Towards Efficient, Multi-Language Dynamic Taint Analysis

JOHANNES KEPLER UNIVERSITY LINZ

Jacob Kreindl, Daniele Bonetta, Hanspeter Mössenböck

ORACLE Labs

Dynamic Taint Analysis

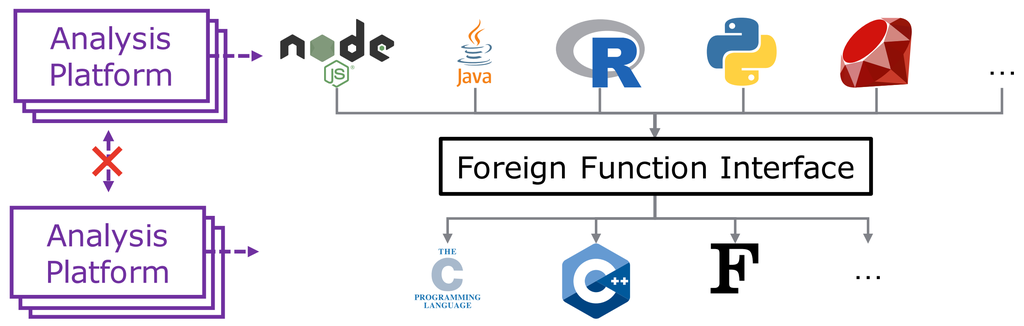
```
function requestHandler(request, response) {
    var code = Buffer.from(dangerousData());   (1)
    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");          (2)
    var html = Buffer.concat([lTag, code, rTag]);
    response.end(html);
}                                               (3)

var http = require('http')
http.createServer(requestHandler).listen(8080);
```

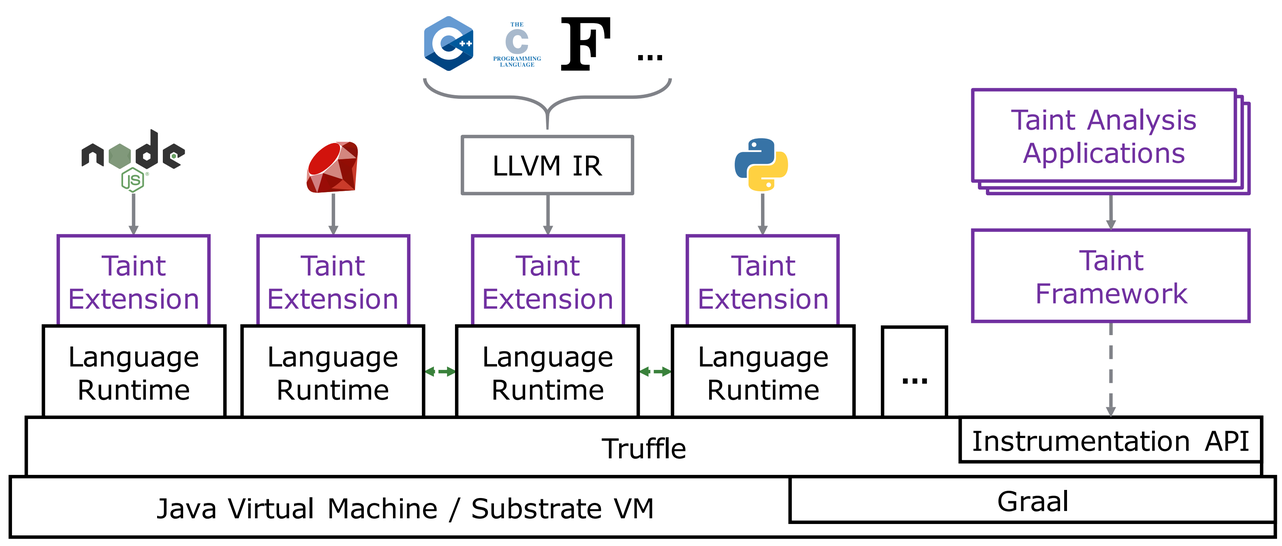
At Run Time

1. Add *Taint Label* to data originating from *Taint Sources*
2. *Propagate* taint labels together with the data
3. Detect tainted data in *Taint Sinks*

