)

- ✓ Introduced stateful REST API fuzzing
  - 500 “Internal Server Error” (Unhandled exceptions)

## Contributions

- RESTler: Stateful REST API Fuzzing (ICSE'19)
- **Checking Security Properties of Cloud Service REST APIs (ICST'20)**
- Pythia: Grammar-Based Fuzzing of REST APIs with Coverage-guided Feedback and Learning-based Mutations (target submission ICSE'21)

## Challenge

- How can we uncover errors that do not cause visible 500s?

## Challenge

- How can we uncover errors that do not cause visible 500s?

## Checking security properties

- Introduce rules that capture desirable security properties of cloud service REST APIs
- Augment stateful REST API fuzzing with checkers that test violation of these rules

## Challenge

- How can we uncover errors that do not cause visible 500s?

## Checking security properties

- Introduce rules that capture desirable security properties of cloud service REST APIs
- Augment stateful REST API fuzzing with checkers that test violation of these rules

## Kinds of error

- Violations of security property rules

## Selected security rules and desirable properties

- ❖ Use-after-free rule
  1. Delete /api/projects/1
  2. Access /api/projects/1 **MUST FAIL**

## Selected security rules and desirable properties

- ❖ Use-after-free rule
  1. Delete /api/projects/1
  2. Access /api/projects/1 **MUST FAIL**
  
- ❖ Resource-hierarchy rule
  1. Create /api/projects/1 and /api/projects/2
  2. Create /api/projects/1/branches/1
  - Access /api/projects/2/branches/1 **MUST FAIL**

## Selected security rules and desirable properties

### ❖ Use-after-free rule

1. Delete `/api/projects/1`
2. Access `/api/projects/1` **MUST FAIL**

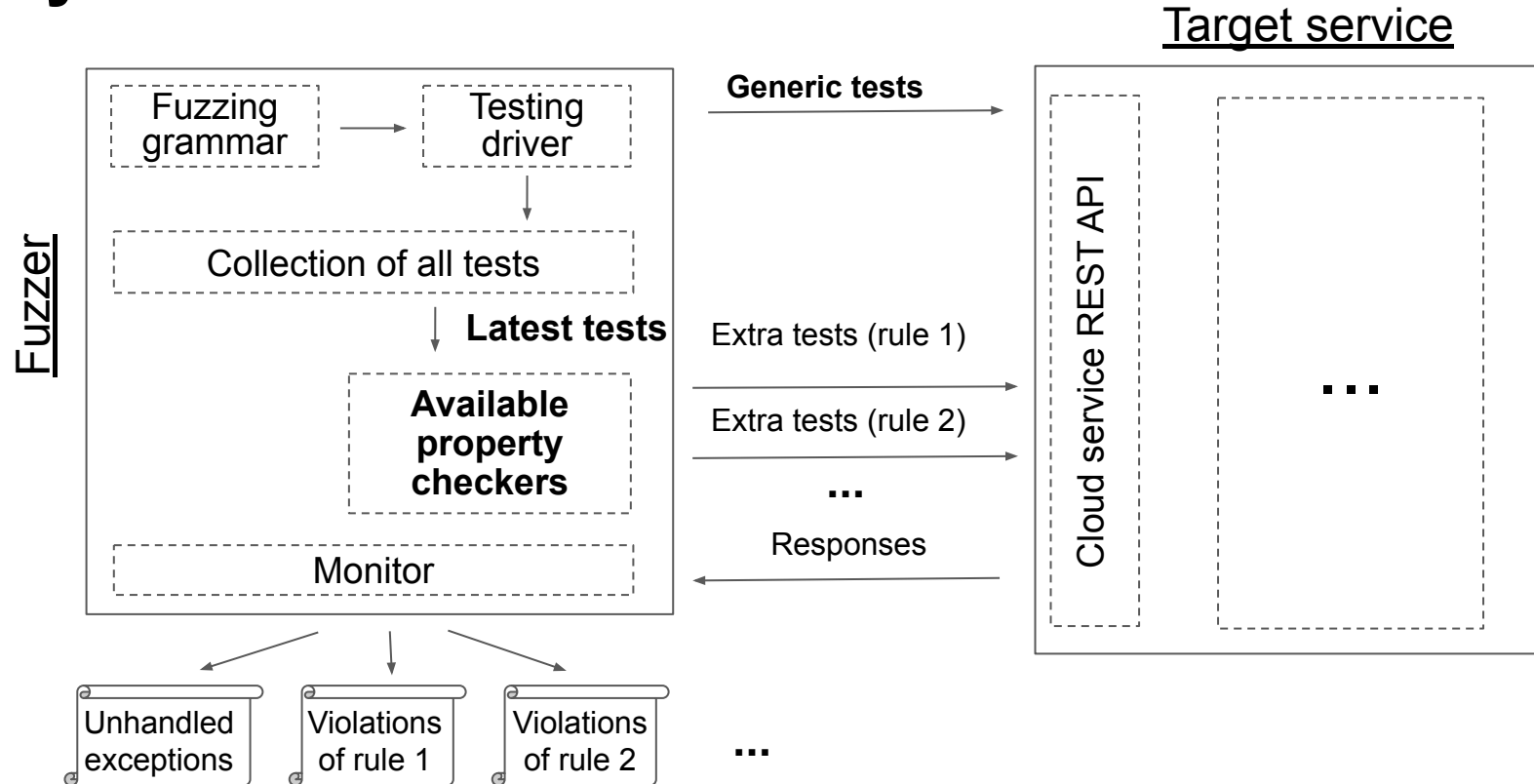
### ❖ Resource-hierarchy rule

1. Create `/api/projects/1` and `/api/projects/2`
  2. Create `/api/projects/1/branches/1`
- Access `/api/projects/2/branches/1` **MUST FAIL**

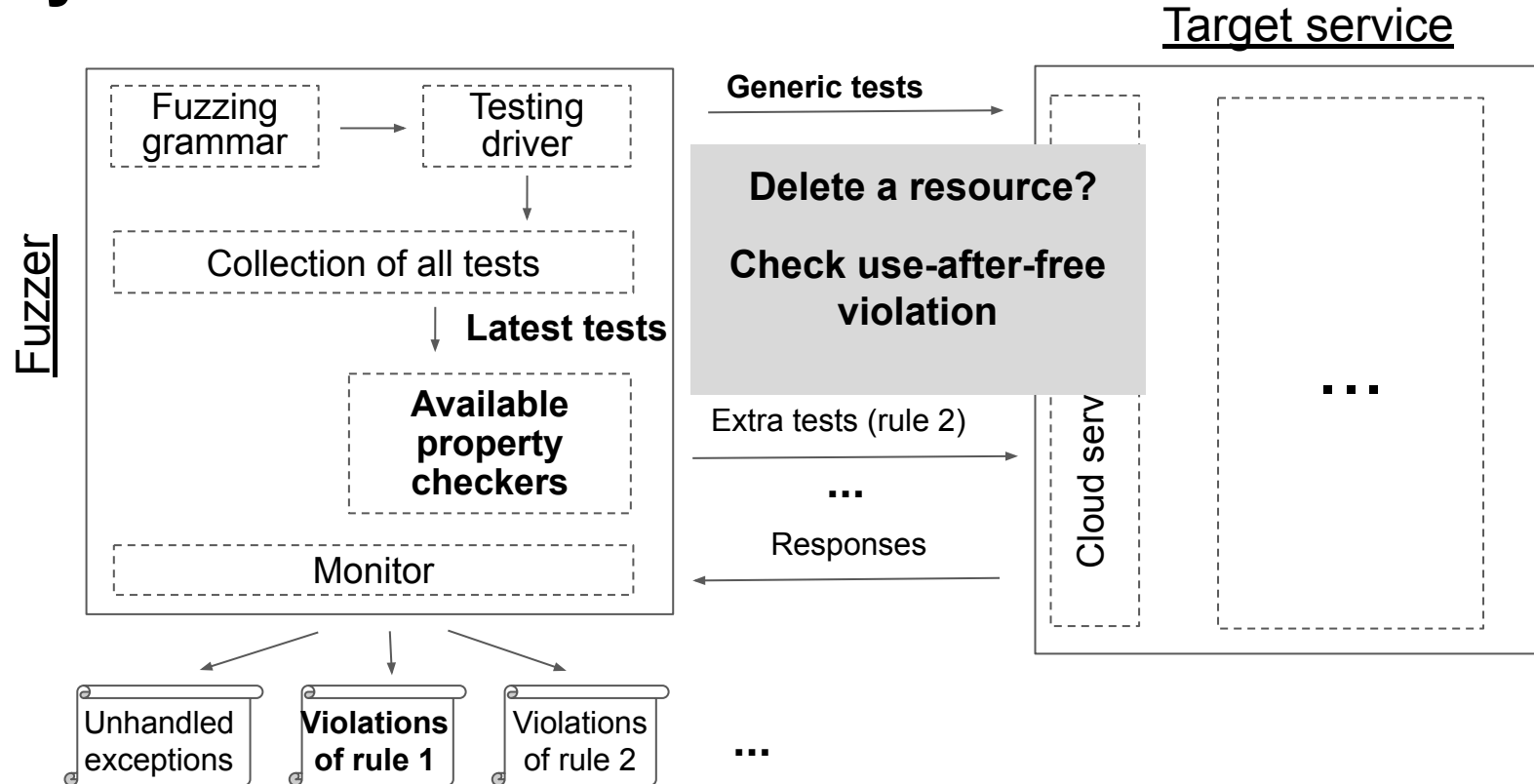
### ❖ Resource-leakage rule

1. Create `/api/projects/1` and receive error code (e.g., 404 or 500 HTTP status)
- Access `/api/projects/1` **MUST FAIL**

