

Intelligent REST API Data Fuzzing

会议：ACM ESEC/FCE 2020，软件顶刊，CCF A类

作者：来自美国微软研究院与普林斯顿大学的研究团队

Presenter: zouyue

1 BACKGROUND AND MOTIVATION

背景:

- 大多数云服务通过REST API访问, 如Azure。
- 目前的最新工具restler对模糊复杂数据的支持有限。

动机:

- 数据格式 (schema) 中包含无限多种数据格式、数据数量以及数据大小, 因此有无限种数据变异的方式。
- 因此本文提出各种针对schema json数据格式变异的方法, 并使用 Azure DNS 服务为测试对象, 结合restler工具来评估各个策略的有效性, 得到最有效的数据变异模糊策略。

```
1  {
2    "etag": "string",
3    "properties": {
4      "registrationVirtualNetworks": [
5        { "id": "string" }
6      ],
7      "maxNumberOfRecordSets": 0,
8      "numberOfRecordSets": 0,
9      "nameServers": [ "string" ],
10     "zoneType": { "enum": ["Public", "Private"] },
11     "registrationVirtualNetworks": [
12       { "id": "string" }
13     ],
14     "resolutionVirtualNetworks": [
15       { "id": "string" }
16     ]
17   },
18   "id": "string",
19   "name": "string",
20   "type": "string",
21   "location": "string",
22   "tags": { "string": "string" }
23 }
```

Figure 1: Example of REST API JSON body schema.

2 Schema Fuzzing

Schema json格式, 定义为树结构: $G=(V,E,\Gamma,T)$ 。

```
1 // example schema
2 // nodes
3 V = { root,
4     tag, properties,
5     id, time
6 }
7 // edges
8 E = { (root, tag),
9     (root, properties),
10    (properties, id),
11    (properties, time)
12 }
13 // type
14 T(root) = object
15 T(tag) = string
16 T(properties) = object
17 T(id) = string
18 T(time) = integer
```

```
1 // example payload
2 {
3   "tag": "global",
4   "properties": {
5     "id": "abcd",
6     "time": 3600
7   }
8 }
```

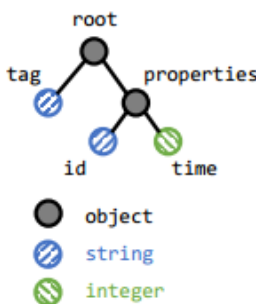


Figure 2: Example of schema and payload.

Node Fuzzing Rules:

- 1.删除 (**drop**):删除给定节点n的一个子节点c
- 2.保留 (**select**):保留给定节点n的一个子节点c, 其余节点删除
- 3.复制 (**Duplicate**): 复制给定节点n的子节点 (包括c的后代节点) 到n节点下。
- 4: 类型变化 (**type**):更改给定节点的数据类型, 并保留类型改变后引起的树结构变化, 例如将object改位string会丢失子节点。

2 Schema Fuzzing

Tree Fuzzing Rules:

- 1.单节点 (**single**) :将节点模糊规则应用于一个节点, 同时保持所有其他节点不变
- 2.路径节点 (**path**) :在模式树中选择一条路径, 然后将节点模糊规则应用于该组中的每个节点
- 3.全部节点 (**all**) : 给定节点模糊规则和模式, 树模糊规则 All 选择模式树中的一组节点, 然后将节点模糊规则应用于该组中的每个节点

