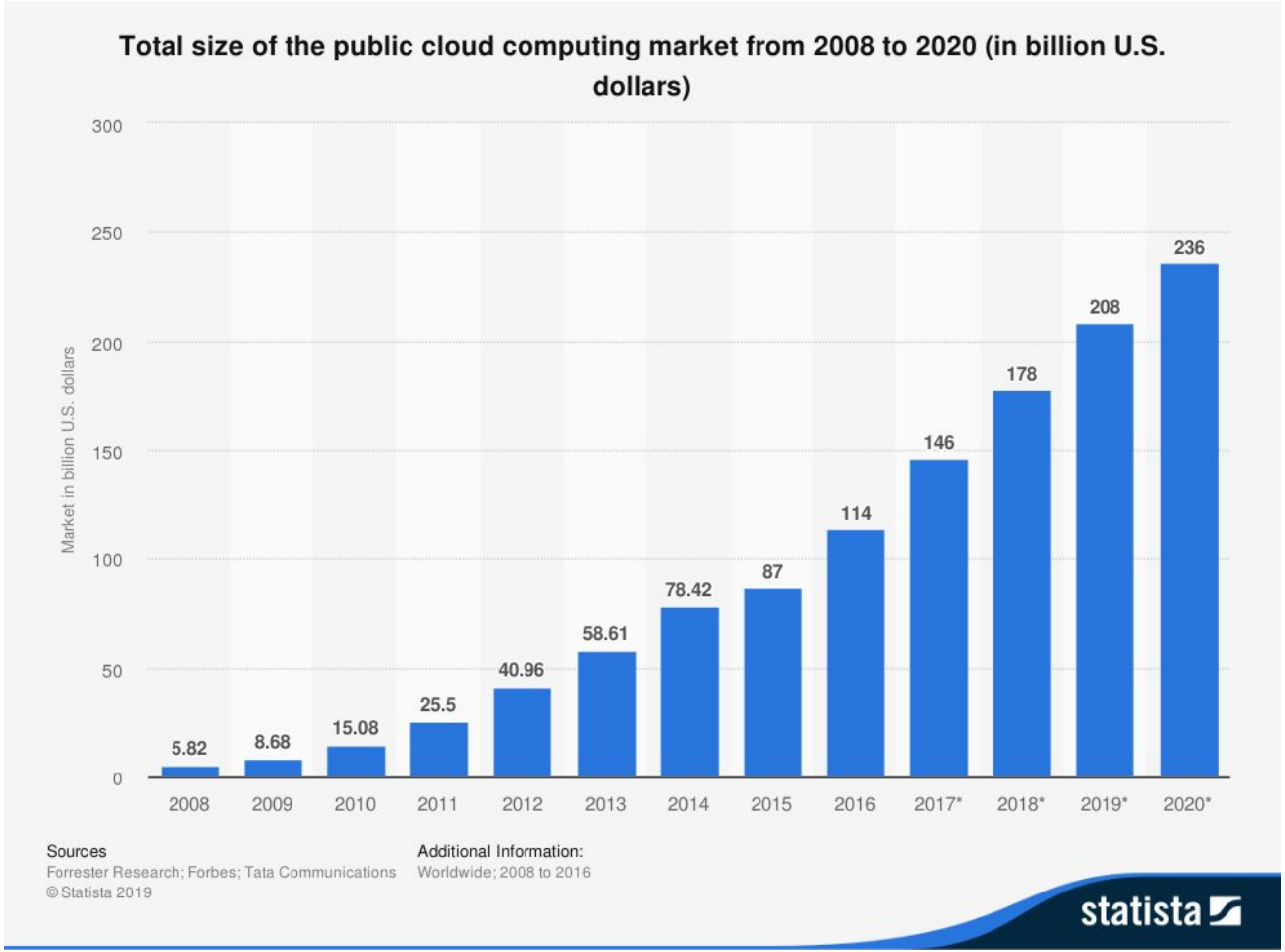
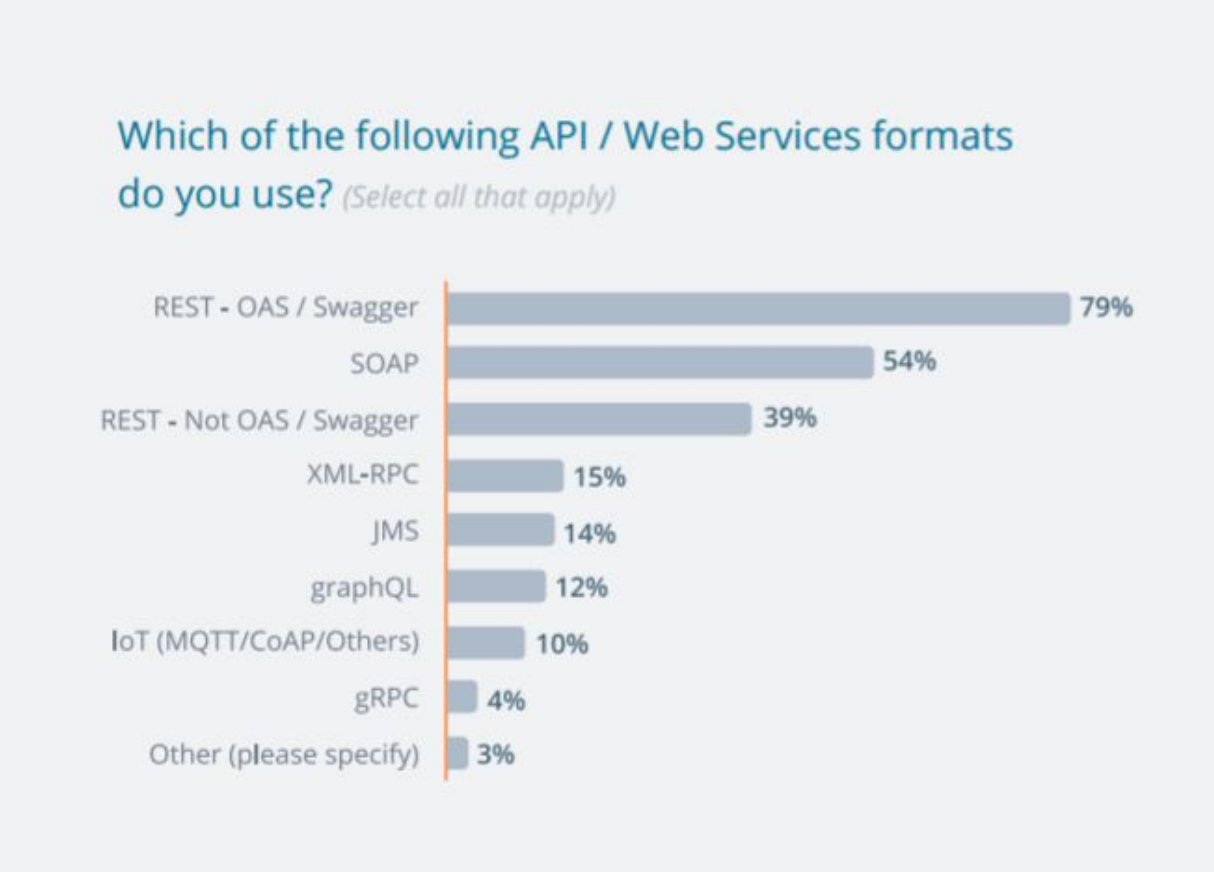
Checking Security Properties of Cloud Service REST APIs

