Towards Efficient, Multi-Language Dynamic Taint Analysis

Jacob Kreindl, Daniele Bonetta, Hanspeter Mössenböck

16th International Conference on Managed Programming Languages and Runtimes

JOHANNES KEPLER UNIVERSITY LINZ

ORACLE Labs

Dynamic Taint Analysis

```
function requestHandler(request, response) {

    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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

Dynamic Taint Analysis

```
function requestHandler(request, response) {

    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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

Add taint label to sensitive data

Dynamic Taint Analysis

```
function requestHandler(request, response) {

    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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

Add taint label to sensitive data

Propagate the taint label with the data

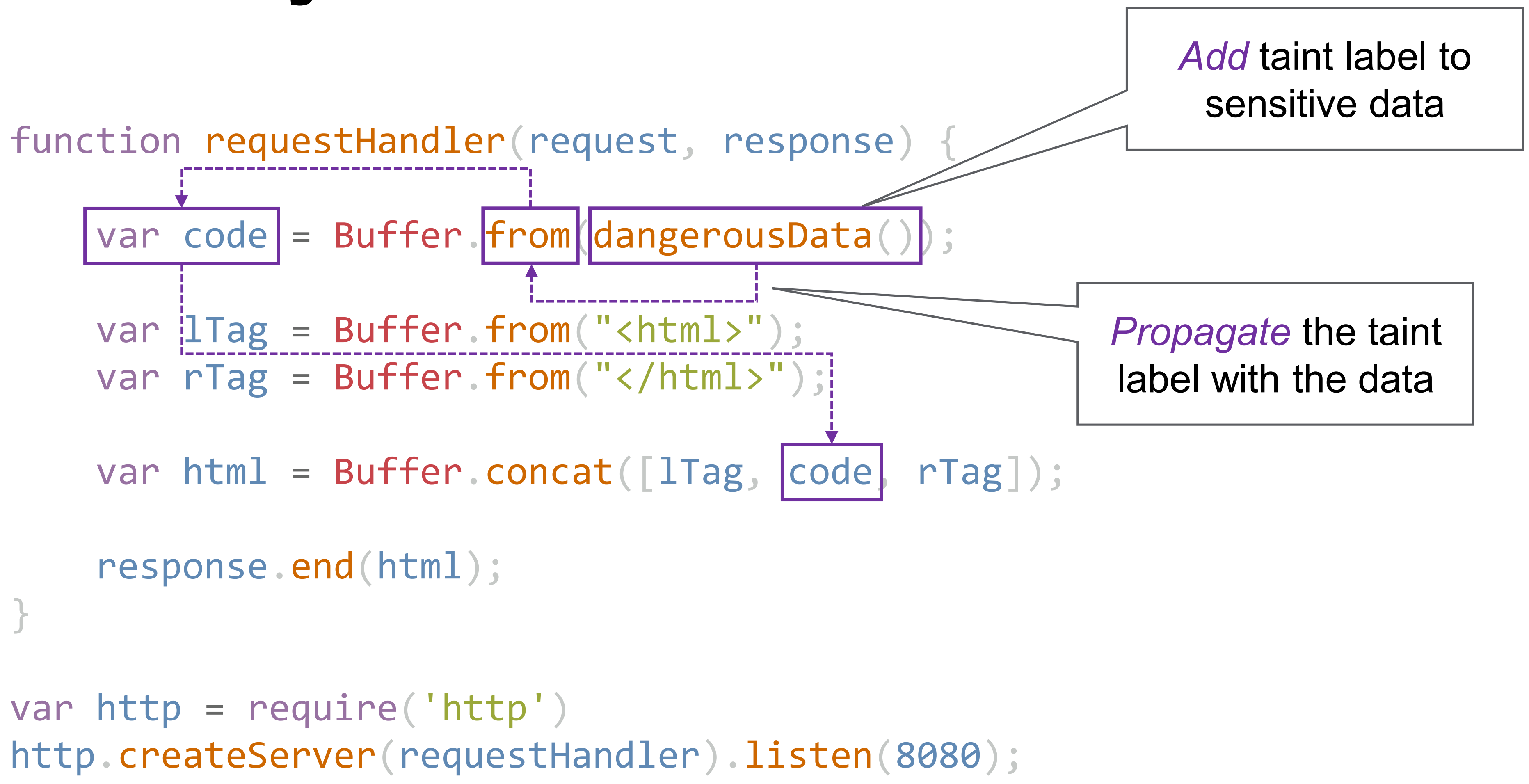
Dynamic Taint Analysis

