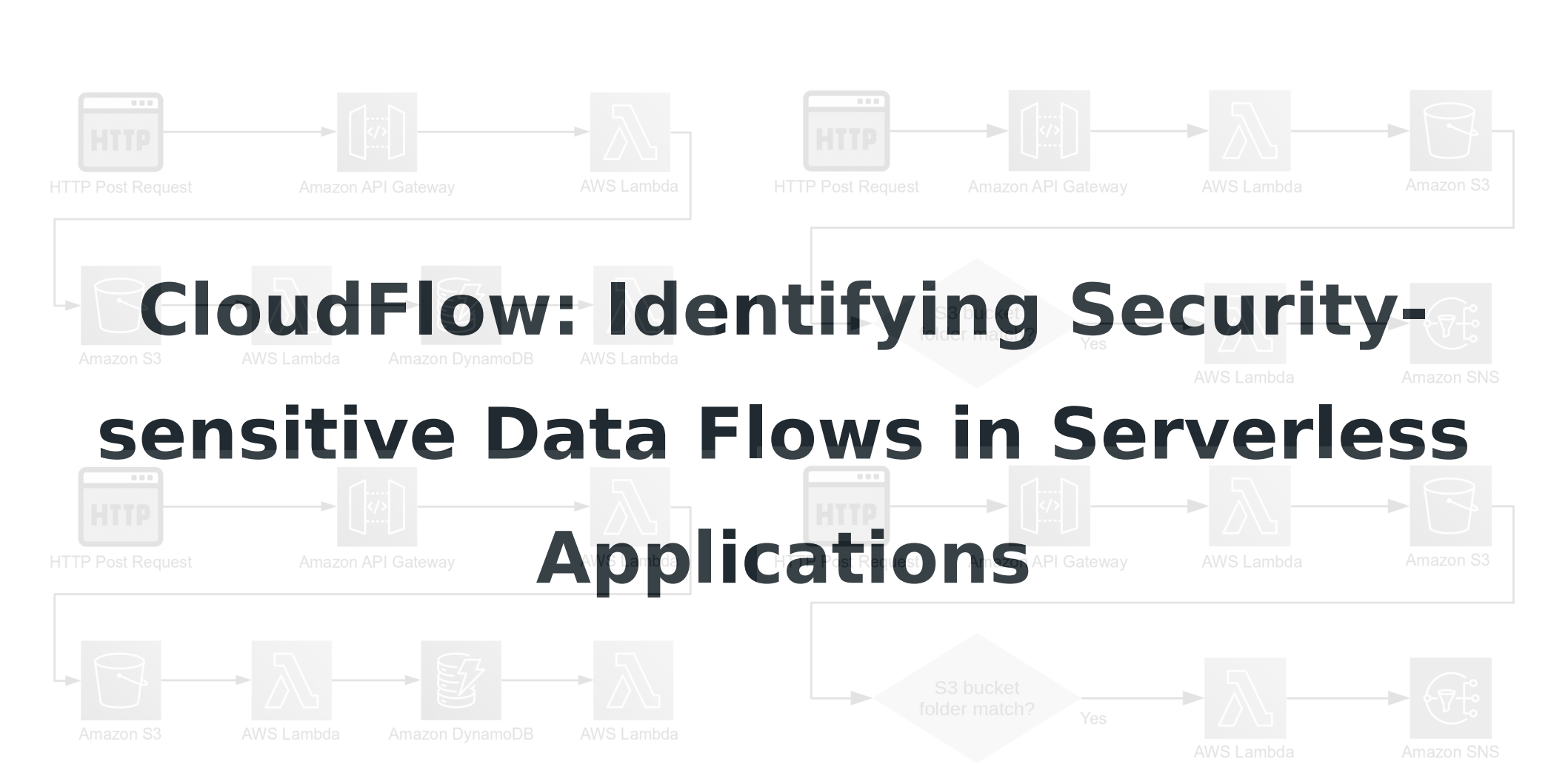


CloudFlow: Identifying Security-sensitive Data Flows in Serverless Applications

Giuseppe Raffa, J. Blasco, D. O'Keeffe, S. K. Dash