Vaggelis Atlidakis (Columbia University), Patrice Godefroid (Microsoft Research), and Marina Polishchuk (Microsoft Research)

Checking Security Properties of Cloud Service REST APIs



Checking Security Properties of Cloud Service REST APIs



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 (1 / 2)

Testing REST APIs (1 / 2)

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

Testing REST APIs (1 / 2)

- 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 (1 / 2)

- 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)

Testing REST APIs (2 / 2)

- RESTler: Stateful REST API Fuzzing
  - Static analysis on API specification and automatically produce fuzzing grammar: encodes sequences of requests
  - Target errors are unhandled exceptions (“500 Internal Server Errors”)

Challenge

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

Checking Security Properties of Cloud Service REST APIs

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

Checking Security Properties of Cloud Service REST APIs

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

Outline

- Limitations of existing solutions
- **Security rules and desirable properties**
- **System overview**
- **Selected errors found with checkers**
- **Conclusions**

Security rules and desirable properties

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

Security rules and desirable properties

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

- Resource-hierarchy rule
  1. Create /api/posts/1 and /api/posts/2
  2. Create /api/posts/1/reply/1
  - ➢ Access /api/posts/2/reply/1 **MUST FAIL**

Security rules and desirable properties

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

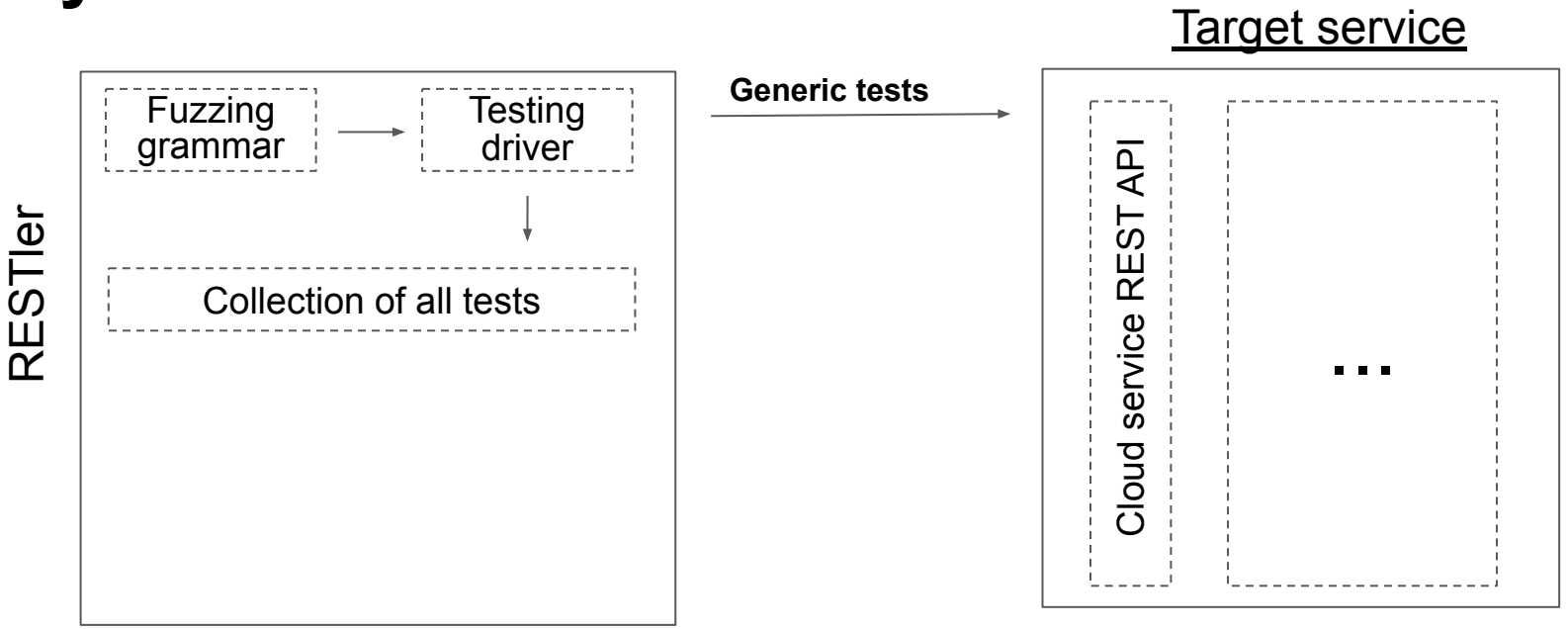
- Resource-hierarchy rule
  1. Create /api/posts/1 and /api/posts/2
  2. Create /api/posts/1/reply/1
  - ➢ Access /api/posts/2/reply/1 **MUST FAIL**

- Resource-leakage rule
  1. Create /api/post/1 and receive error code (e.g., 404 or 500 HTTP status)
  - ➢ Access /api/post/1 **MUST FAIL**