## System overview

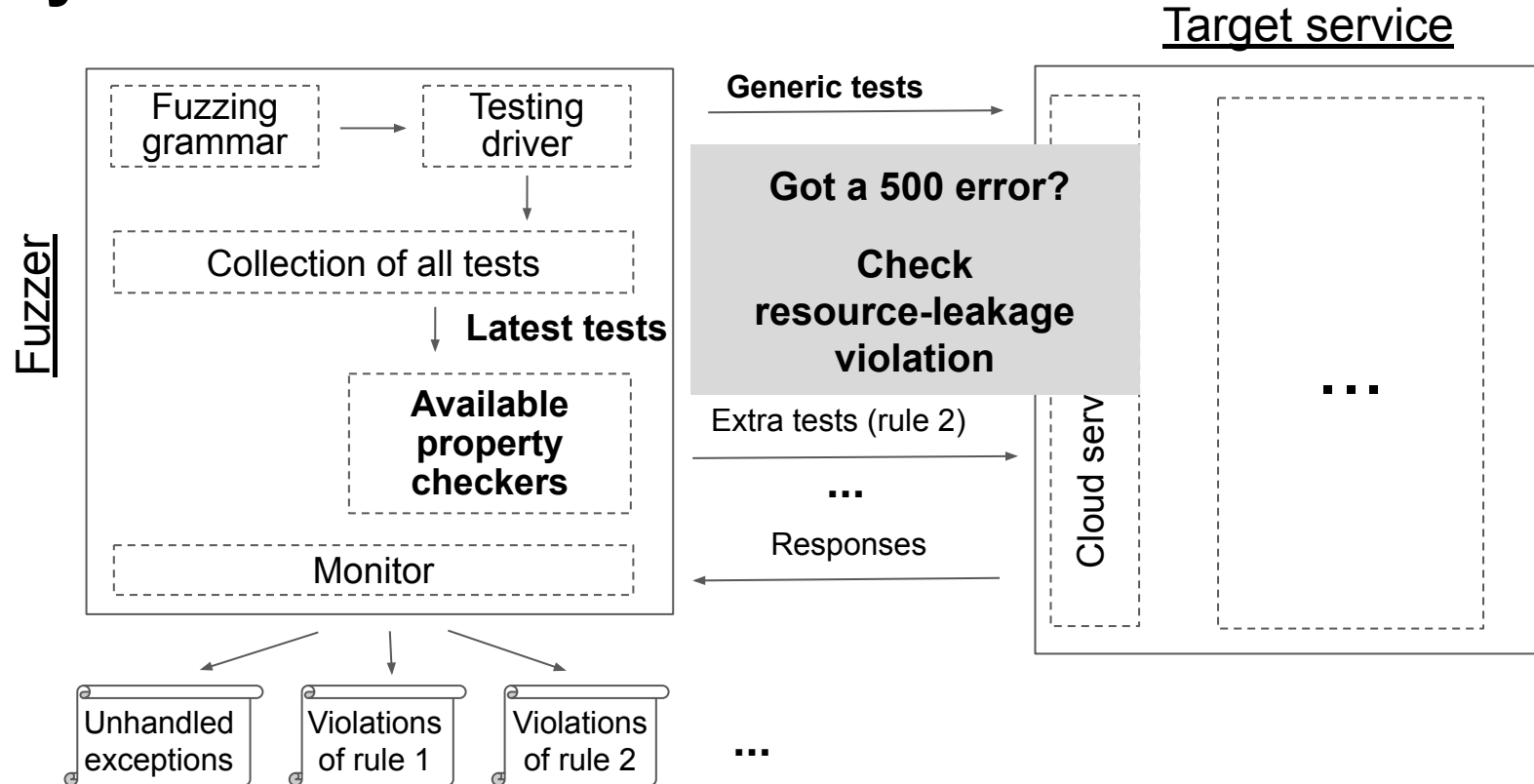


## System overview





## System overview



## Selected errors found with checkers in Azure & O365

### ❖ Use-after-free rule violation

1. Create a new resource R
2. Delete resource R
3. Create a new child resource as the deleted resource R
  - 500 "Internal Server Error" (should have been: 404 Not Found)

## Selected errors found with checkers in Azure & O365

### ❖ Use-after-free rule violation

1. Create a new resource R
2. Delete resource R
3. Create a new child resource as the deleted resource R
  - 500 "Internal Server Error" (should have been: 404 Not Found)

### ❖ Resource-hierarchy rule violation

1. Create two messages (POST /api/posts/1212 and POST /api/posts/1313)
2. Create a reply to the first message (POST /api/posts/1212/replies/12121)
3. Edit the reply using the second message as parent (PUT /api/posts/1313/replies/12121)
  - 202 "Accepted" (should have been: 404 Not Found)

## Selected errors found with checkers in Azure & O365

### ❖ Use-after-free rule violation

1. Create a new resource R
2. Delete resource R
3. Create a new child resource as the deleted resource R
  - 500 "Internal Server Error" (should have been: 404 Not Found)

### ❖ Resource-hierarchy rule violation

1. Create two messages (POST /api/posts/1212 and POST /api/posts/1313)
2. Create a reply to the first message (POST /api/posts/1212/replies/12121)
3. Edit the reply using the second message as parent (PUT /api/posts/1313/replies/12121)
  - 202 "Accepted" (should have been: 404 Not Found)

### ❖ Resource-leakage rule violation

1. Create a resource of type T and name X with malformed body (this results in a 500 error)
2. Get a list of all resource of type T: the returned result is empty
3. Create a new resource of type T with the same name X in different region
  - 409 "Conflict" Inconsistent service state (should have been: 404 Not Found)

## Main take-aways (so far)

- ✓ Introduced stateful REST API fuzzing
- ✓ Extended stateful REST API fuzzing to new classes of errors

## Contributions

- RESTler: Stateful REST API Fuzzing (ICSE'19)
- Checking Security Properties of Cloud Service REST APIs (ICST'20)
- Pythia: Grammar-Based Fuzzing of REST APIs with Coverage-guided Feedback and Learning-based Mutations (target submission ICSE'21)

## Challenge

- Automatically-generated grammars have few, predefined fuzzing values and no code coverage feedback.

## Challenge

- Automatically-generated grammars have few, predefined fuzzing values and no code coverage feedback. **Can we augment the set of possible values?**

## Challenge

- Automatically-generated grammars have few, predefined fuzzing values and no code coverage feedback. **Can we augment the set of possible values?**

## Pythia

- Learn common usage patterns from seed test cases
- Generate learning-based mutation & use code coverage to prioritize test cases

## Challenge

- Automatically-generated grammars have few, predefined fuzzing values and no code coverage feedback. **Can we augment the set of possible values?**

## Pythia

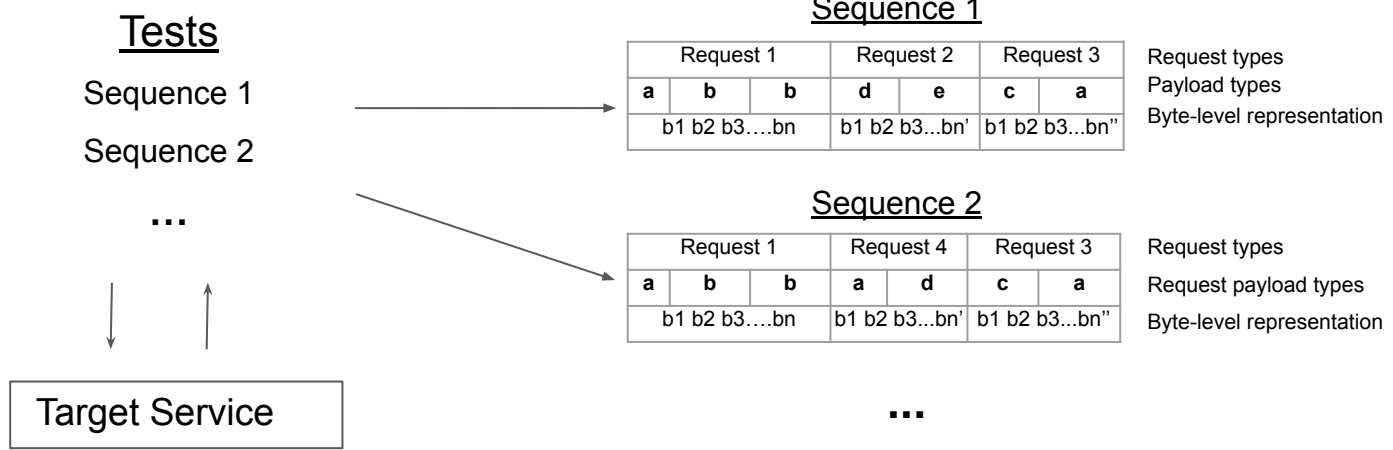
- Learn common usage patterns from seed test cases
- Generate learning-based mutation & use code coverage to prioritize test cases