```
function requestHandler(request, response) {
    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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

Add taint label to sensitive data

Propagate the taint label with the data

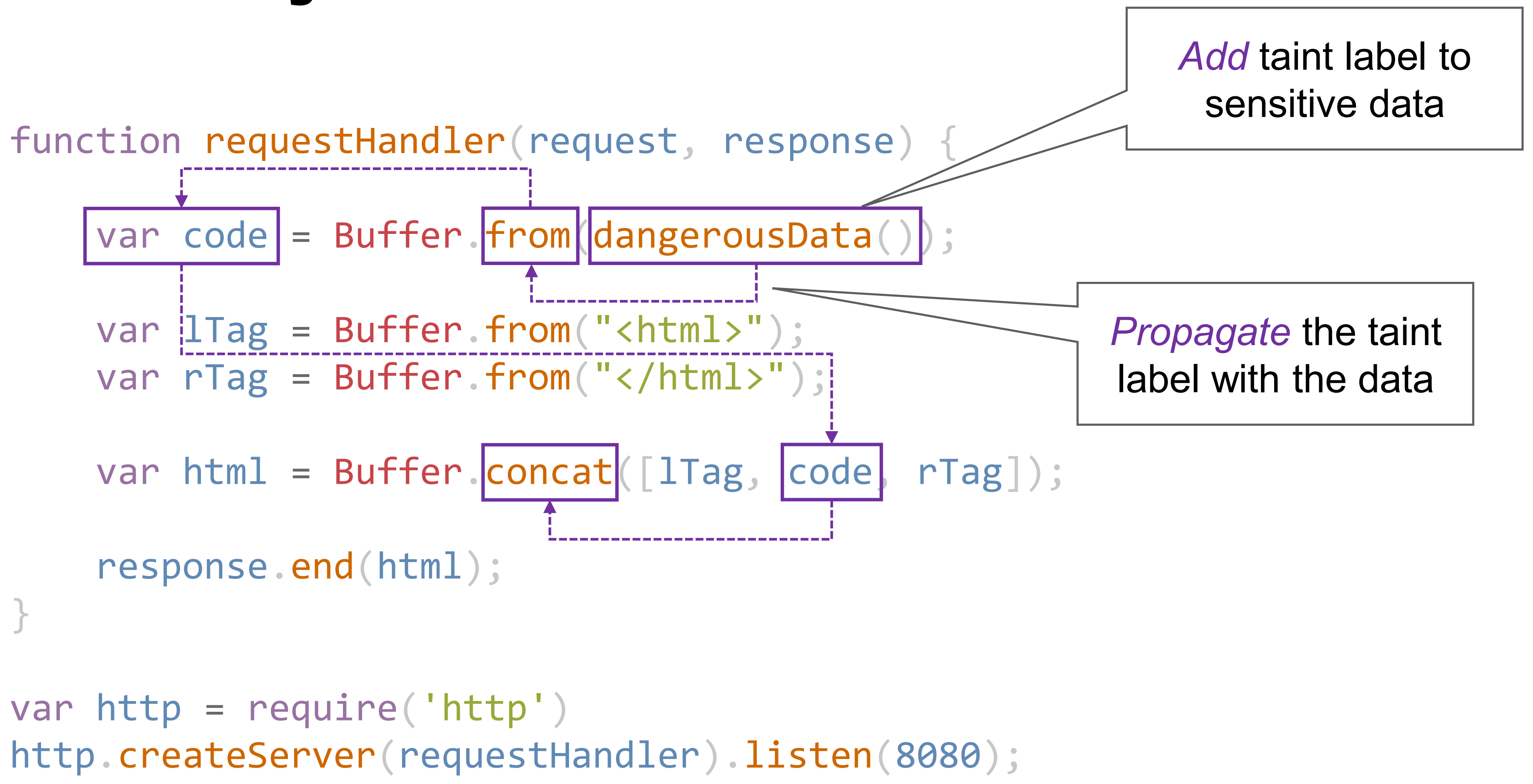
Dynamic Taint Analysis



```
function requestHandler(request, response) {