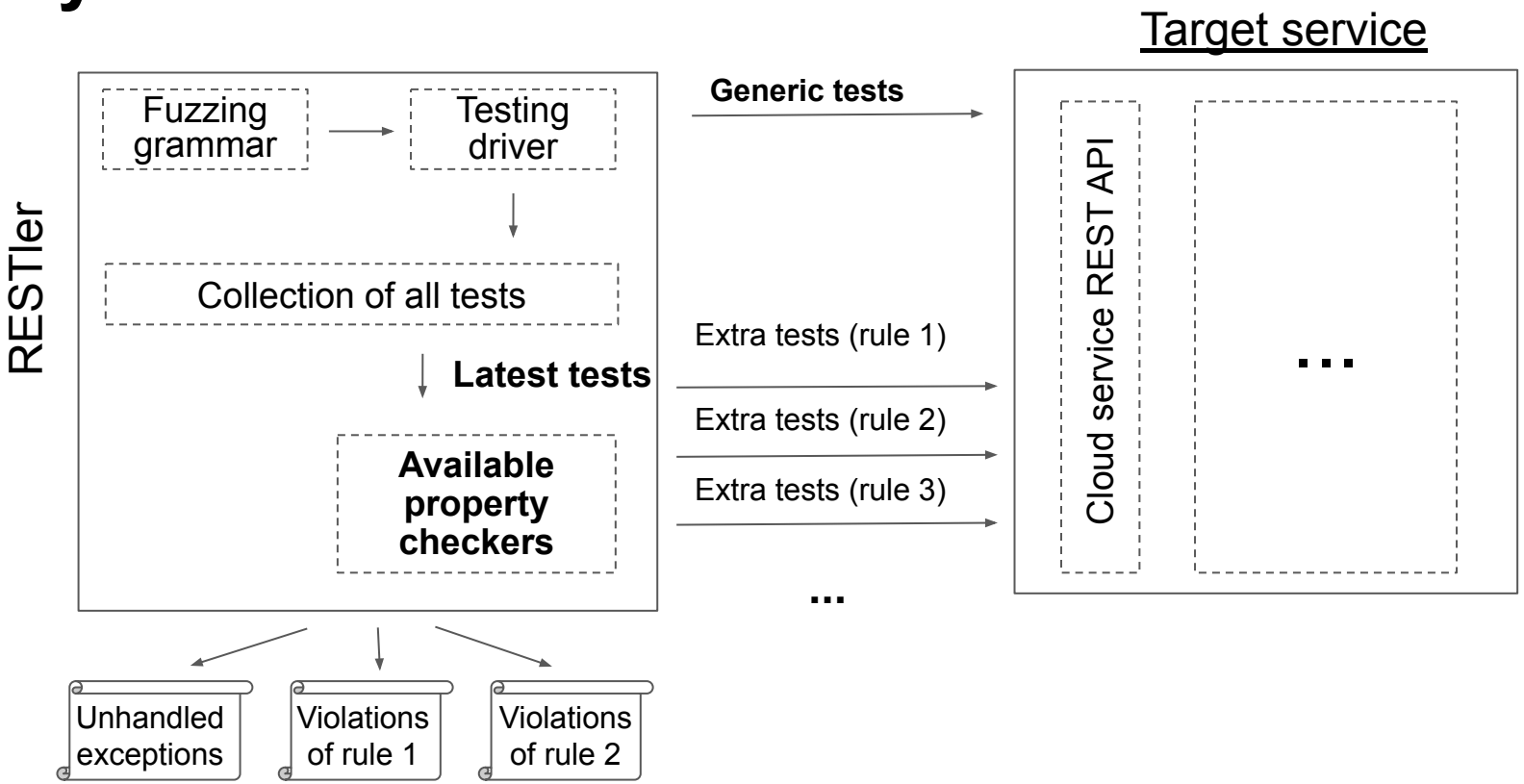
Outline

- Limitations of existing solutions
- Security rules & desirable properties
- **System overview**
- **Selected errors found with checkers**
- **Conclusions**