## Kinds of errors RESTler can find

- “500 Internal Server Error” (unhandled exceptions)

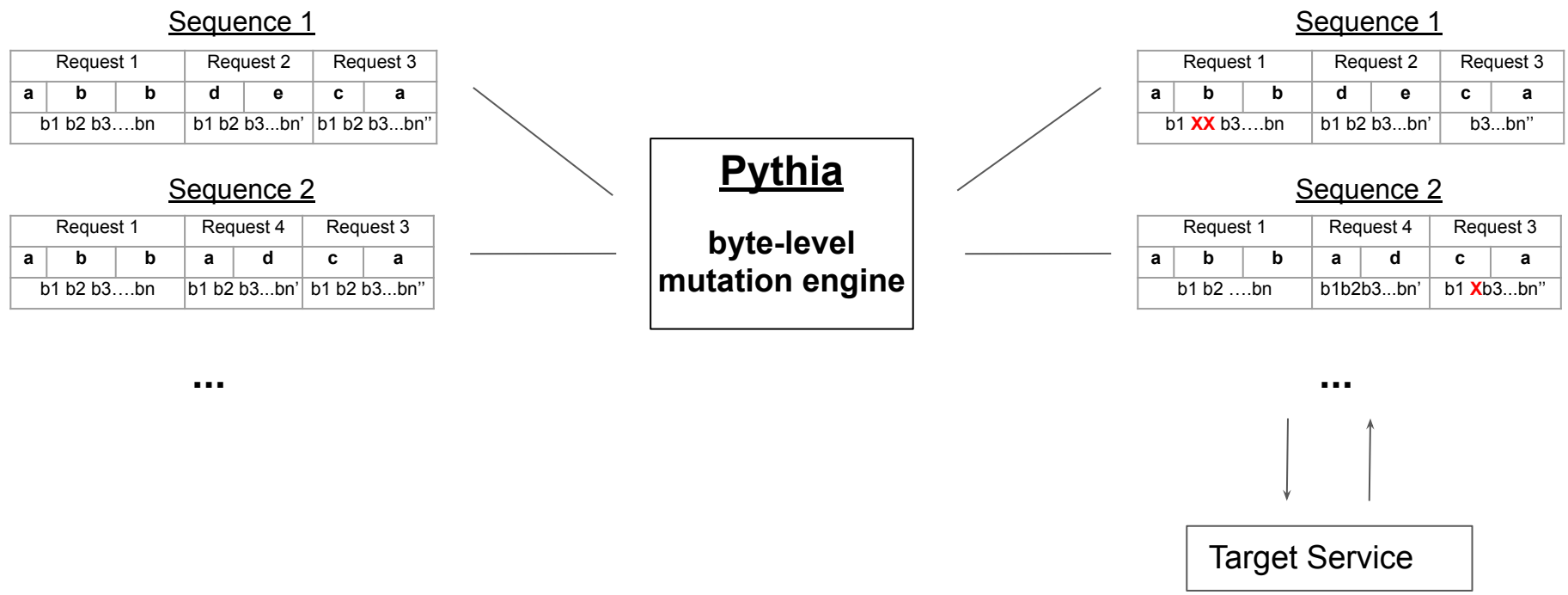
# Structure and Feedback in Cloud Service API Fuzzing

## Example



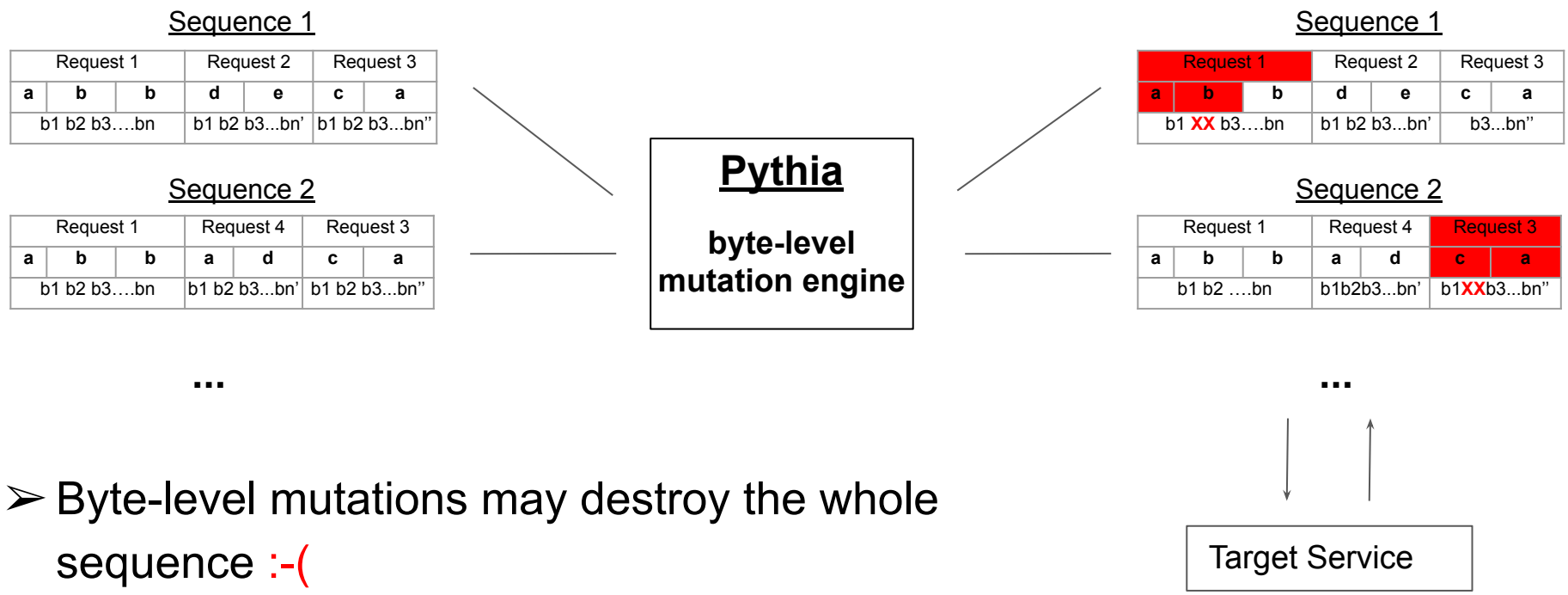
# Structure and Feedback in Cloud Service API Fuzzing

## First approach: Random byte-level mutations



# Structure and Feedback in Cloud Service API Fuzzing

## First approach: Random byte-level mutations



# Structure and Feedback in Cloud Service API Fuzzing

## First approach: Random byte-level mutations

Old rule @382:'E'  
New rule @382:' **lxa4** '

Sent: 'POST /api/v4/projects/997190/repository/branches  
HTTP/1.1\r\n...\r\nPRIVATE-TOKEN: DRiX47nu**lxa4**  
P2ARa4APFr\r\n\r\n{"ref":"master","branch":"string"}\r\n'

Recv: 'HTTP/1.1 401 Unauthorized\r\nDate: Thu, 31 Oct 2019  
06:19:53 GMT...\r\n\r\n{"message":"**401 Unauthorized**"}'

Parsing exception:: **Exception Parsing Response item:** 'name'

### Sequence 1

Request 1			Request 2		Request 3	
a	b	b	d	e	c	a
b1 <b>XX</b> b3...bn			b1 b2 b3...bn'		b3...bn''	

### Sequence 2

Request 1			Request 4		Request 3	
a	b	b	a	d	c	a
b1 b2 ....bn			b1b2b3...bn'		b1 <b>XX</b> b3...bn''	

...



