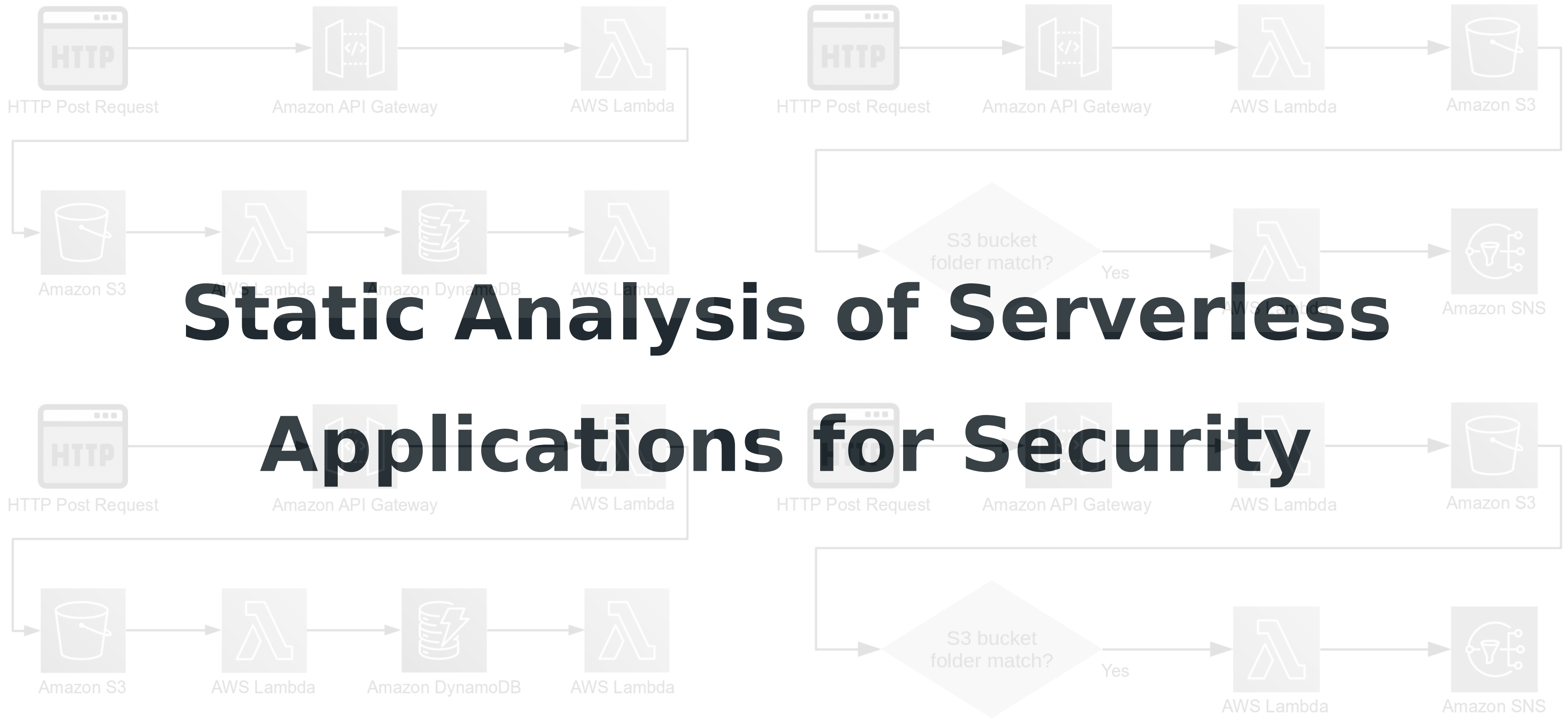


Static Analysis of Serverless Applications for Security

Speaker: Giuseppe Raffa

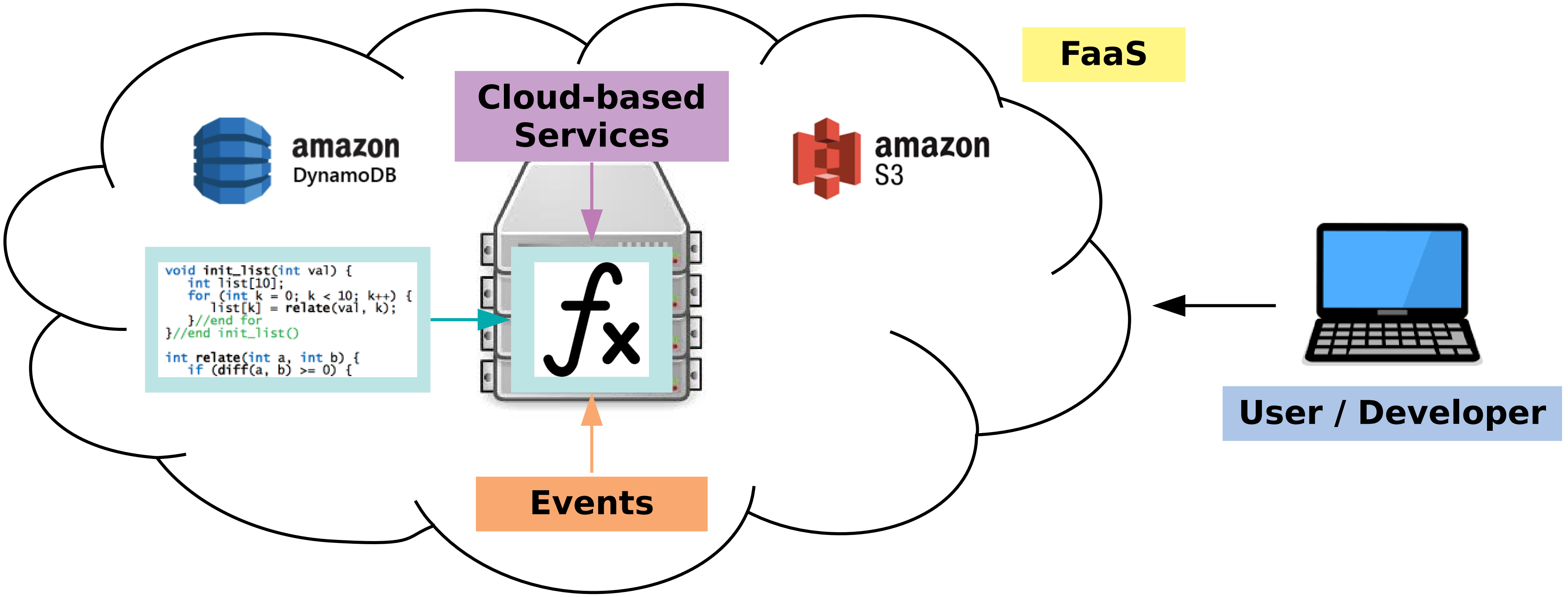
Co-authors: J. Blasco, D. O'Keeffe, S. K. Dash

CDT Showcase Day, 24th April 2025

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

ROYAL HOLLOWAY UNIVERSITY Of LONDON