    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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

Add taint label to sensitive data
Propagate the taint label with the data

Dynamic Taint Analysis

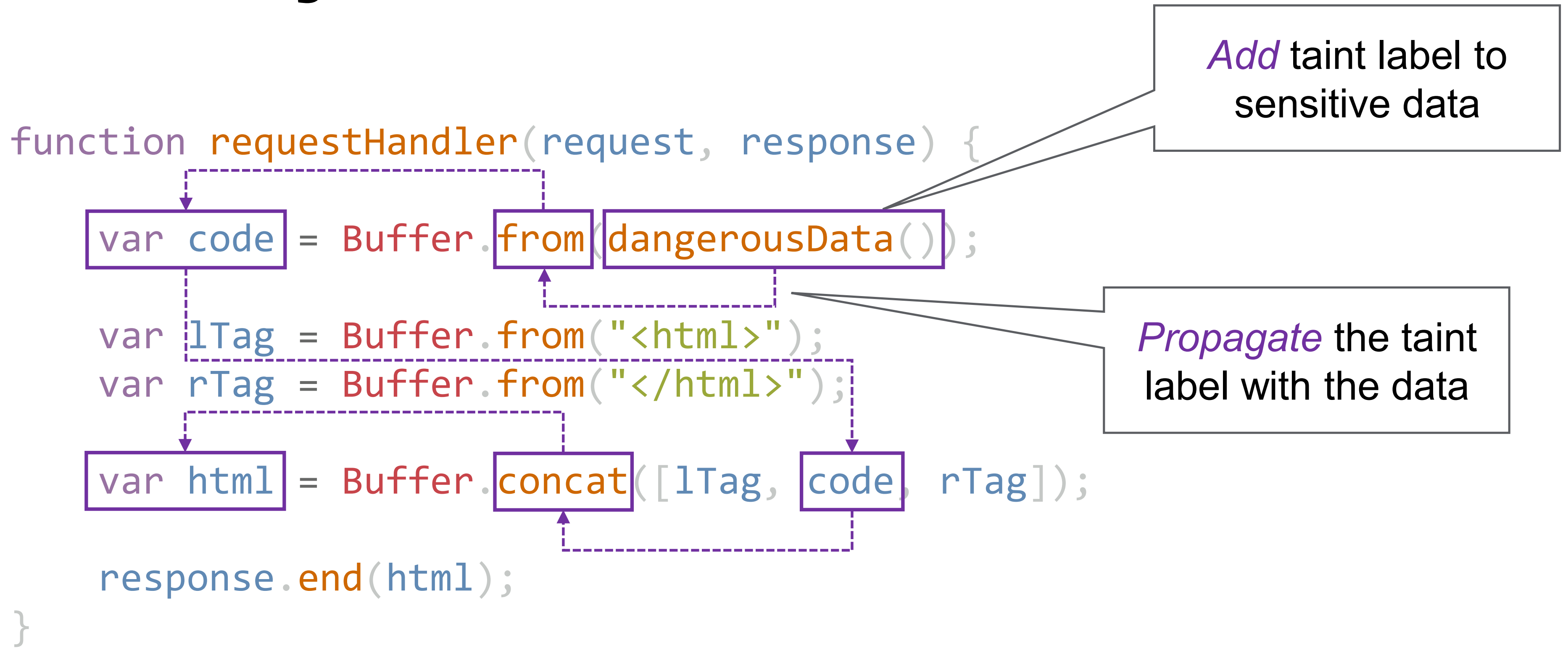


```
function requestHandler(request, response) {

    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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