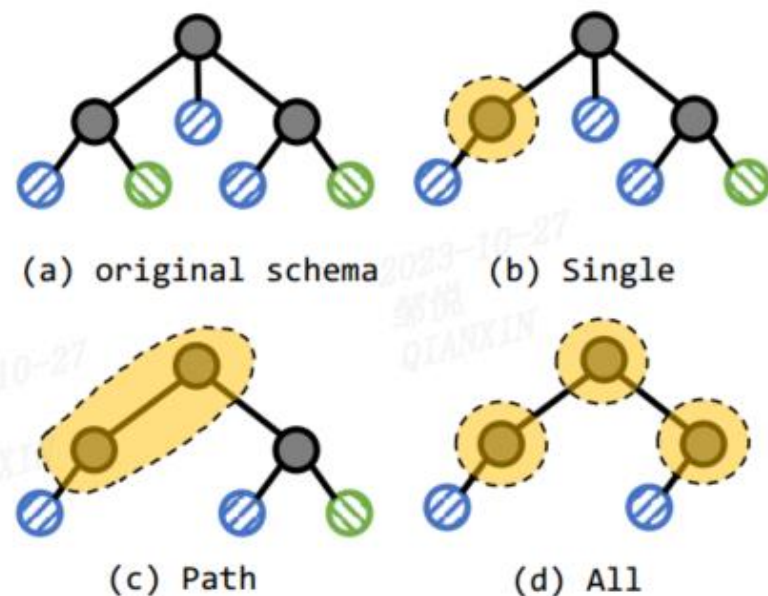


Figure 3: The original schema and the fuzzed-schemas generated by applying the node fuzzing rule Drop using different tree fuzzing rules (Single, Path, and All).

2 Schema Fuzzing

实验评估指标

评估指标：响应错误类型覆盖指标 (**error type coverage metric**)

- 1.错误码 (Error code)** :服务器返回的HTTP错误状态码, 包括4XX、5XX
- 2.错误信息 (Error Message)** :失败请求响应中包括的详细错误描述消息, 如“The resource record is missing field ‘target’ ”。
- 3.错误类型 (Error Type)** : 将错误类型定义为一对错误代码和错误消息。不同错误类型的数量在实验中用作有效性指标

实验设置

作为restler的扩展对Azure的服务进行fuzz测试。通过对每种请求类型(request type)生成不超过1000种变异数据来测试12种节点变异与树变异方式的组合。

对于具体参数值部分, 分别使用“fuzzstring”、0、false、{}和[]作为字符串、整数、布尔、对象和数组类型的输入

2 Schema Fuzzing

实验结果总结

Drop & select: 结合path策略能够发现更多的错误类型。
All策略也有效但消耗较大, 因此path最佳。

Type & duplicate: 前者可造成数据类型不匹配, 后者造成json格式错误, 均能引起反序列化问题。但不合适对过多节点进行操作, 因此结合single策略最佳。

不同的策略引起的错误类型不同, 因此是互补的。最终选择使用路径删除 (DROP)、使用路径选择 (SELECT)、使用单一重复 (DUPLICATE) 和使用单一类型 (TYPE)) 进行进一步的研究

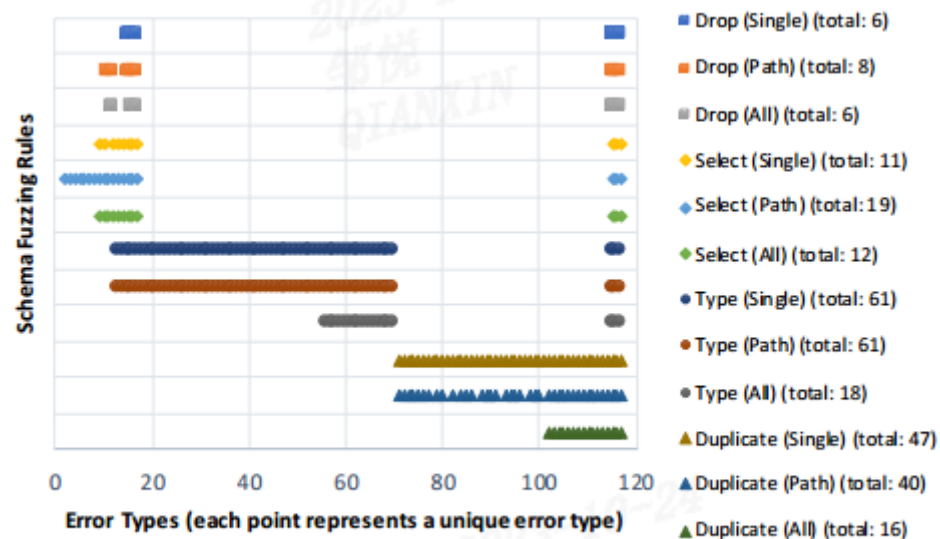


Figure 4: Error type coverage for each schema fuzzing rule.