Serverless Computing Model



```
void init_list(int val) {
   int list[10];
   for (int k = 0; k < 10; k++) {
      list[k] = relate(val, k);
   }//end for
}//end init_list()

int relate(int a, int b) {
   if (diff(a, b) >= 0) {
```

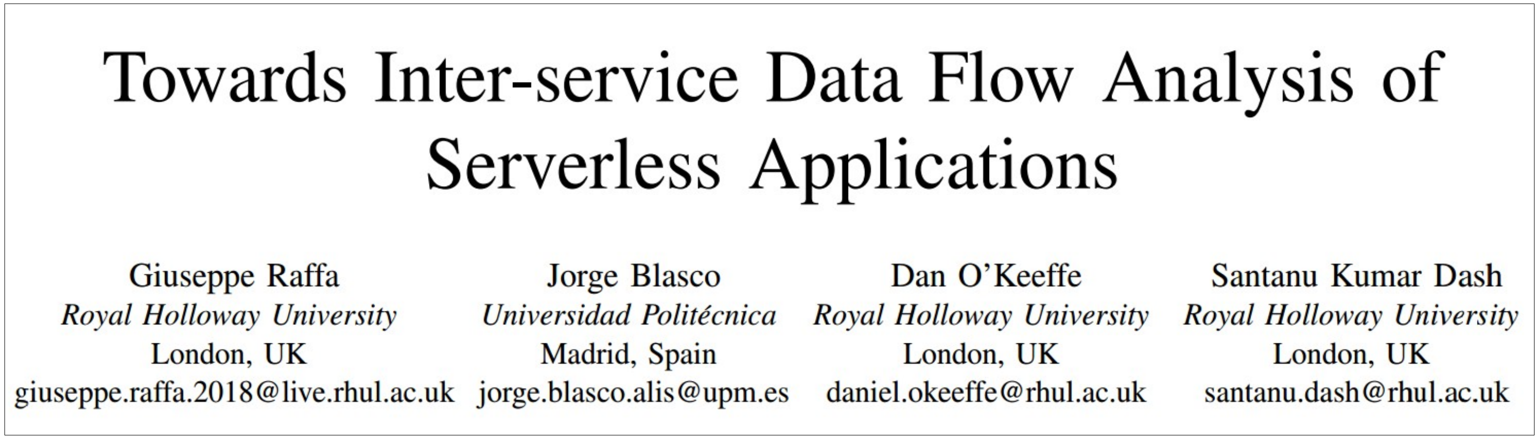
- **Advantage**
  - No infrastructure management
- **Challenge**
  - Security