Add taint label to sensitive data

Propagate the taint label with the data

Dynamic Taint Analysis



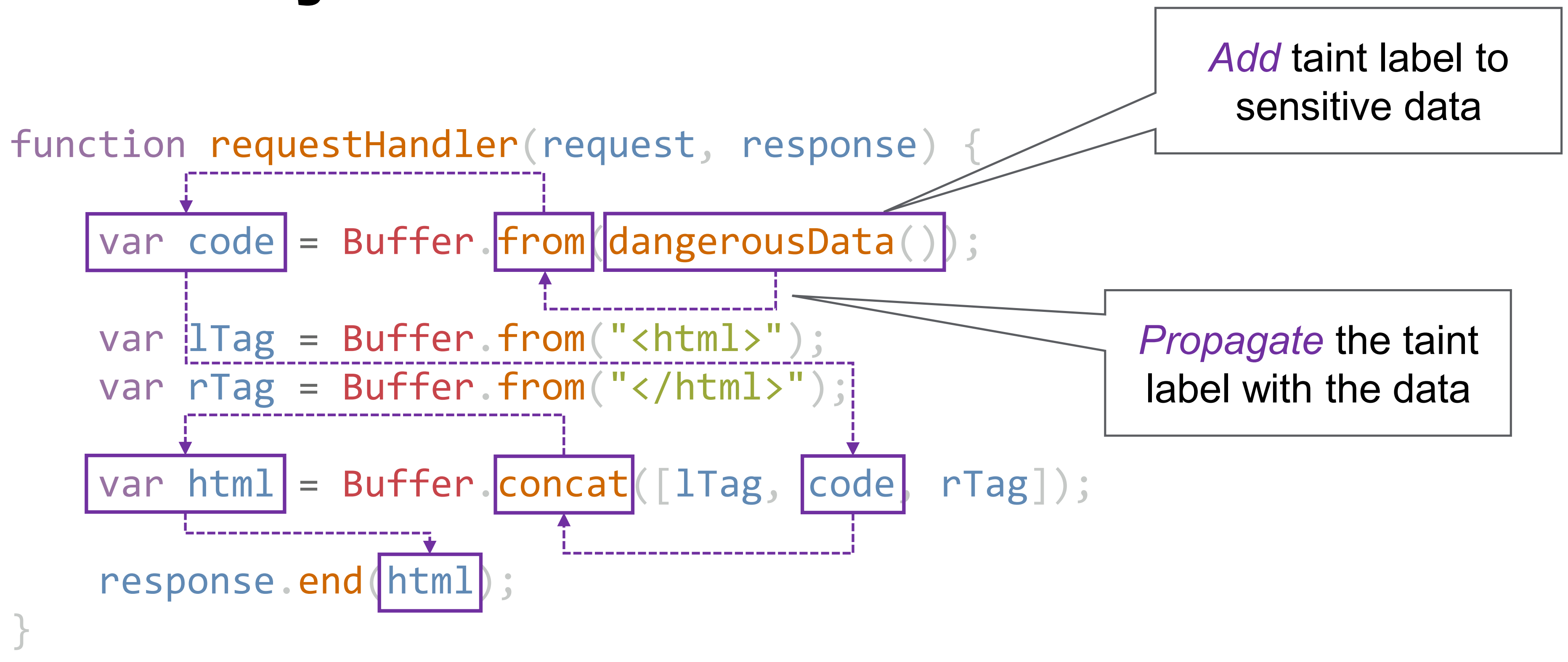
```
function requestHandler(request, response) {

    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}
```