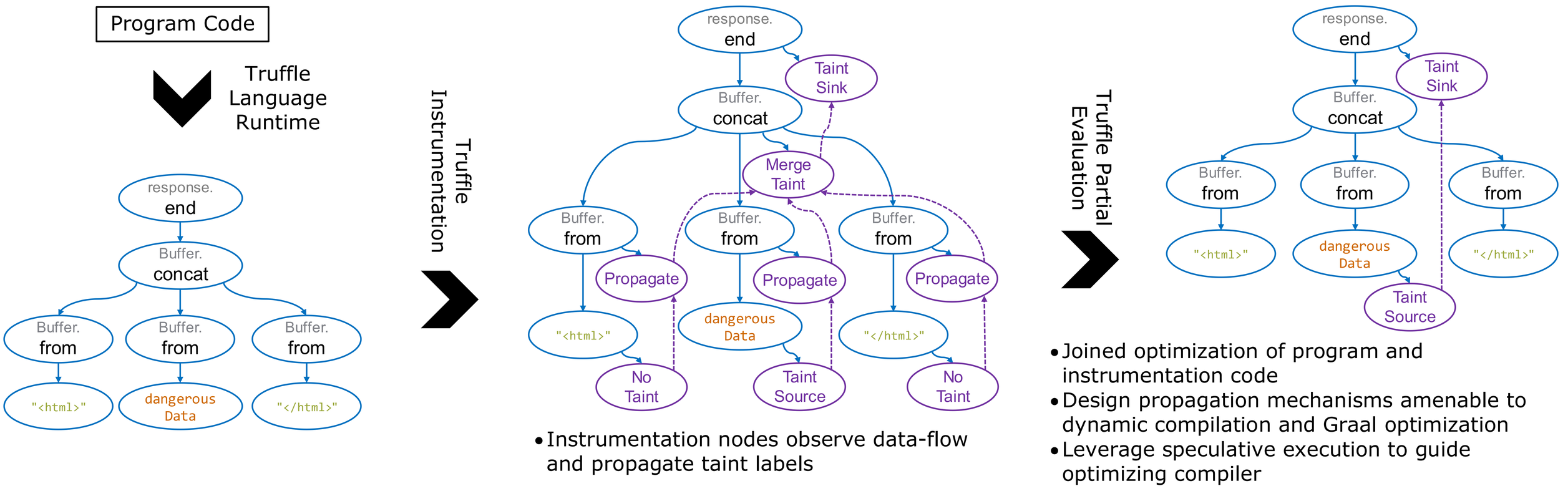
Problem: Language Boundary as Barrier for Taint Analysis



Proposed Solution: Polyglot Taint Analysis with GraalVM



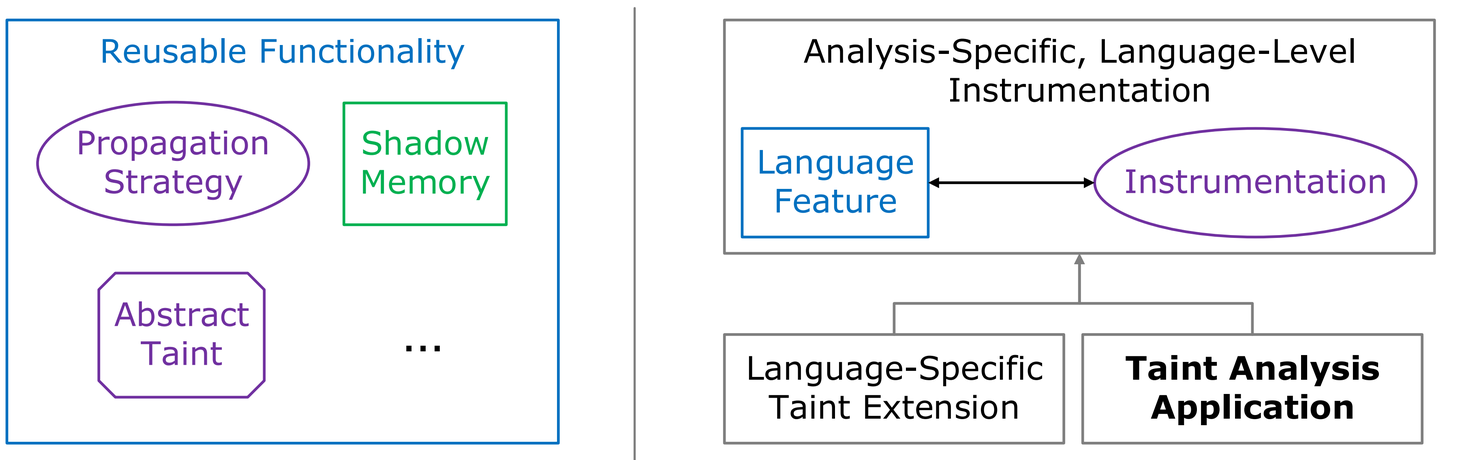
Truffle Instrumentation and Partial Evaluation for Efficient Taint Propagation