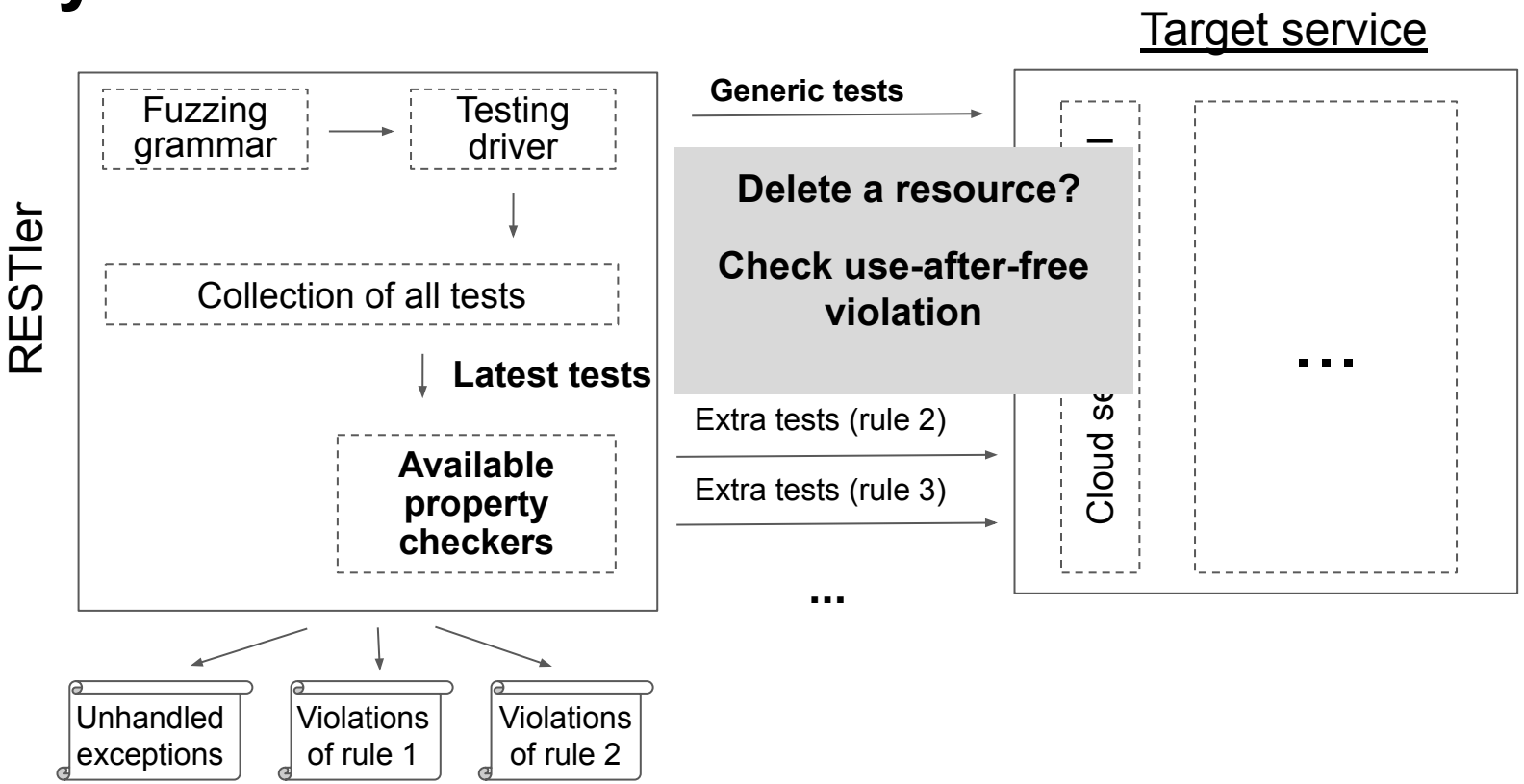
System overview



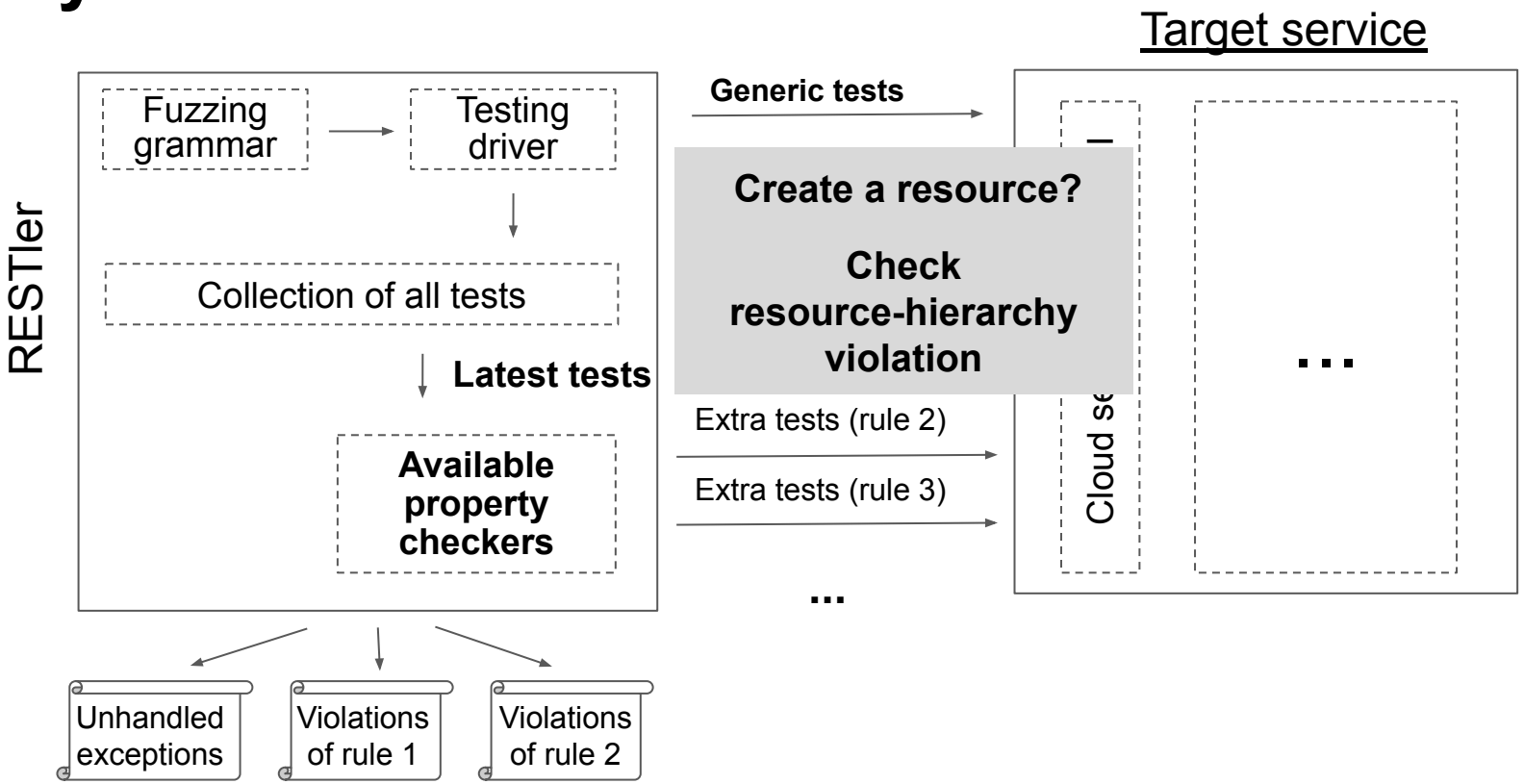
System overview



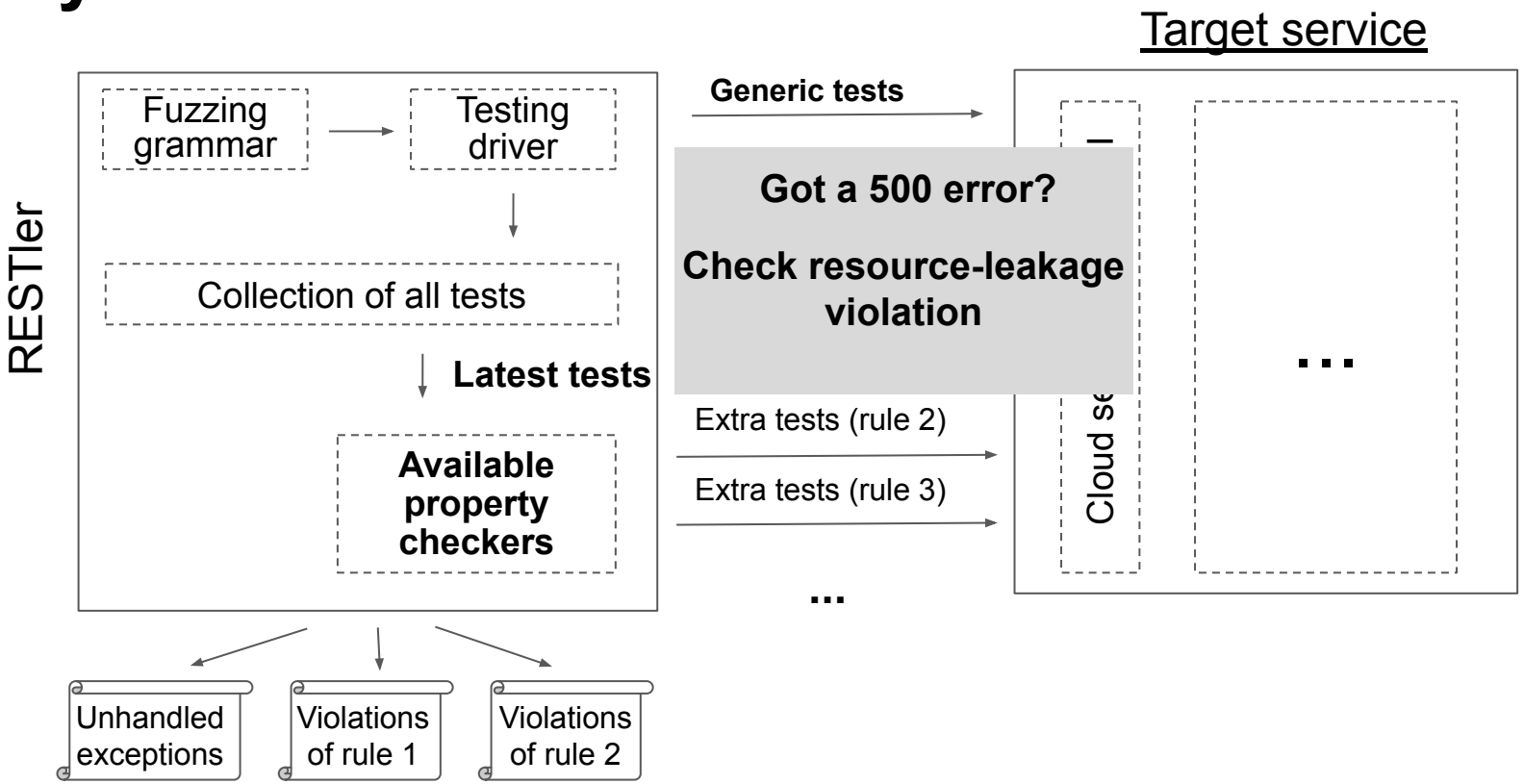
System overview



System overview



System overview



Outline

- Limitations of existing solutions
- Security rules & desirable properties
- System overview
- **Selected errors found with checkers**
- **Conclusions**

Experimental setup

- Target three cloud services from Azure & O-365
- Production services w\ public API specifications
- Complex API requests (~16 API requests per service)

Selected errors found with checkers

Selected errors found with checkers

- Use-after-free rule violation
  1. Create a new resource R
  2. Delete resource R
  3. Create a new child of the deleted resource R
     - ➢ 500 "Internal Server Error" (should have been: 404 Not Found)

Selected errors found with checkers

- Use-after-free rule violation
  1. Create a new resource R
  2. Delete resource R
  3. Create a new child of 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

- Use-after-free rule violation
  1. Create a new resource R
  2. Delete resource R
  3. Create a new child of 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)

Conclusions

- Introduced rules that capture desirable security properties of cloud service REST APIs
- Extended stateful REST API fuzzing with active checkers
- Reported violation of security rules in production Azure & O-365 services

Conclusions

- ❖ Introduced rules that capture desirable security properties of cloud service REST APIs
- ❖ Extended stateful REST API fuzzing with active checkers
- ❖ Reported violation of security rules in production Azure & O-365 services

➢ **All bugs reported have been fixed!**

Thank you!

Paper link

https://tinyurl.com/y45k2kd8