Add taint label to sensitive data

Propagate the taint label with the data

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

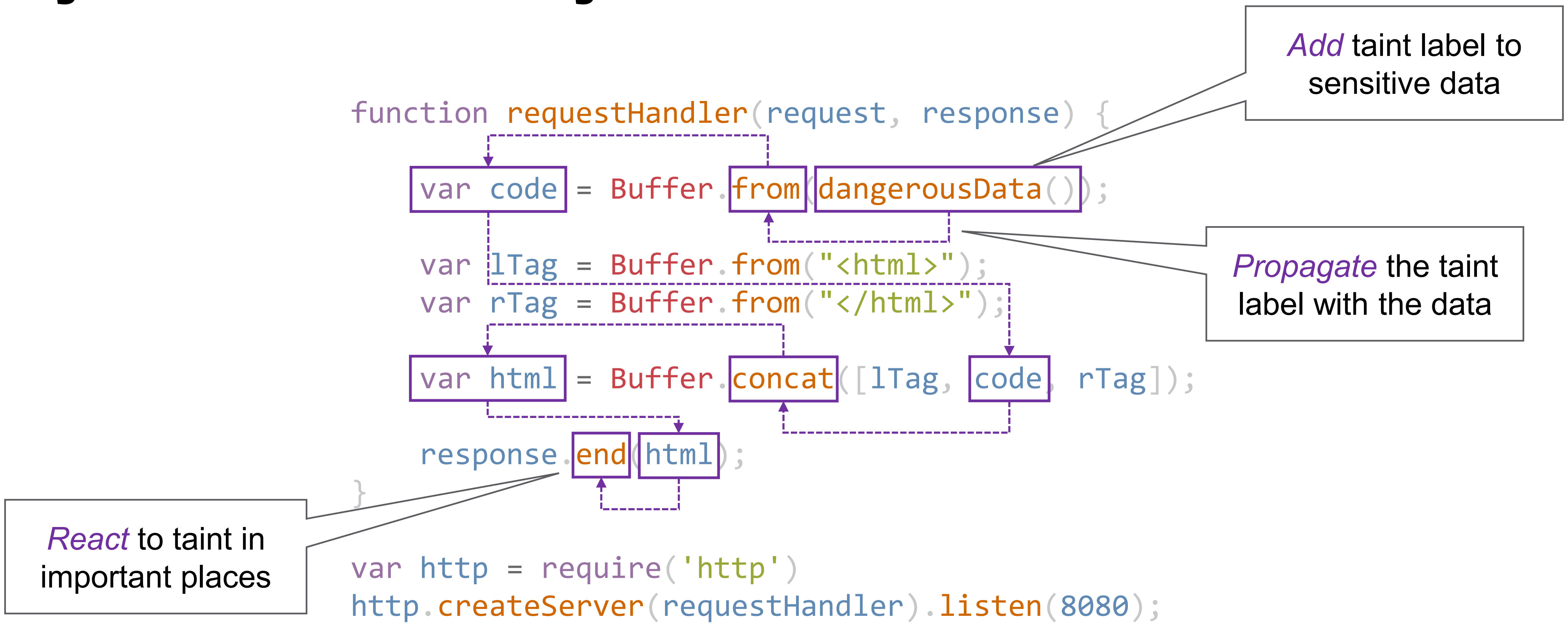
Dynamic Taint Analysis



```
function requestHandler(request, response) {
    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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

Add taint label to sensitive data
Propagate the taint label with the data

Dynamic Taint Analysis



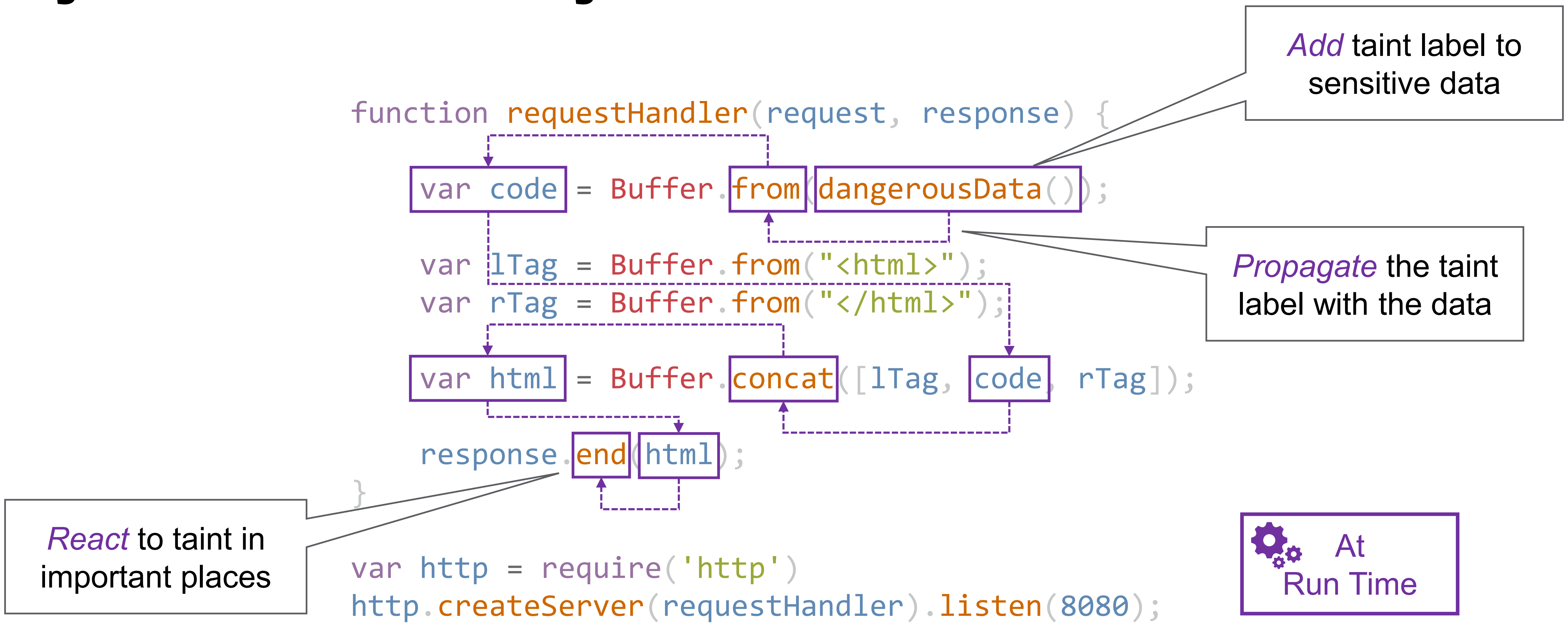
Add taint label to sensitive data

Propagate the taint label with the data

React to taint in important places