3 COMBINING SCHEMA FUZZING RULES

$$\text{DROP-TYPE}(G) = \bigcup_{G_i \in \text{DROP}(G)} \text{TYPE}(G_i)$$

多种变异策略结合

一个模糊规则应用于初始主体模式 (schema) 并生成一组模糊后模式, 然后第二个模糊规则应用于所有这些模糊模式并生成甚至更多的模糊模式

组合后会产生大量新的schema, 但预算有限, 因此提出并评估 3 种启发式方法 (Search Heuristics) 来选择由流水线模糊规则生成的模糊模式: (1) 深度优先 (DF), (2) 广度优先 (BF) 和 (3) 随机策略 (RD)。

- **深度优先 (DF)**: 给定最大界限M, 搜索启发式 DF以深度优先顺序生成模糊模式, 并选择前M个模糊模式。
- **广度优先 (BF)**: 搜索启发式 BF 通过以广度优先顺序生成模糊模式, 在早期阶段优先考虑模糊测试。
- **随机策略 (RD)**: 搜索启发式 RD 使用不支持特定阶段的随机搜索顺序。

3 COMBINING SCHEMA FUZZING RULES

实验评估

根据上一节的研究将四种变异规则分为三类：（1）Drop & select (2) type (3) duplicate。实现了23个模糊规则组合来覆盖这三组的不同组合，每组均进行三种搜索启发式的实验。

实验结果

- 规则组合方面： Drop & select 与type结合有增益，且Drop & select 先于Type执行覆盖率更高； Duplicate 与其他模糊测试规则相结合并没有提供明显的改进。

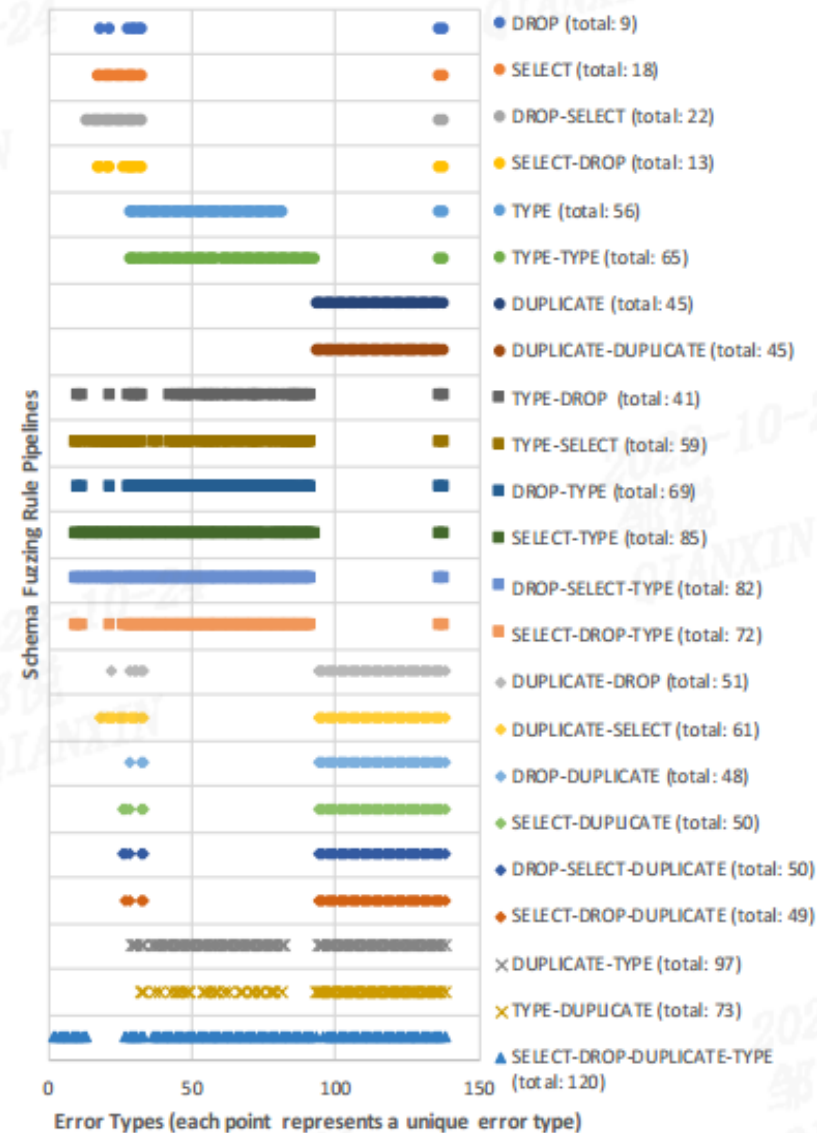


Figure 5: Error type coverage for each schema fuzzing rule pipeline (search heuristic: RD, random seed: 0).