Target Service

- Byte-level mutations may destroy the whole sequence :-('

# Structure and Feedback in Cloud Service API Fuzzing

## Second approach: AST-level mutations

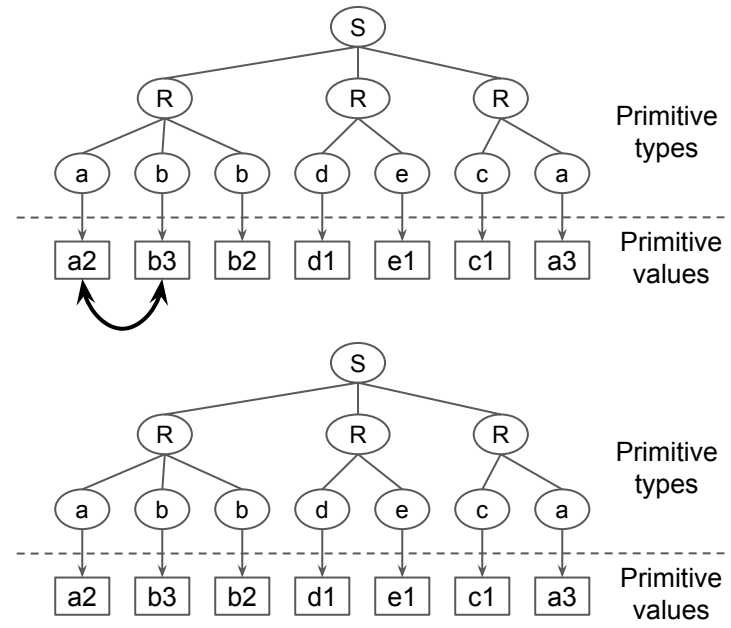
Sequence 1

Request 1	Request 2	Request 3
a   b   b	d   e	c   a
b1 b2 b3...bn	b1 b2 b3...bn'	b1 b2 b3...bn''

Sequence 2

Request 1	Request 4	Request 3
a   b   b	a   d	c   a
b1 b2 b3...bn	b1 b2 b3...bn'	b1 b2 b3...bn''

...



# Structure and Feedback in Cloud Service API Fuzzing

## Second approach: AST-level mutations

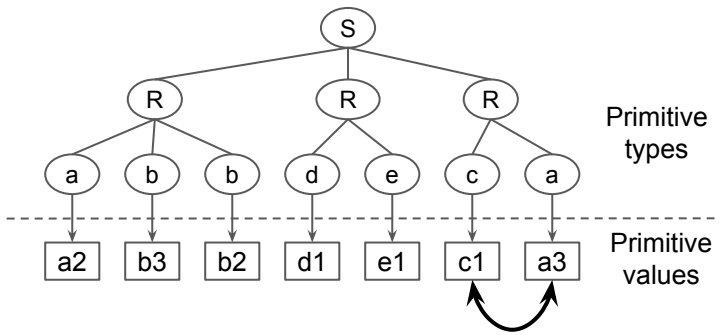
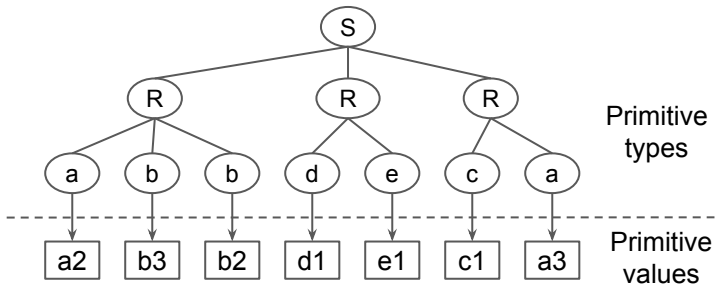
Sequence 1

Request 1			Request 2		Request 3	
a	b	b	d	e	c	a
b1 b2 b3...bn			b1 b2 b3...bn'		b1 b2 b3...bn''	

Sequence 2

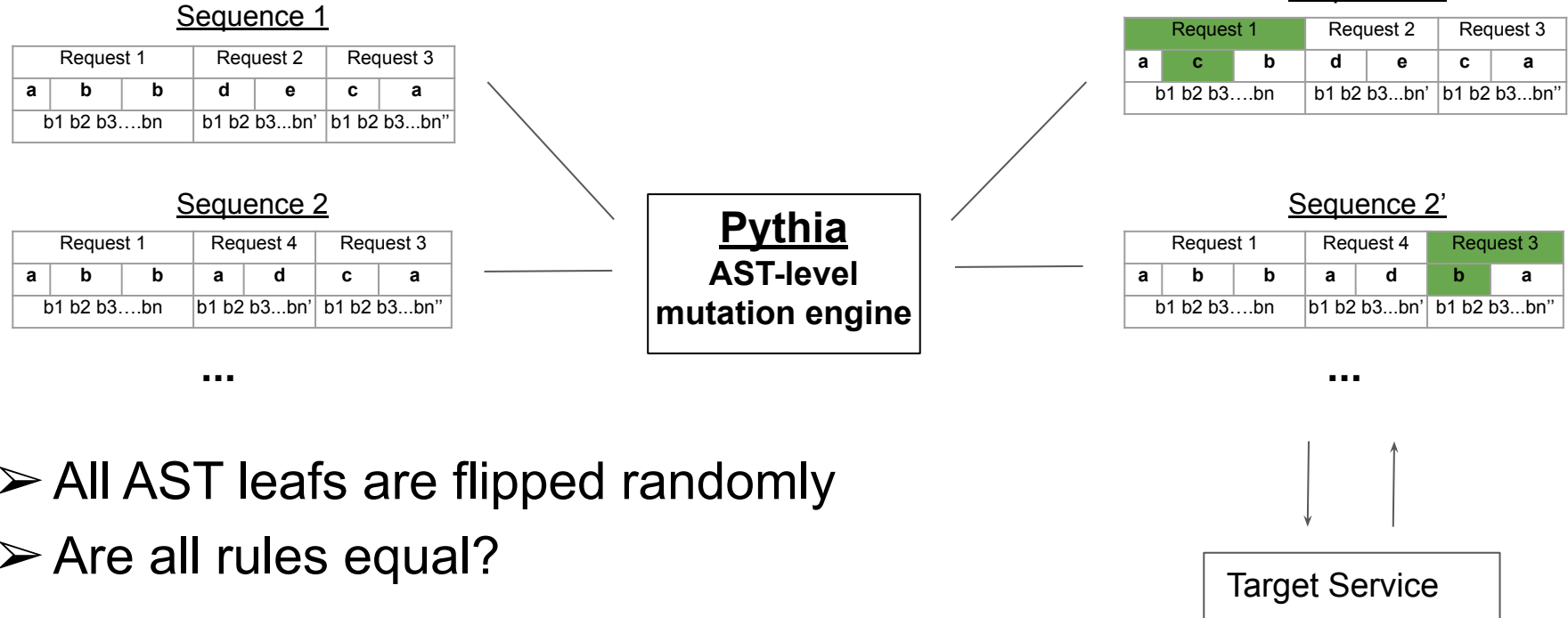
Request 1			Request 4		Request 3	
a	b	b	a	d	c	a
b1 b2 b3...bn			b1 b2 b3...bn'		b1 b2 b3...bn''	

...



# Structure and Feedback in Cloud Service API Fuzzing

## Second approach: AST-level mutations



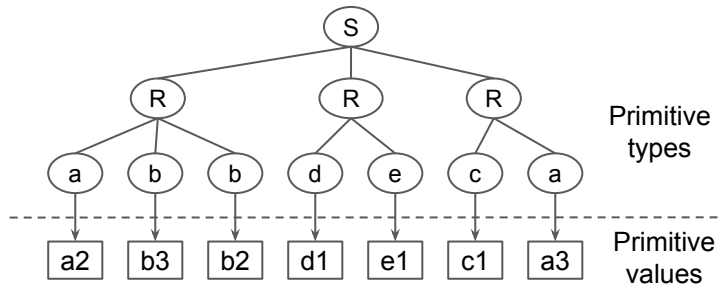
- All AST leafs are flipped randomly
- Are all rules equal?

# Structure and Feedback in Cloud Service API Fuzzing

## Third approach: Probabilistic AST-level mutations

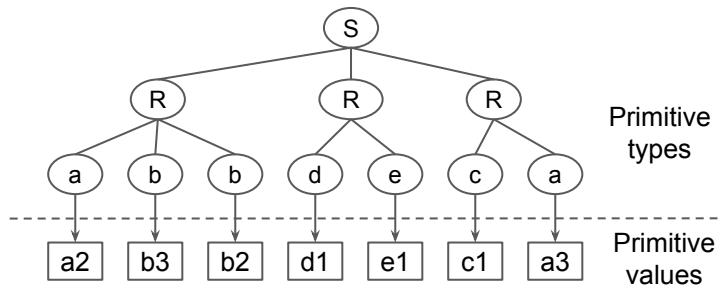
Sequence 1

Request 1	Request 2	Request 3
a   b   b	d   e	c   a
b1 b2 b3...bn	b1 b2 b3...bn'	b1 b2 b3...bn''



Sequence 2

Request 1	Request 4	Request 3
a   b   b	a   d	c   a
b1 b2 b3...bn	b1 b2 b3...bn'	b1 b2 b3...bn''