Critical Risks for Serverless

- Risks identified by the Cloud Security Alliance

Function Event Data Injection	Broken Authentication	Insecure Serverless Deployment Config.
Over-Privileged Functions & Roles	Inadequate Function Monitoring	Insecure Third-Party Dependencies
Insecure Application Secrets Storage	DOS & Financial Resource Exhaustion	Business Logic Manipulation
Improper Exception Handling	Obsolete Functions, Resources & Events	Cross-Execution Data Persistency

SANER 2024 Paper



Towards Inter-service Data Flow Analysis of Serverless Applications

Giuseppe Raffa
Royal Holloway University
London, UK
giuseppe.raffa.2018@live.rhul.ac.uk

Jorge Blasco
Universidad Politécnica
Madrid, Spain
jorge.blasco.alis@upm.es

Dan O'Keeffe
Royal Holloway University
London, UK
daniel.okeeffe@rhul.ac.uk

Santanu Kumar Dash
Royal Holloway University
London, UK
santanu.dash@rhul.ac.uk

- **Results**

Infrastructure & Application code → **Automated code instrumentation** → **General-purpose tool**

Security-oriented microbenchmarks

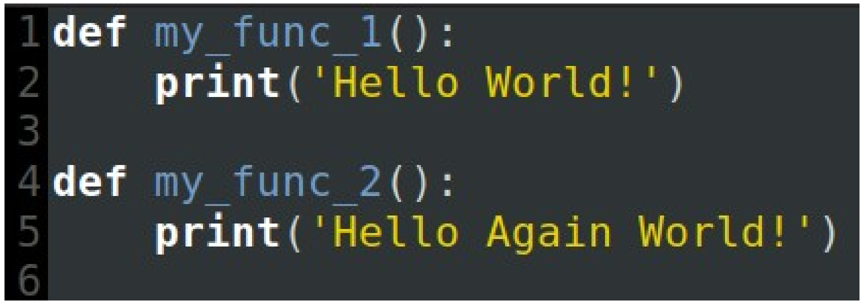
Analysis of large dataset

New paper under review

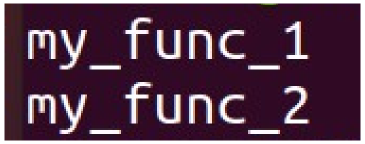
ROYAL HOLLOWAY UNIVERSITY OF LONDON

AST-based Processing (1)

- **Extraction of function names**

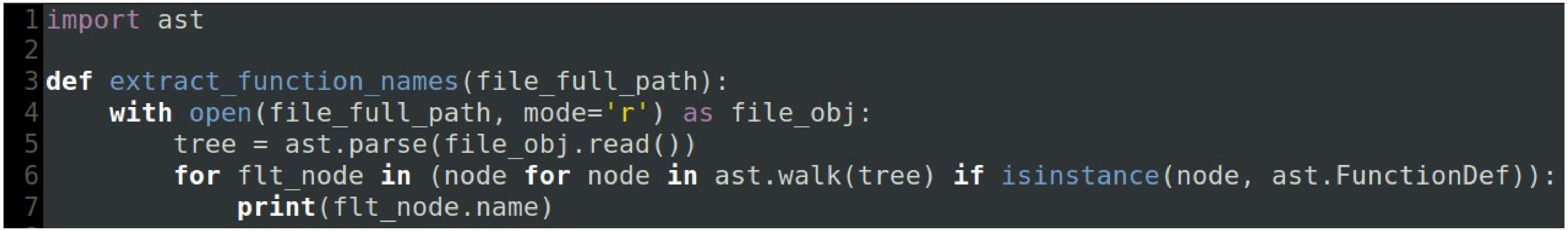


```python
def my_func_1():
    print('Hello World!')

def my_func_2():
    print('Hello Again World!')