3 COMBINING SCHEMA FUZZING RULES

实验结果

- 搜索启发式 (Search Heuristic)：将规则组合SELECT-DROP-DUPLICATE-TYPE应用于请求PUT DNS-Zone和PUT DNS-Record-Set。
- 如图所示，无论使用何种随机种子，使用RD都能提供更稳定的增长率，在预算有限时更有利。

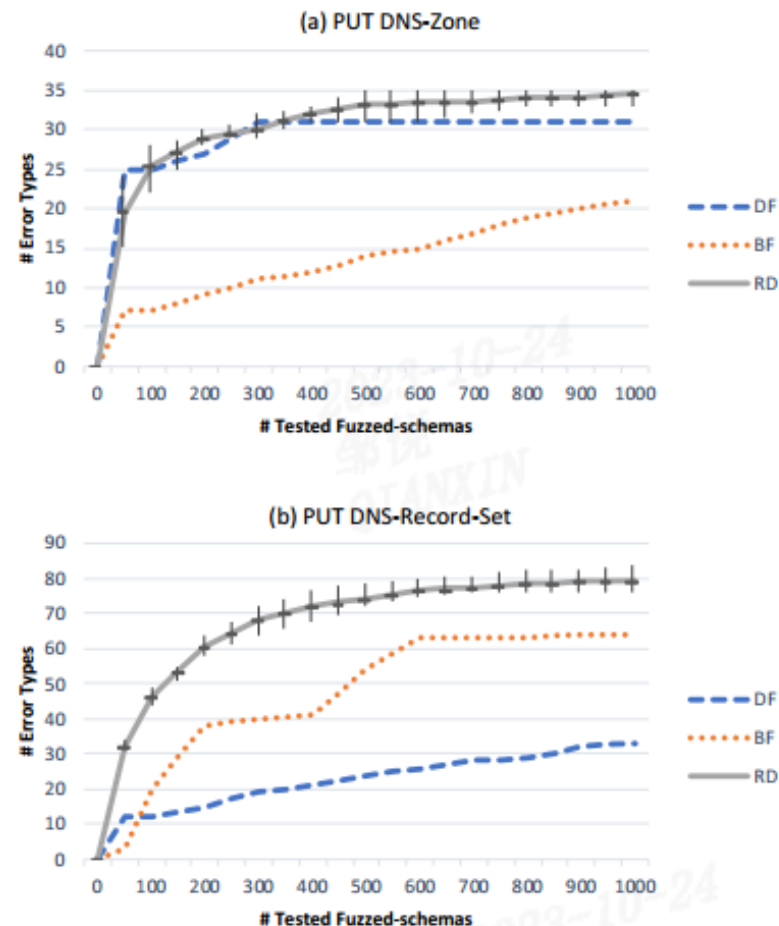


Figure 6: The growth trends of the number of error types discovered over the number of tested fuzzed-schemas (schema fuzzing rule pipeline: SELECT-DROP-DUPLICATE-TYPE).

4 DATA VALUE RENDERING

- 许多请求最终由于其主体有效负载中的一个节点的单个特定无效值呈现而被拒绝
- 参数生成的难点：缺少特定于客户端的信息、缺乏特定于域的信息、缺乏运行时相关信息。

参数生成策略（Value Rendering Strategies）：

- 静态类型值映射
- 使用openAPI/swagger规范中的示例
- 从response中学习得到。提取响应负载正文中每个叶节点的值，并使用其在JSON层次结构中的路径对其进行标记。为节点选择具体值时，使用模式匹配将候选值的标签与模糊模式树结构中的节点路径进行比较。分为激进型与保守型两种。

ESEC/FSE '20, November 8–13, 2020, Virtual Event, USA

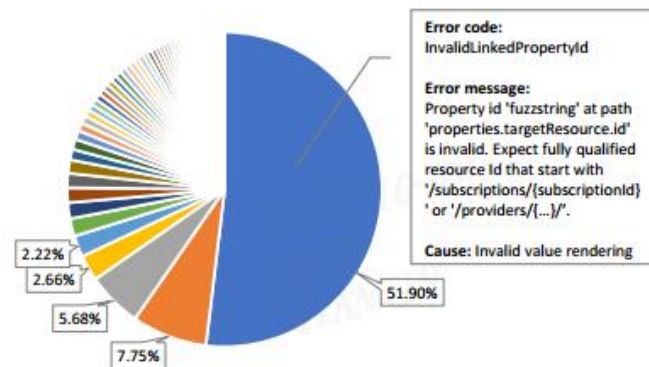


Figure 7: Percentage of tests triggering each error type (schema fuzzing rule pipeline: DROP-SELECT-TYPE).

```
1 {
2   "name": "object-1234-abcd",
3   "properties": {
4     "type": "Public",
5     "numberOfTags": 2,
6     "maxNumberOfTags": 5000
7   },
8   "location": "global",
9   "id": "/subscriptions/subid/resourceGroup/..."
10 }
```

Figure 8: Example response payload of a successful request.