### Production rules

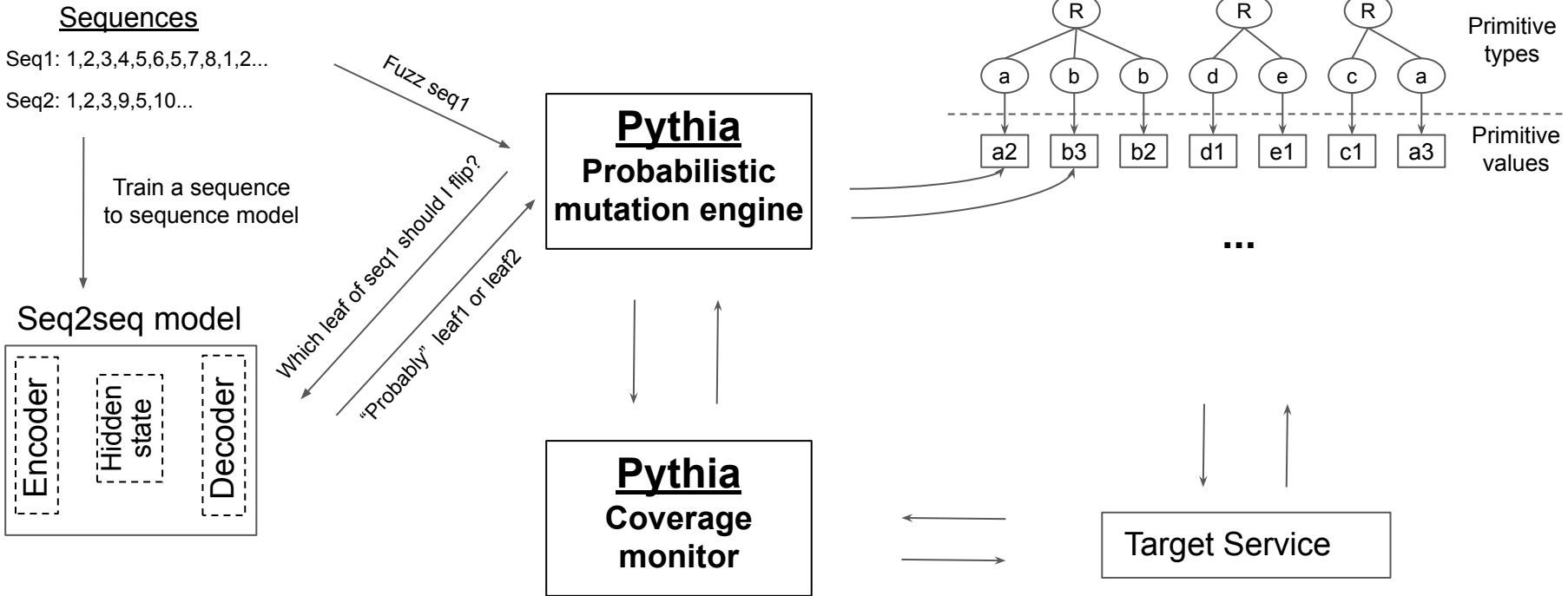
- 1: S → S + R
- 2: R → R + P
- 3: P → P + a
- 4: a → a2
- 5: P → P + b
- 6: b → b3
- 5: P → P + b
- 7: b → b2
- 8: R → e
- 1: S → S + R
- 2: R → R + P
- ...

- 1: S → S + R
- 2: R → R + P
- 3: P → P + a
- 9: a → a2
- 5: P → P + b
- 10: b → b3
- ...

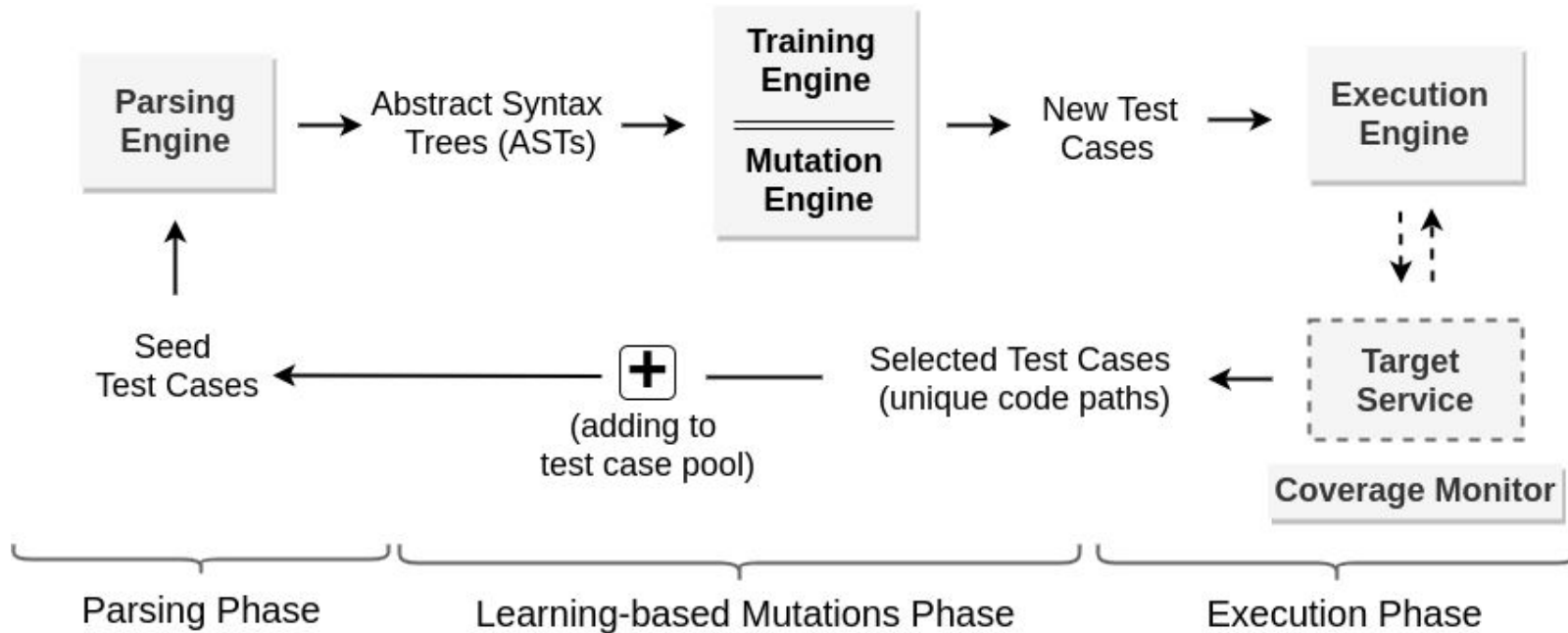
...

# Structure and Feedback in Cloud Service API Fuzzing

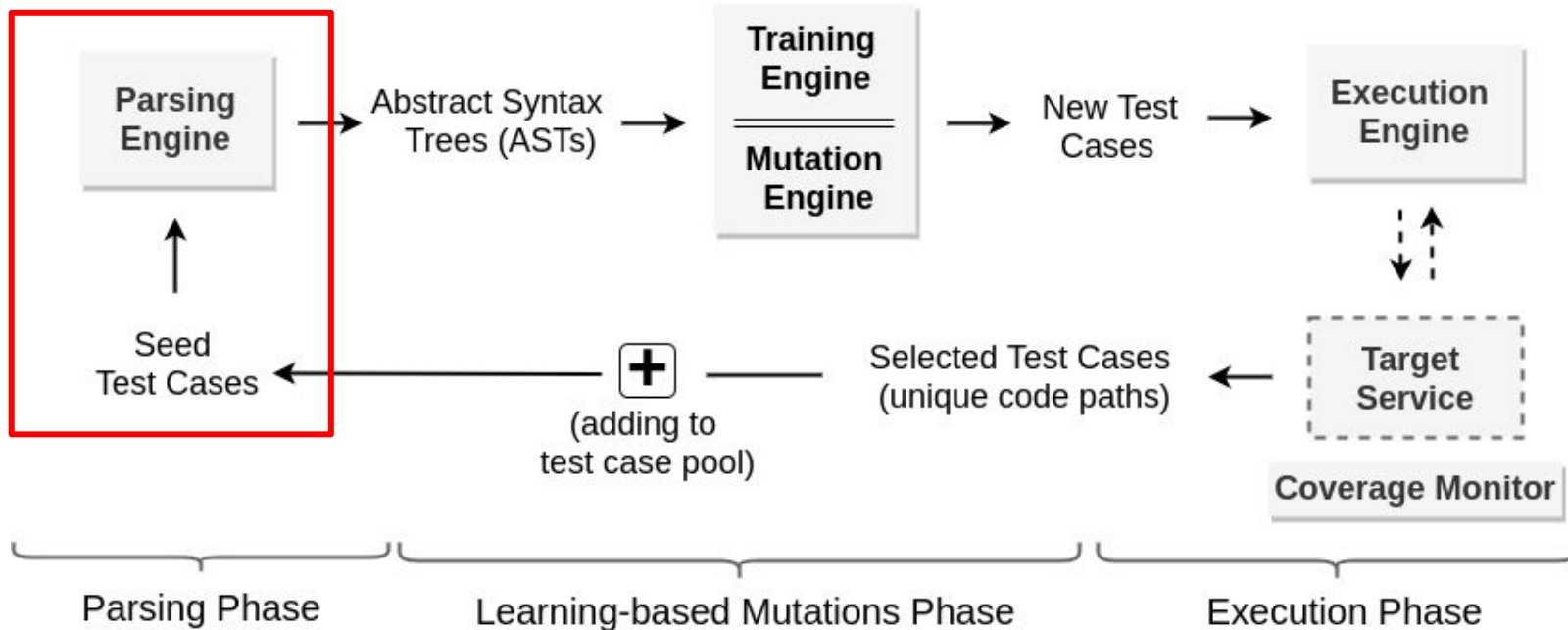
## Third approach: Probabilistic AST-level mutations