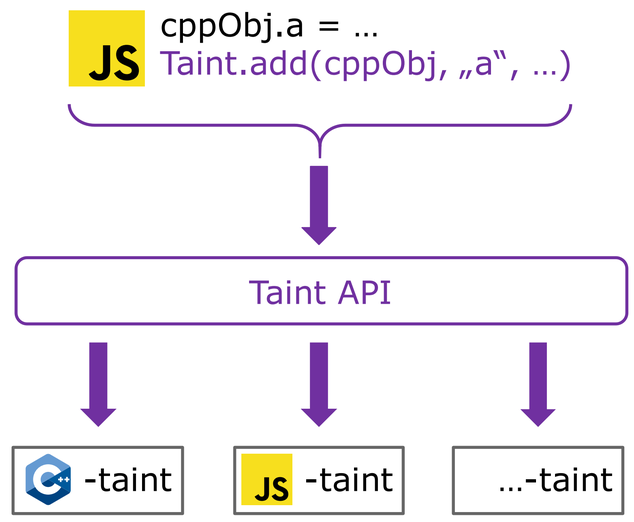


- Instrumentation nodes observe data-flow and propagate taint labels

- Joined optimization of program and instrumentation code
- Design propagation mechanisms amenable to dynamic compilation and Graal optimization
- Leverage speculative execution to guide optimizing compiler

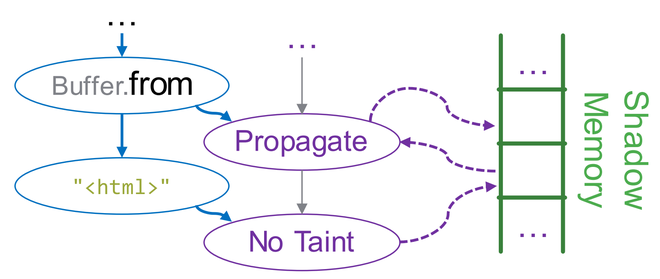
Extensible Framework for Multi-Language Dynamic Taint Analysis



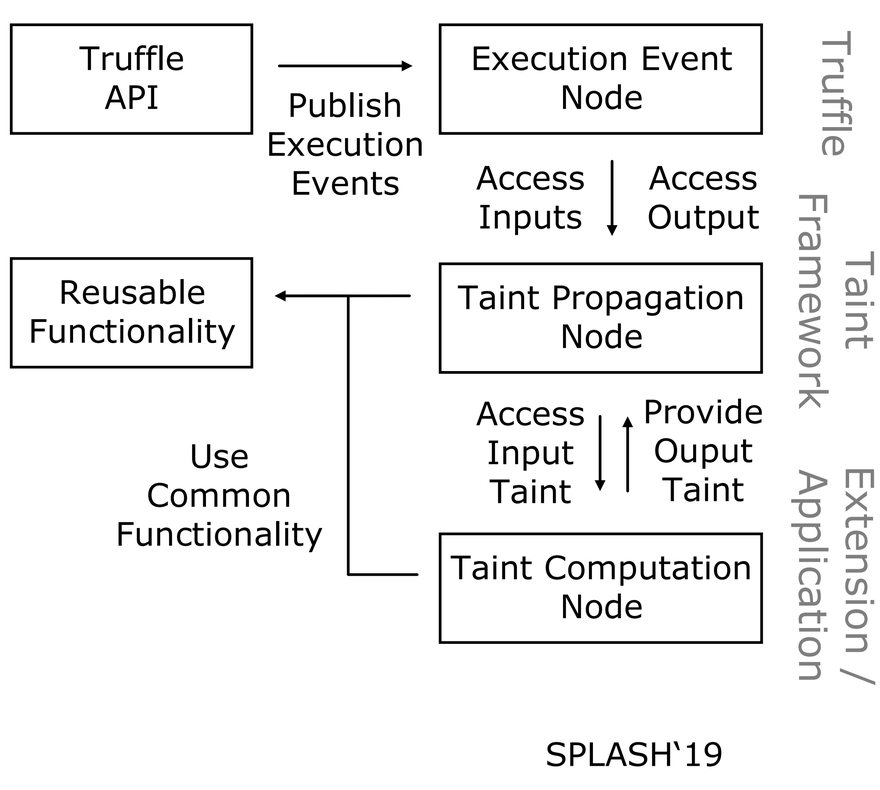


- Focused language extensions and analysis code
  - Choose suitable abstractions
  - Implement abstractions efficiently
- Adaptable language-level instrumentation
  - Adapt to define propagation semantics
    - Taint Sources and Sinks
    - Implicit flows and granularity
  - Analysis-defined taint labels
  - Code reuse across target languages
  - Language-agnostic taint analysis
- Integration of multiple languages
  - Language-specific storage strategy
  - Granularity of label attachment

Taint Propagation with Shadow Memory



Instrumentation Node Implementation



jacob.kreindl@jku.at @JKreindl https://github.com/jkreindl