4 DATA VALUE RENDERING

支持的参数生成策略：

- **baseline (BAS)**：使用类型值映射
- **Example only (EXM)**：选择示例值，如果没有示例值则选择类型映射值。
- **Responses only (conservative) (CON)**：选择保守模式下的响应值为参数值，没有则使用类型值映射
- **Responses only (aggressive) (AGG)**：选择激进模式下的响应值为参数值，没有则使用类型值映射
- **Responses (conservative) and examples (CON+EXM)**：选择保守模式下的响应值为参数值，没有则使用示例值。
- **Responses (aggressive) and examples (CON+EXM)**：选择激进模式下的响应值为参数值，没有则使用示例值。

实验设置：

- 分别使用6中参数生成方法对 Azure DNS的同一组模糊模式进行测试。
- 选择两个“最佳”模式模糊规则-
DUPLICATE 和 DROP-SELECT-TYPE
来生成一组模糊模式 (fuzzed-schemas)，每个规则生成的最大数量为1000，并使用随机种子0的RD搜索启发式。

4 COMBINING SCHEMA FUZZING RULES

实验结果：使用openAPI/swagger规范示例与response（尤其是激进模式）中的值均有助于覆盖更多错误类型。

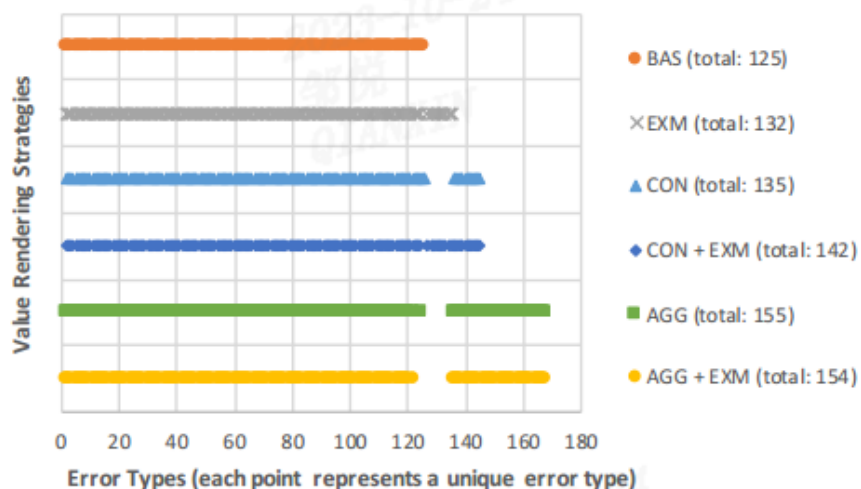


Figure 9: Error type coverage for each value rendering strategy (pipelines: DUPLICATE and DROP-SELECT-TYPE).

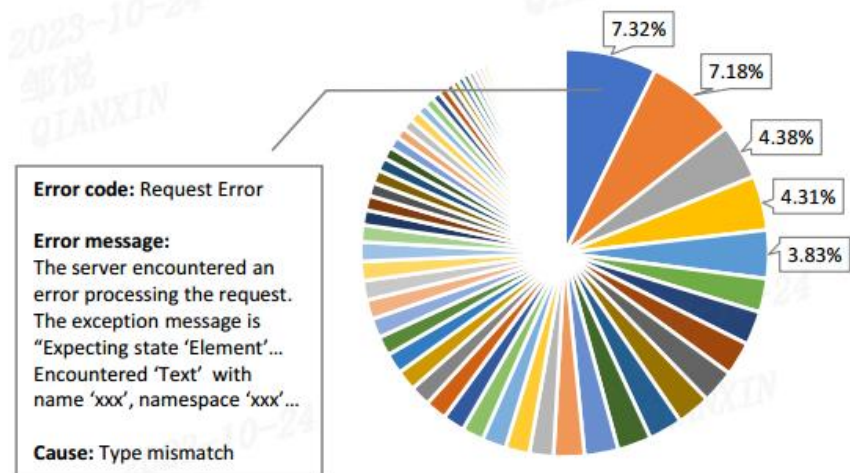


Figure 10: Percentage of tests triggering each error type (rendering strategy: AGG+EXM, pipeline: DROP-SELECT-TYPE).