## System overview



## System overview



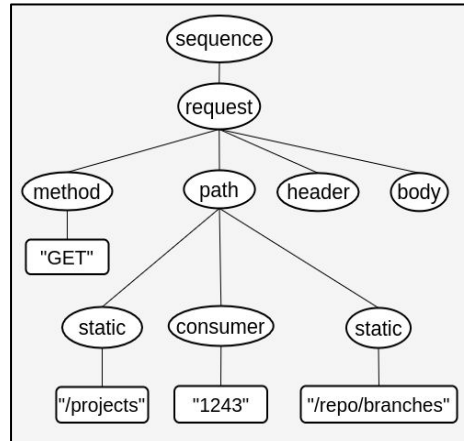
## Parsing

GET /projects/1243/repo/branches

$G = (N, \Sigma, R, S) \downarrow$

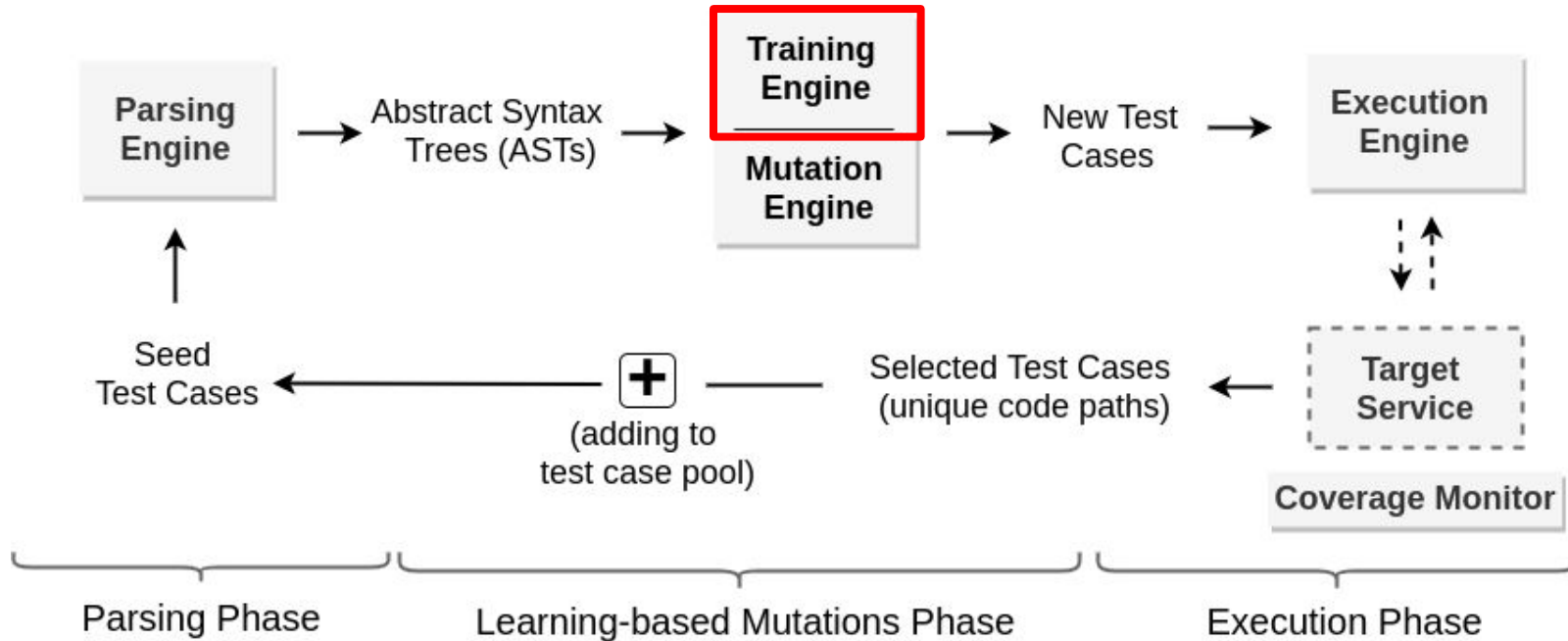
R1: *sequence* -> *request* + *sequence*  
R2: *request* -> *method* + *path*  
          + *header* + *body*  
R3: *method* -> "GET"  
R4: *path* -> *static* + *path*  
R5: *static* -> "/projects"  
R6: *path* -> *consumer* + *path*  
R7: *consumer* -> "1243"  
R8: *path* -> *static* + *path*  
R9: *static* -> "/repo/branches"  
R10: *header* -> *e*  
R11: *body* -> *e*  
R12: *sequence* -> *e*

- ❖ Parse seed test cases with a regular grammar and construct Abstract Syntax Trees (ASTs) from test cases
- ❖ Convert ASTs to sequences (of grammar rules)



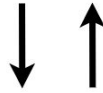
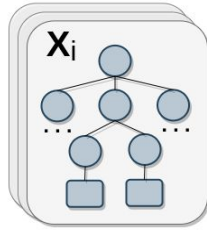
X = < R1,R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12 >