```



```
my_func_1
my_func_2
```

- **Implementation**



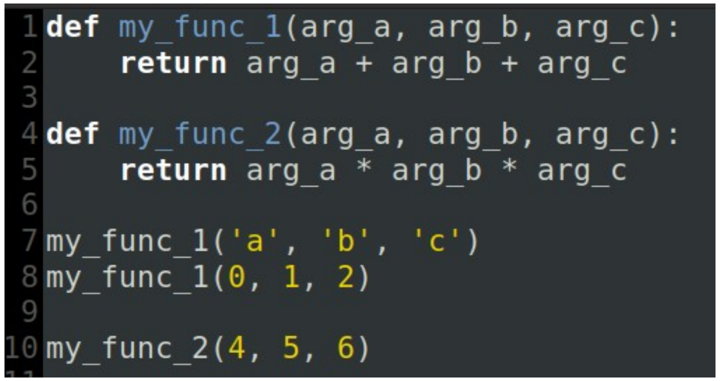
```python
import ast

def extract_function_names(file_full_path):
    with open(file_full_path, mode='r') as file_obj:
        tree = ast.parse(file_obj.read())
        for flt_node in (node for node in ast.walk(tree) if isinstance(node, ast.FunctionDef)):
            print(flt_node.name)
```

Create in-memory data structure with AST

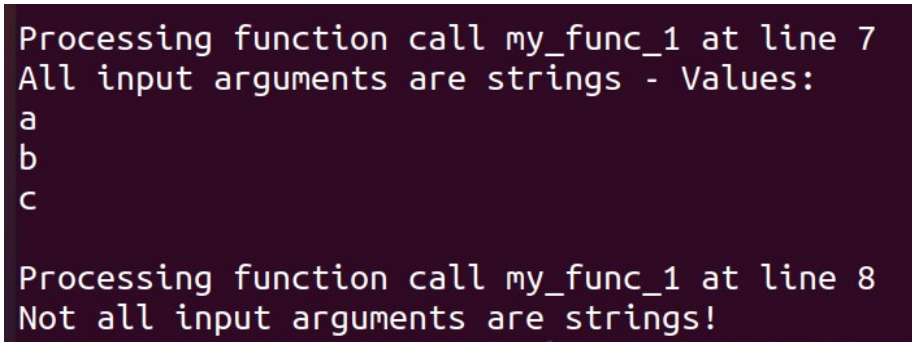
AST nodes inspection (ast.FunctionDef)

AST-based Processing (2)

- **Processing of function call arguments**

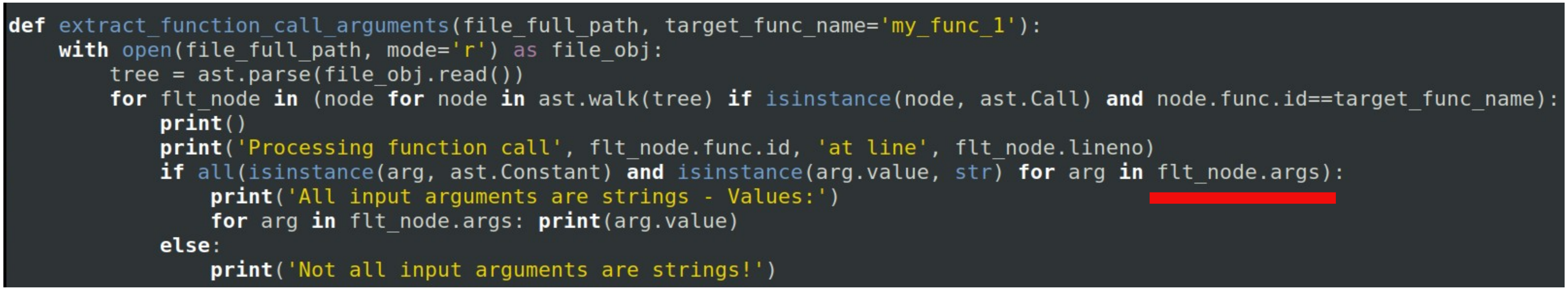


```python
def my_func_1(arg_a, arg_b, arg_c):
    return arg_a + arg_b + arg_c

def my_func_2(arg_a, arg_b, arg_c):
    return arg_a * arg_b * arg_c

my_func_1('a', 'b', 'c')
my_func_1(0, 1, 2)

my_func_2(4, 5, 6)
```



```
Processing function call my_func_1 at line 7
All input arguments are strings - Values:
a
b
c

Processing function call my_func_1 at line 8
Not all input arguments are strings!
```