USENIX Security '25 Conference, 13th August '25

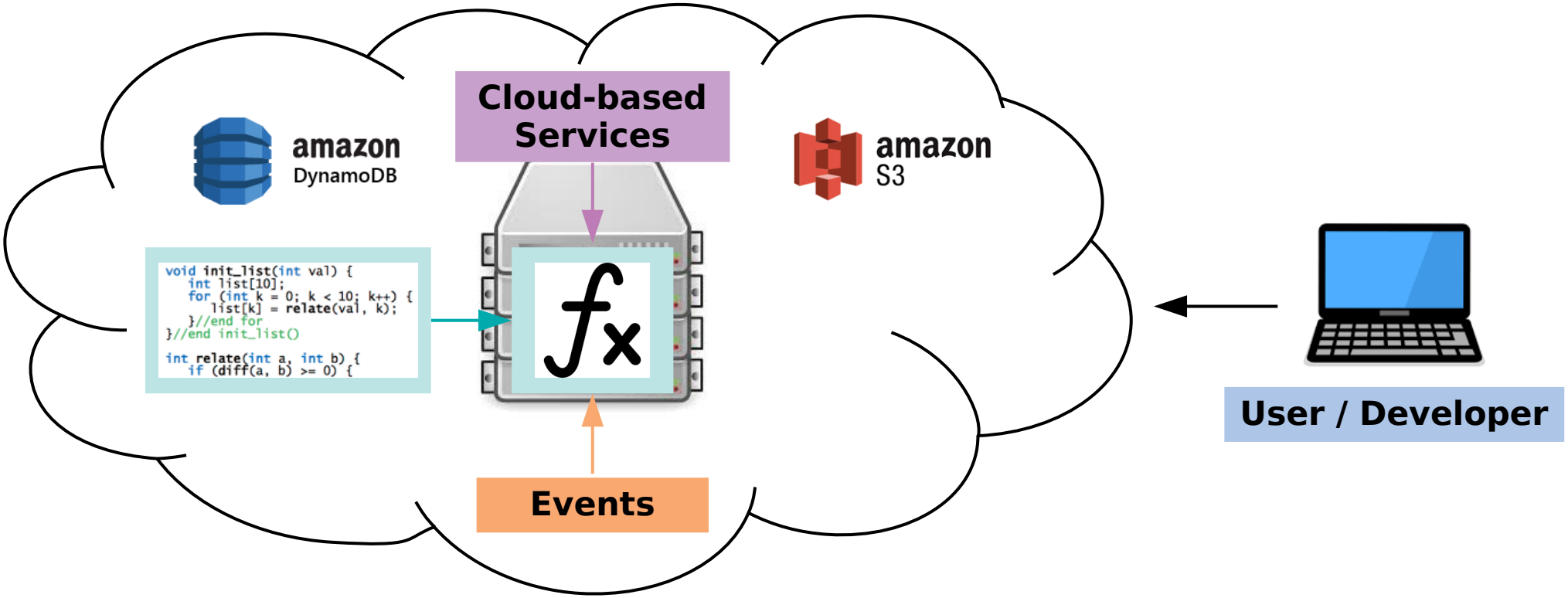
Email: giuseppe.raffa.2018@live.rhul.ac.uk