## System overview

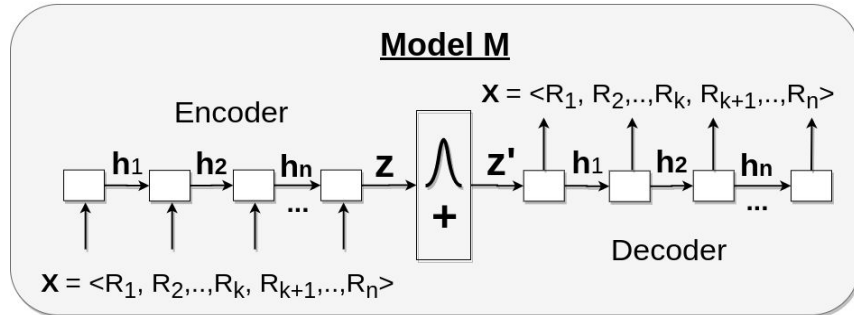


## Training

### Seed Test Cases

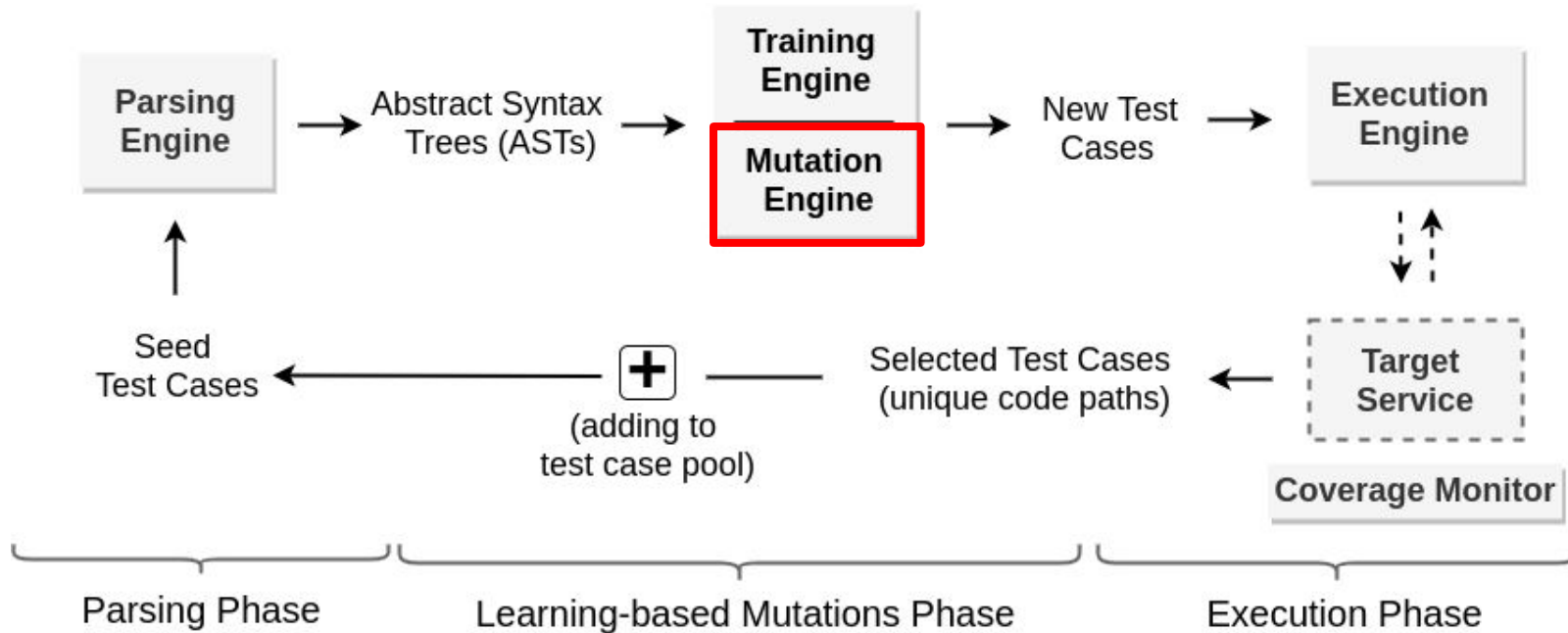


### Model M

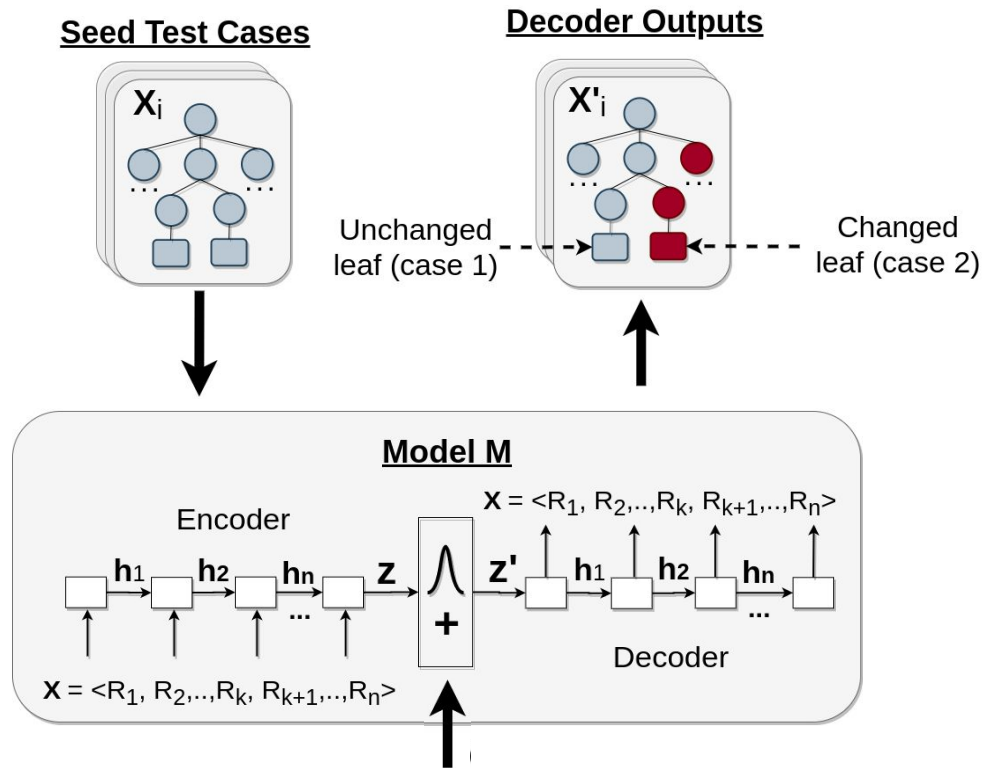


- ❖ Use grammatically valid seed test case sequences
- ❖ Train a seq2seq Recurrent Neural Network (RNN)
- ❖ Learn common usage patterns of a target cloud service

## System overview

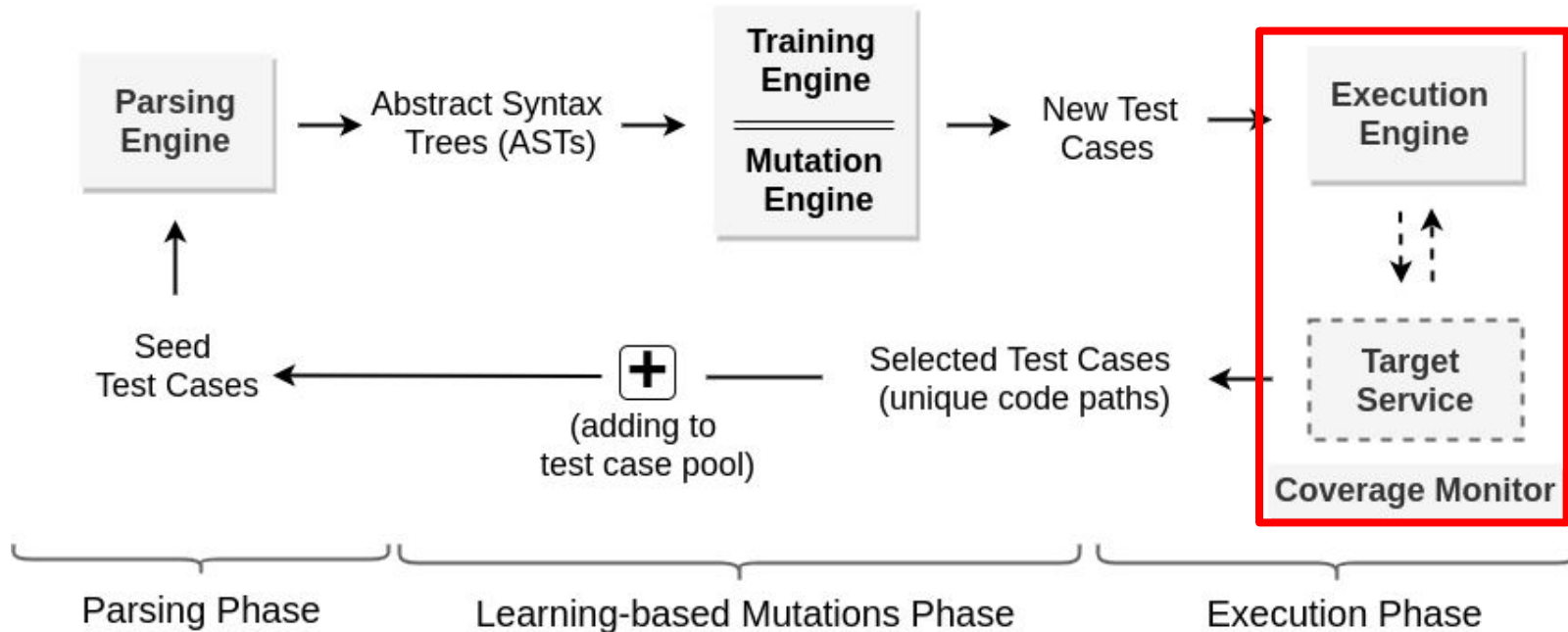


## Mutations



- ❖ Feed seed test cases to encoder of trained model
- ❖ Perturb hidden state of encoder with random noise
- ❖ Compare decoder output with original test cases and generate new mutants

## System overview



## Selected evaluation results

- Q1: How does Pythia compare to other baselines w.r.t. code coverage?
- Q2: Can Pythia find bugs in production-scale cloud services?

## Study subjects

- ❖ Gitlab, Spree, and Mastodon APIs

## Selected evaluation results

- Q1: How does Pythia compare to other baselines w.r.t. code coverage?
- Q2: Can Pythia find bugs in production-scale cloud services?