5 BUG HUNTING IN CLOUD SERVICES

REST API Data Fuzzing Algorithm:

该算法分为两个部分，分别使用模糊规则DUPLICATE和DROP-SELECT-TYPE。每个部分的预算是schema中节点数的10倍。第一部分侧重于生成违反JSON格式的数据有效负载，而第二部分侧重于测试各种树结构和类型不匹配。RD模式与AGG+EXM模式

在第二部分DROP-SELECT-TYPE中分为两个阶段，DROP-SELECT阶段进行预算10%的变异，分析相应的response，之后进行type变异

实验设置

测试对象为与网络相关的Microsoft Azure云服务，包括前面作为实验基准的Microsoft Azure DNS服务，共约58000行swagger说明。

所有实验均在一台配备Intel i7处理器和32GB主内存、运行Windows 10的商用PC上进行。

5 BUG HUNTING IN CLOUD SERVICES

实验结果： Azure DNS方面上报了7个unique bug； Azure Networking方面分类后上报了10个unique bug。

实验总结： 如果没有进行组合的变异策略， 无法发现以下的错误； RESTler无法发现以下错误； TYPE变异是关键变异。

Table 1: Overview of the bugs found in Azure DNS REST APIs.

#	API Request	Schema node	Bug description	Variants
1	PUT DNS-Zone	location	Type mismatch (string to array or boolean)	2
2	PUT DNS-Record-Set	Records	Enum type mismatch between request path and body (e.g., SRV and caaRecords)	7
3	PUT DNS-Record-Set	Records	Missing node (e.g., "ARecords": {} or "ARecords": [{}])	3
4	PUT DNS-Record-Set	value	Type mismatch (integer to boolean or string)	13
5	PATCH DNS-Record-Set	Records	Enum type mismatch between request path and body (e.g., AAAA and SRVRecords)	5
6	PATCH DNS-Record-Set	Records	Missing node (e.g., "TXTRecords": {} or "TXTRecords": [{}])	2
7	PATCH DNS-Record-Set	TTL	Type mismatch (integer to boolean or string)	7

Table 2: Overview of the bugs found in Azure Networking REST APIs.

#	API Request	Schema node	Bug description	Variants
1	PUT virtualWans	location	Type mismatch (string to array or boolean)	28
2	PUT virtualNetworks	addressSpace	Type mismatch (object to boolean)	1
3	PUT virtualNetworks	addressPrefixes	Type mismatch (array to boolean)	1
4	PUT virtualNetworks	addressPrefixes_array	Type mismatch (object to boolean)	1
5	PUT virtualNetworks	subnets_name	Type mismatch (string to integer)	1
6	PUT virtualNetworks	subnets_addressPrefix	Type mismatch (array to boolean)	1
7	PUT virtualNetworks	subnets_delegations_name	Type mismatch (string to boolean)	1
8	PUT virtualNetworks	subnets_delegations_properties	Type mismatch (array to boolean)	1
9	PUT virtualNetworks	subnets_delegations	Missing node (e.g., properties)	1
10	PUT virtualNetworks	subnets_delegations_properties	Missing node (e.g., "properties": {})	7



6 CONCLUSION

- 提出四种结点变异方法与三种树结构变异方法，并通过实验比较两者结合后的有效性，从中选取四种组合进行深入的探索，依次是使用路径删除（DROP）、使用路径选择（SELECT）、使用单一重复（DUPLICATE）和使用单一类型（TYPE）
- 依据不同变异方法的特性，对23种组合变异方法进行实验比较，得到效果最佳且不重复的两组策略：DUPLICATE与DROP-DUPLICATE-TYPE。
- 比较三种不同的搜索启发式，通过实验发现RD模式最稳定且效果最佳
- 提出不同的参数负载生成策略，结合schema变异策略进行实验比较。
- 对Azure云服务的API进行模糊测试，最终发现并上报17个unique bug，并且这些bug无法通过其他方法检测出。



THANKS



请输入你的内容