ROYAL HOLLOWAY UNIVERSITY Of LONDON

UNIVERSIDAD POLITÉCNICA DE MADRID

UNIVERSITY OF SURREY

Serverless Computing Model



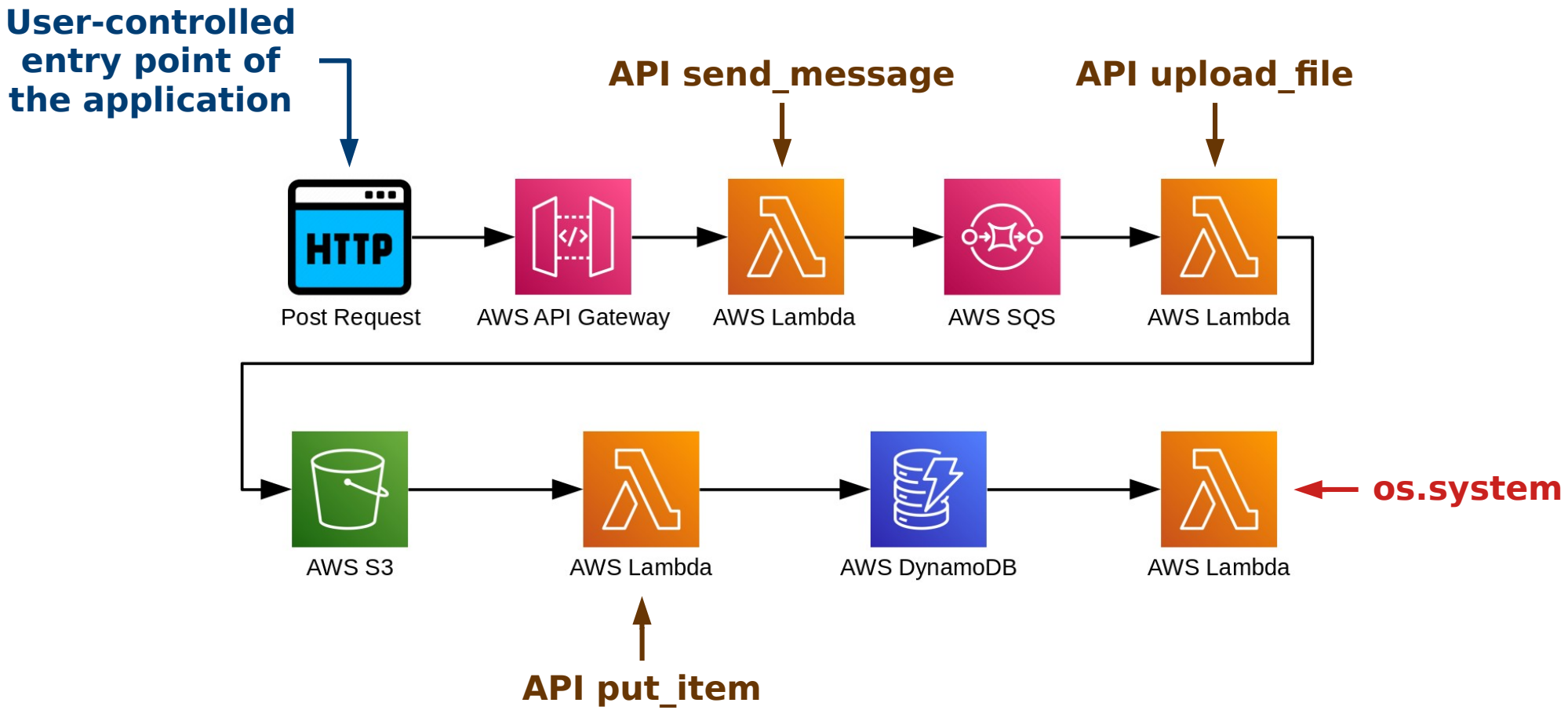
- **Advantage**
  - No infrastructure management
- **Challenge**
  - Security

Motivation & Challenges

- **Why static data flow analysis?**
  - Analysis of an application prior to its deployment
- **What are the challenges?**
  - Event-driven applications
  - Multitude of mechanisms to define application permissions
  - Black-box nature of platform services
- **Contributions**