- **Implementation**

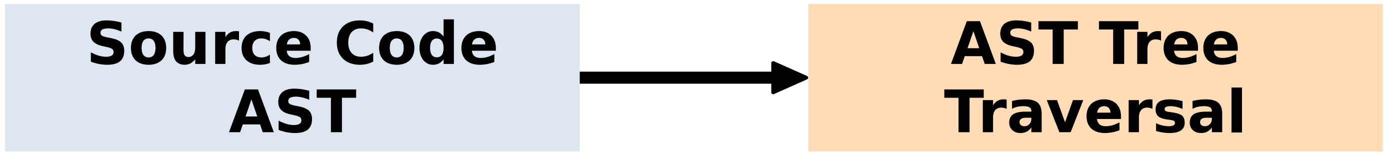


```python
def extract_function_call_arguments(file_full_path, target_func_name='my_func_1'):
    with open(file_full_path, mode='r') as file_obj:
        tree = ast.parse(file_obj.read())
        for flt_node in (node for node in ast.walk(tree) if isinstance(node, ast.Call) and node.func.id==target_func_name):
            print()
            print('Processing function call', flt_node.func.id, 'at line', flt_node.lineno)
            if all(isinstance(arg, ast.Constant) and isinstance(arg.value, str) for arg in flt_node.args):
                print('All input arguments are strings - Values:')
                for arg in flt_node.args: print(arg.value)
            else:
                print('Not all input arguments are strings!')
```

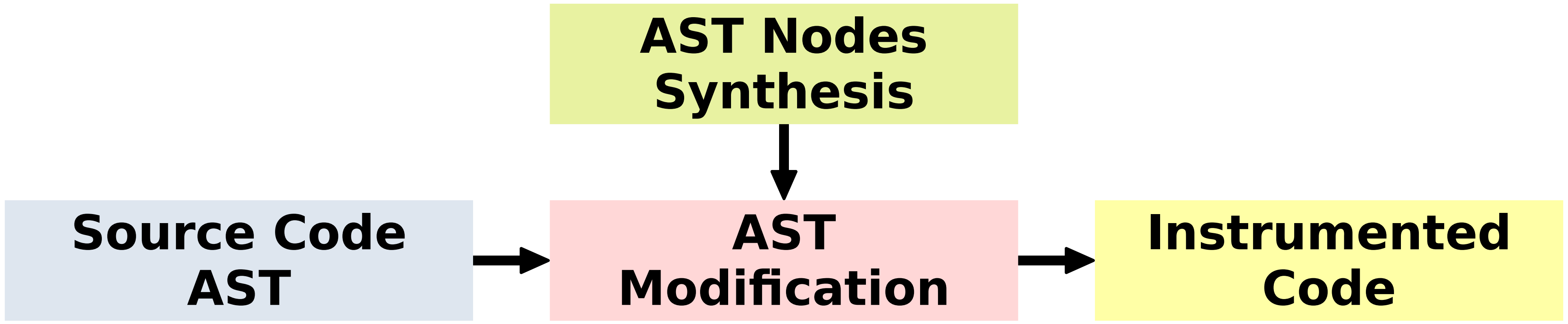
AST nodes inspection (ast.Call)

AST-based Approach

- Extraction of information



- Source code modification



Conclusion

- **PhD overview**

Security-sensitive data flows	Novel suite of microbenchmarks	Analysis of large dataset

- **Questions?**

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

https://github.com/giusepperaffa

PyCon Wroclaw