Betrusted

Finding security bugs in your code with semantic analysis

betrusted.it

Davide Ornaghi

Offensive Security Specialist – Co-founder @ Betrusted

- Penetration Tester
- Linux Vulnerability Researcher
- Speaker at NoHat and HITB

Betrusted

Static Analysis

"A process that allows you to analyze an application's code for potential errors without executing the code itself."

Finding security bugs in your code with semantic analysis

Grep –E vuln sw.c?

01 GREP

Betrusted

Finding security bugs in your code with semantic analysis

Grep –E vuln sw.c?

01 GREP

02 LEXICAL ANALYSIS (TOKENS)

Betrusted

Finding security bugs in your code with semantic analysis

Grep –E vuln sw.c?

01 GREP

02 LEXICAL ANALYSIS (TOKENS)

03 SYNTACTIC ANALYSIS (AST)

Betrusted

Finding security bugs in your code with semantic analysis

Grep –E vuln sw.c?

01 GREP

02 LEXICAL ANALYSIS (TOKENS)

03 SYNTACTIC ANALYSIS (AST)

04 SEMANTIC ANALYSIS (CFG)

Betrusted

Tokens

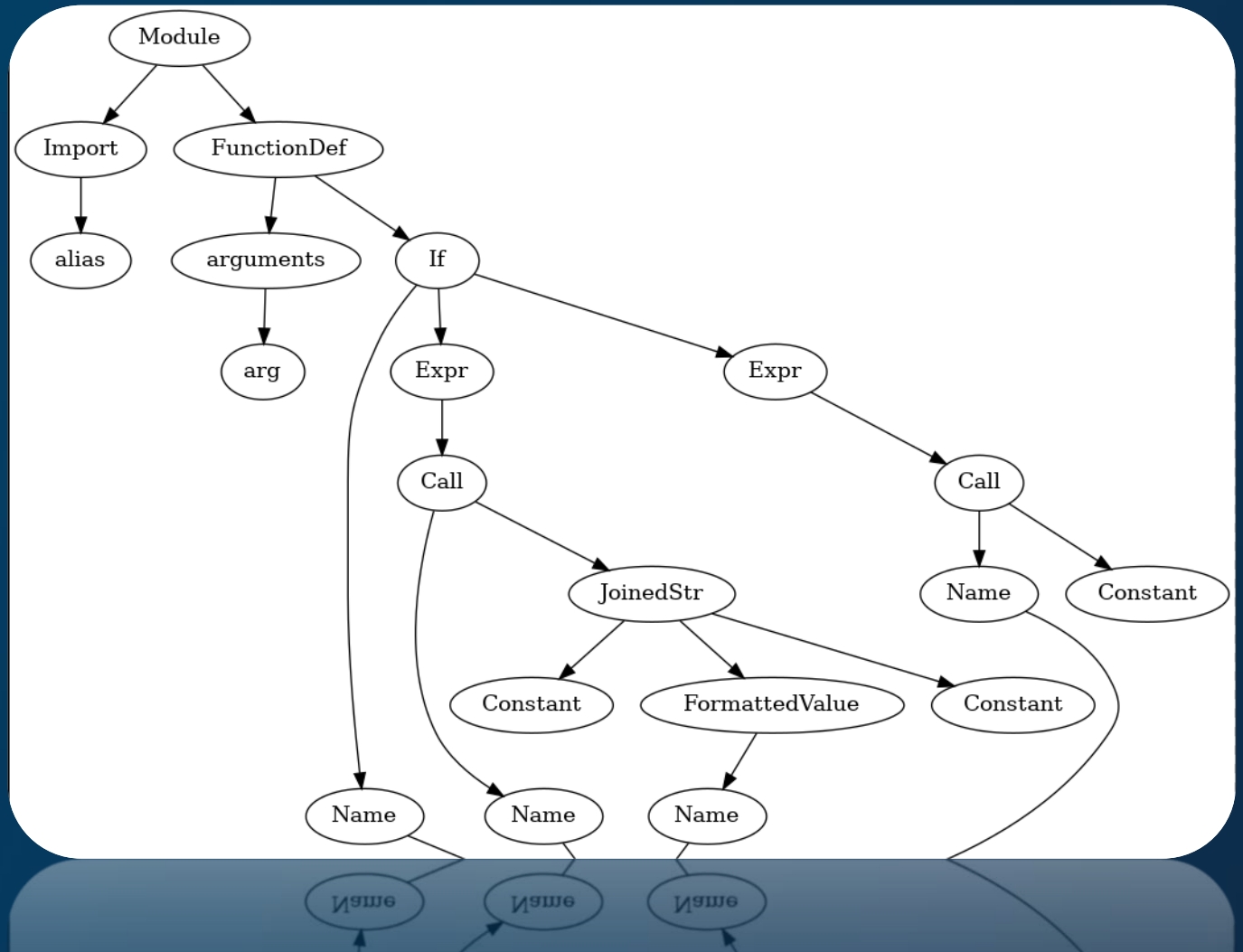
```
X = 42
```

```
TokenInfo(type=62 (ENCODING), string='utf-8', start=(0, 0), end=(0, 0), line='')
TokenInfo(type=1 (NAME), string='x', start=(1, 0), end=(1, 1), line='x = 42')
TokenInfo(type=54 (OP), string='=', start=(1, 2), end=(1, 3), line='x = 42')
TokenInfo(type=2 (NUMBER), string='42', start=(1, 4), end=(1, 6), line='x = 42')
TokenInfo(type=4 (NEWLINE), string='', start=(1, 6), end=(1, 7), line='x = 42')
TokenInfo(type=0 (ENDMARKER), string='', start=(2, 0), end=(2, 0), line='')
```