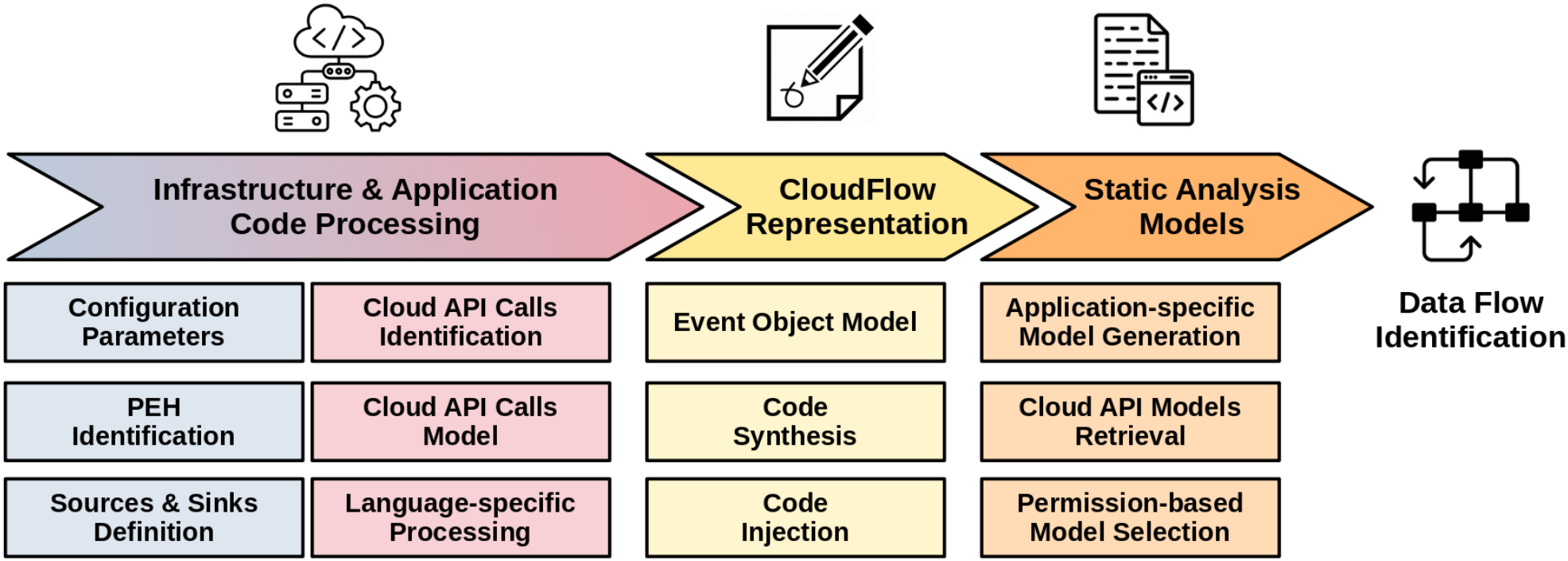
Framework for data flows analysis	New suite of microbenchmarks	Analysis of real-world apps

Motivating Example



A general-purpose static analysis tool would ignore the triggered events

CloudFlow Analysis Pipeline



- **CloudFlow representation**
  - Application code instrumentation
  - Synchronous-like program execution flow

Code Instrumentation Example

```
def onSQSMessage(event, context):
    # Application code
    s3_client: S3Client = boto3.client('s3')
    s3_client.upload_file(localFile,
                          os.environ['BUCKET_NAME'],
                          s3BucketKey)
    event = {'Records': [{'eventName': 'ObjectCreated:*',
             's3': {'bucket': {'name': os.environ['BUCKET_NAME'],
             'arn': os.environ['BUCKET_NAME']},
             'object': {'key': s3BucketKey}}}]}
    s3uploadhandler.onS3Upload(event, context)
    return
```

Cloud API call

Event object initialization

Handler call

Type annotation

Microbenchmark-based Evaluation

- **CloudBench suite**
  - 40 AWS Python applications
  - 30 inter-procedural
  - 9 intra-procedural
  - 27 cases of code injection

Category	Amount
Inter-procedural	30
Intra-procedural	9
Simple Applications	1

- **Evaluation results**
  - CloudFlow passes 37 microbenchmarks out of 40 ✔
  - 1 false positive (CloudFlow limitation) ✖
  - 2 false negatives (underlying static analysis tool) ✖

Real-world Application Analysis

- **Findings**
  - 104 real-world applications analysed

CloudFlow detects 205 data flows

Category	Total No.
Vulnerability	11
Potential Vulnerability	64
No Vulnerability	130

- **Vulnerabilities**

Unchecked CLI parameters

Data flow triggered by HTTP request

Vulnerability Type	Total No.
Code Injection via CLI	2
Code Injection via HTTP	5
Exception Traceback Leakage	4