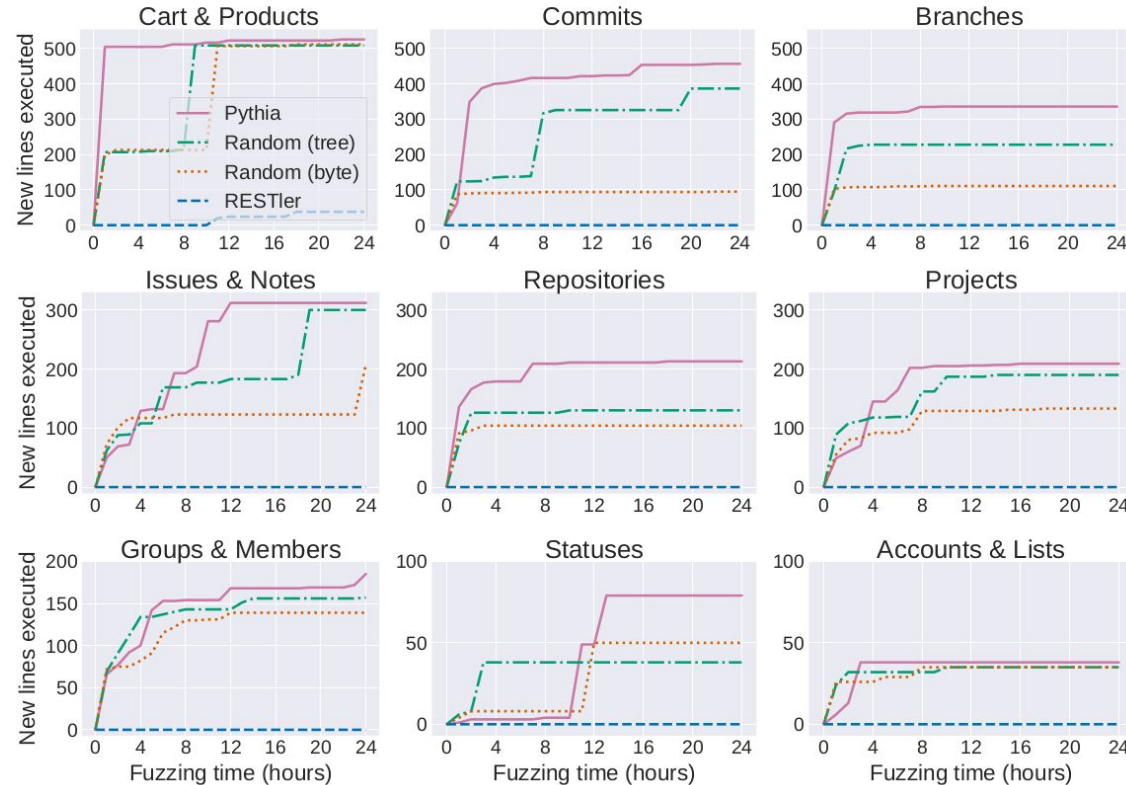
## Study subjects

- ❖ Gitlab, Spree, and Mastodon APIs

## Baselines

- ❖ RESTler: 24 hours initial seed creation & 24 hours fuzzing
- ❖ Byte-level mutations: Random byte alternations on seeds
- ❖ AST-level mutations: Mutations on AST representations

## Code coverage (Q1)



- ❖ RESTler finds no new lines (plateau after the first 24 hours)
- ❖ Pythia finds new lines after RESTler plateau in all APIs
- ❖ Relative ordering is same in all APIs: Pythia > AST-level > byte-level mut.

## New bugs found (Q2)

APIs	RESTler			Pythia		
	Tests	500s	Bugs	Tests	500s	Bugs
Commits	11.6K	0	0	10.7K	132	3
Branches	10.3K	0	0	12.3K	135	4
...	...	...	...	...	...	..
Statuses	58.8K	336	1	56K	962	1
Cart	15.5K	2018	1	18.7K	401	3
<b>Total</b>	<b>194.5K</b>	<b>2.3K</b>	<b>2</b>	<b>220.5K</b>	<b>3.7K</b>	<b>29</b>

Comparison of #tests and #errors after 24h fuzzing

- ❖ Both tools produce same order of magnitude of tests

## New bugs found (Q2)

APIs	RESTler			Pythia		
	Tests	500s	Bugs	Tests	500s	Bugs
Commits	11.6K	0	0	10.7K	132	3
Branches	10.3K	0	0	12.3K	135	4
...	...	...	...	...	...	..
Statuses	58.8K	336	1	56K	962	1
Cart	15.5K	2018	1	18.7K	401	3
Total	194.5K	2.3K	2	220.5K	3.7K	29

Comparison of #tests and #errors after 24h fuzzing

- ❖ Both tools produce same order of magnitude of tests
- ❖ Pythia more 500s than RESTler

## New bugs found (Q2)

APIs	RESTler			Pythia		
	Tests	500s	Bugs	Tests	500s	Bugs
Commits	11.6K	0	0	10.7K	132	3
Branches	10.3K	0	0	12.3K	135	4
...	...	...	...	...	...	..
Statuses	58.8K	336	1	56K	962	1
Cart	15.5K	2018	1	18.7K	401	3
Total	194.5K	2.3K	<b>2</b>	220.5K	3.7K	<b>29</b>

Comparison of #tests and #errors after 24h fuzzing

- ❖ Both tools produce same order of magnitude of tests
- ❖ Pythia more 500s than RESTler
- ❖ Pythia found previously-unknown bugs across all APIs

## Main take-aways

- ✓ Introduced stateful REST API fuzzing
- ✓ Extended stateful REST API fuzzing to new classes of bugs
- ✓ Augmented stateful REST API fuzzing learning-based mutations and code coverage feedback

## Conclusions

- Fuzzing cloud services by leveraging the structure of REST APIs and feedback obtained during interaction is new
- New types of bugs: Unlike buffer overflows, use-after-free bugs, or cross-site-scripting attacks, it is still unclear how serious the errors that hide behind REST APIs are
- Mark a clear path forward to test cloud services in automatic, efficient, and learning-based way

# Structure and Feedback in Cloud Service API Fuzzing

## Sample bugfix in Gitlab

Showing 3 changed files +1E

```
▼ changelogs/unreleased/api-empty-commit-message.yml 0 → 100644  View file @ ceae58c
1 + ---
2 + title: 'API: Catch empty commit messages'
3 + merge_request: 21322
4 + author: Robert Schilling
5 + type: fixed
```

```
▼ lib/api/files.rb  View file @ ceae58c
...  ...  @@ -59,7 +59,7 @@ module API
59  59      params :simple_file_params do
60  60          requires :file_path, type: String, desc: 'The url encoded path to the file. Ex. lib%2Fclass%2Erb'
61  61          requires :branch, type: String, desc: 'Name of the branch to commit into. To create a new branch,
        also provide `start_branch`.'
62  - requires :commit_message, type: String, desc: 'Commit message'
62  + requires :commit_message, type: String, regexp: /^\\S+$/, desc: 'Commit message'
63  63          optional :start_branch, type: String, desc: 'Name of the branch to start the new commit from'
64  64          optional :author_email, type: String, desc: 'The email of the author'
65  65          optional :author_name, type: String, desc: 'The name of the author'
...  ...
```

```
▼ spec/requests/api/files_spec.rb  View file @ ceae58c
...  ...  @@ -337,6 +337,18 @@ describe API::Files do
337  337      expect(response).to have_gitlab_http_status(400)
338  338      end
339  339
340  + it 'returns a 400 bad request if the commit message is empty' do
341  +     invalid_params = {
342  +         branch: 'master',
343  +         content: 'puts 8',
344  +         commit_message: ''
345  +     }
346  +
347  +     post api(route(file_path), user), invalid_params
348  +
349  +     expect(response).to have_gitlab_http_status(400)
350  +     end
351  +
340  352     it "returns a 400 if editor fails to create file" do
341  353         allow_any_instance_of(Repository).to receive(:create_file)
342  354         .and_raise(Gitlab::Git::CommitError, 'Cannot create file')
...  ...
```

# Structure and Feedback in Cloud Service API Fuzzing

## Developers' Responses

Patrice, thank you for reporting the bugs!  
Plz provide instructions on how integrate the tool into the build

Please file VSO for each – these are real bugs. We already fixed a few in DNS.

```
Showing 3 changed files with 47 additions and 18 deletions
changelogs/unreleased/api-empty-commit-message.yml
1 + ---
2 + title: 'API: Catch empty commit messages'
3 + merge_request: 21322
4 + author: Robert Schilling
5 + type: fixed

lib/api/files.rb
59 @@ -59,7 +59,7 @@ module API
60   params :simple_file_params do
61     requires :file_path, type: String, desc: 'The url encoded path to the file. Ex.
62     lib%2Fclass%2Eerb'
63     requires :branch, type: String, desc: 'Name of the branch to commit into. To create a new
64     branch, also provide 'start_branch'.'
65     - requires :commit_message, type: String, desc: 'Commit message'
66     + requires :commit_message, type: String, allow_blank: false, desc: 'Commit message'
```

**Mark Fletcher** @markglenfletcher · 3 weeks ago

@vatlidak1 These bug reports are great. Please continue to raise

Mark Fletcher @markglenfletcher · 1 week ago #50276 Maintainer

@vatlidak1 This one appears to be associated with the `code` attribute rather than `title`? Please can you check the title and description of this issue?

Mark Fletcher @markglenfletcher added `Create` `api` `bug` `devops.create` `snippets` labels · 1 week ago

Mark Fletcher @markglenfletcher · 8 months ago #50677

@vatlidak1 Thanks for the report. I was able to reproduce it.

Let's open this for a ~"Community Contribution" with ~"Accepting Merge Requests"

Mark Fletcher @markglenfletcher added `Accepting merge requests` `Create` `api` `bug` `reproduced on GitLab.com` labels 8 months ago

Mark Fletcher @markglenfletcher · 9 months ago #50272 Maintainer

@vatlidak1 Thanks for the report. I presume that the project is still available on disk at this point and the deferred project execution has not yet been completed

Mark Fletcher @markglenfletcher added `Manage` `api` `bug` `project` `repository` labels 9 months ago

500 Internal Server Error: Delete repository file with empty commit message #50268

**1 Related Merge Request**

!21322 **API: Catch empty commit messages** Open

When this merge request is accepted, this issue will be closed automatically.