Finding security bugs in your code with semantic analysis

Abstract Syntax Trees

```python
import math
def greet(who):
    if who:
        print(f"Hello, {who}!")
    else:
        print("Hello, World!")
```



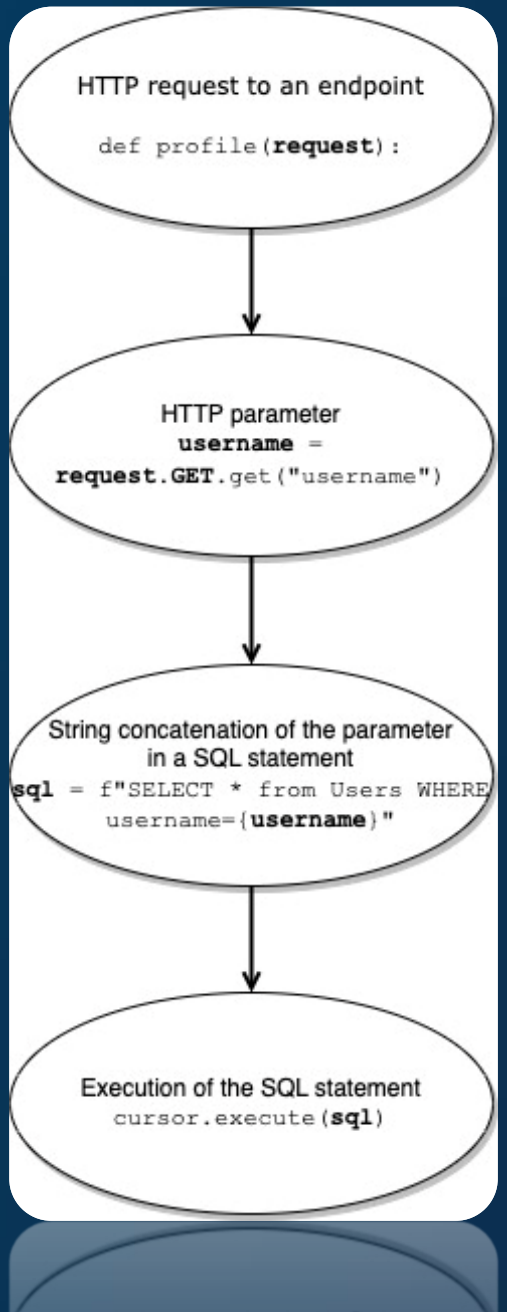
Betrusted

Finding security bugs in your code with semantic analysis

Control Flow Graphs

```python
from django.db import connection

def profile(request):
    with connection.cursor() as cursor:
        username = request.GET.get("username")
        sql = f"SELECT * FROM users WHERE username={username}"
        cursor.execute(sql)
```



Betrusted

Exercise 1: Syntactic Analysis of C/C++ code

- Spotting the vulnerability
- Writing a *weggli* query
- Creating a CodeQL database
- Setting up CodeQL and query pack in VSCode
- Finding all interesting memcpy calls

DataFlow and TaintTracking

- CFGs describe the flow of control, by giving ASTs an *ordering relation*

DataFlow and TaintTracking

- CFGs describe the flow of control, by giving ASTs an *ordering relation*
- CFGs allow specifying untrusted inputs (**sources**) and dangerous targets (**sinks**), and finding the connection between them

DataFlow and TaintTracking

- CFGs describe the flow of control, by giving ASTs an *ordering relation*
- CFGs allow specifying untrusted inputs (**sources**) and dangerous targets (**sinks**), and finding the connection between them
- DataFlow analysis only tracks value-preserving data

DataFlow and TaintTracking

- CFGs describe the flow of control, by giving ASTs an *ordering relation*
- CFGs allow specifying untrusted inputs (**sources**) and dangerous targets (**sinks**), and finding the connection between them
- DataFlow analysis only tracks value-preserving data
- TaintTracking marks certain inputs as *tainted* and follows their propagation

DataFlow and TaintTracking

- CFGs describe the flow of control, by giving ASTs an *ordering relation*
- CFGs allow specifying untrusted inputs (**sources**) and dangerous targets (**sinks**), and finding the connection between them
- DataFlow analysis only tracks value-preserving data
- TaintTracking marks certain inputs as *tainted* and follows their propagation
- We can also specify additional sanitizers (**taint steps**) to customize our scanning

Exercise 2: Semantic Analysis of Grafana (CVE-2021-43798)

- Testing out the exploit
- *Semgrep* rule with vulnerable pattern
- CodeQL query with Local Tracking
- CodeQL query with Global Tracking

Exercise 3: Semantic Analysis of Linux (C)

- Finding CVE-2023-0179 with Range Analysis
- Looking for Memory Corruption vulnerabilities with Guard Conditions

Implementing CodeQL in your CI/CD

- Fork a repository to scan:
  - https://github.com/GitHubSecurityLab/codeql-zero-to-hero
- Enable code scanning from the Security tab
- Specify queries to run and trigger events
- Check status in the Security tab
- Investigate single alerts
- Find out more queries: https://github.com/github/codeql/tree/main/cpp/ql/src

THANK YOU!

Q & A

Betrusted


Betrusted