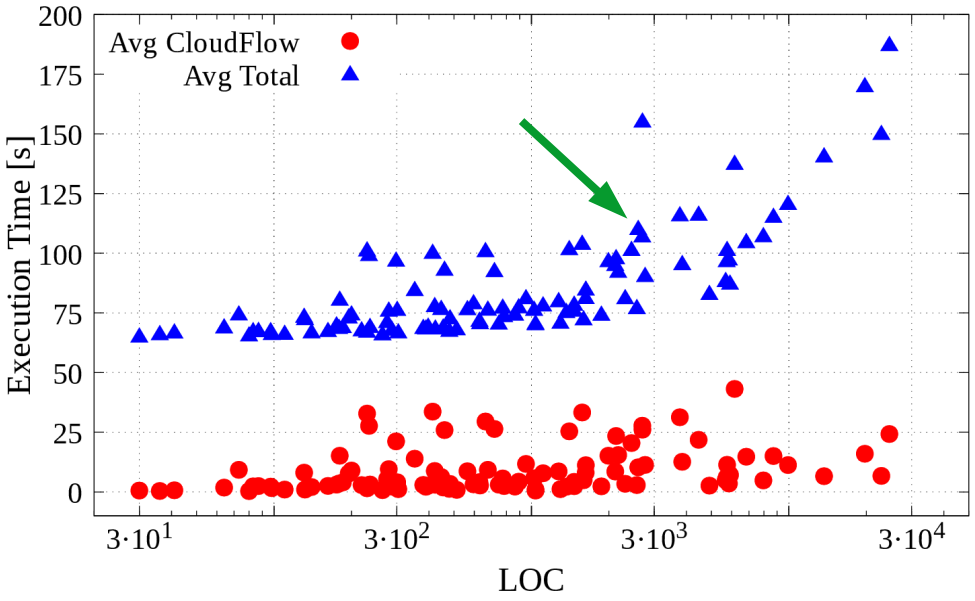
Vulnerability Case Study

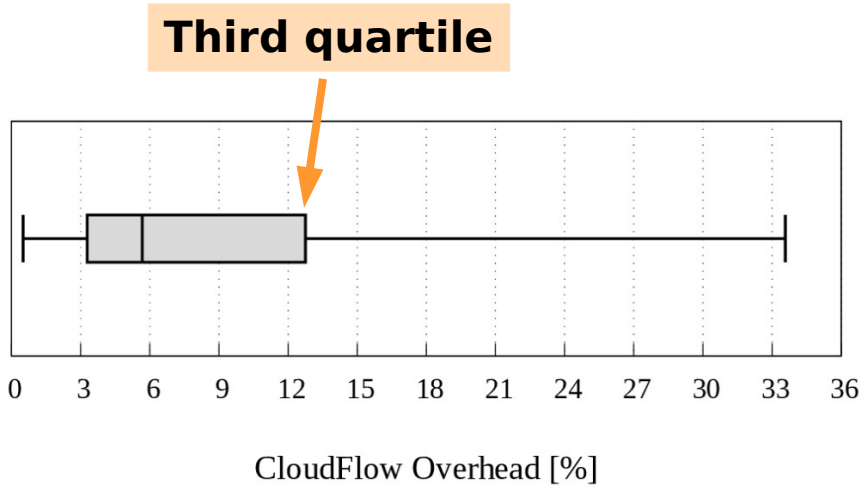
- **Exception traceback leakage**
  - Information about exception returned in HTTP response
  - Not directly exploitable, but should be used only for testing
  - Case of intra-procedural data flow
  - Inter-procedural case study in the paper

```
def event_process(missing_event) -> Response:
    try:
        # Processing and logging commands
    except Exception as e:
        return Response(str(e), status=500)
```

Performance Evaluation

- **Execution times**
  - Total execution times increase when LOC ≥ 3,000
  - CloudFlow-specific execution times do not show same trend
  - For 75% of the analysed applications overhead ≤ 12.8%





Conclusion & Future Work

- **Key takeaways**

CloudFlow framework	CloudBench microbenchmarks	Analysis of real-world apps
↓	↓	↓
Synchronous representation	CloudFlow passes 37 out of 40	11 vulnerabilities in 104 apps

- **Future work**

Higher number of services & APIs	Automated API docs analysis	Alternative static analysis tools