```
function requestHandler(request, response) {
    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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

Dynamic Taint Analysis



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

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

Add taint label to sensitive data

Propagate the taint label with the data

React to taint in important places

At Run Time

Applications

```
function requestHandler(request, response) {

    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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

Applications

Prevent Data Leaks

„<body>secret: … </body>"

```
function requestHandler(request, response) {

    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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

Applications

Prevent Data Leaks

„<body>secret: … </body>"

Trace Origin of Data

from: dangerousData();

```
function requestHandler(request, response) {

    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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

Applications

```
function requestHandler(request, response) {

    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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

Prevent Data Leaks

```
„<body>secret: … </body>"
```

Trace Origin of Data

```
from: dangerousData();
```

Detect vulnerabilities

```
<html>
<body onLoad=\"installMalware();\"/>
</html>
```

Language Embeddings

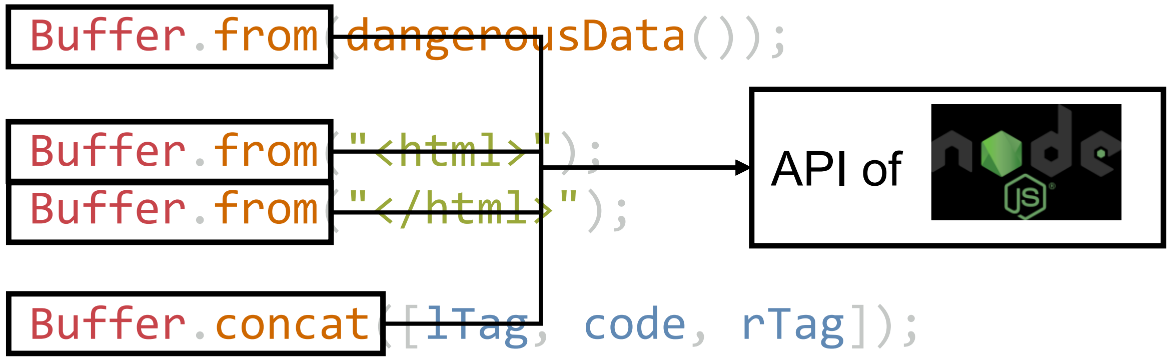
```
function requestHandler(request, response) {
    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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

Language Embeddings

```
function requestHandler(request, response) {
    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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

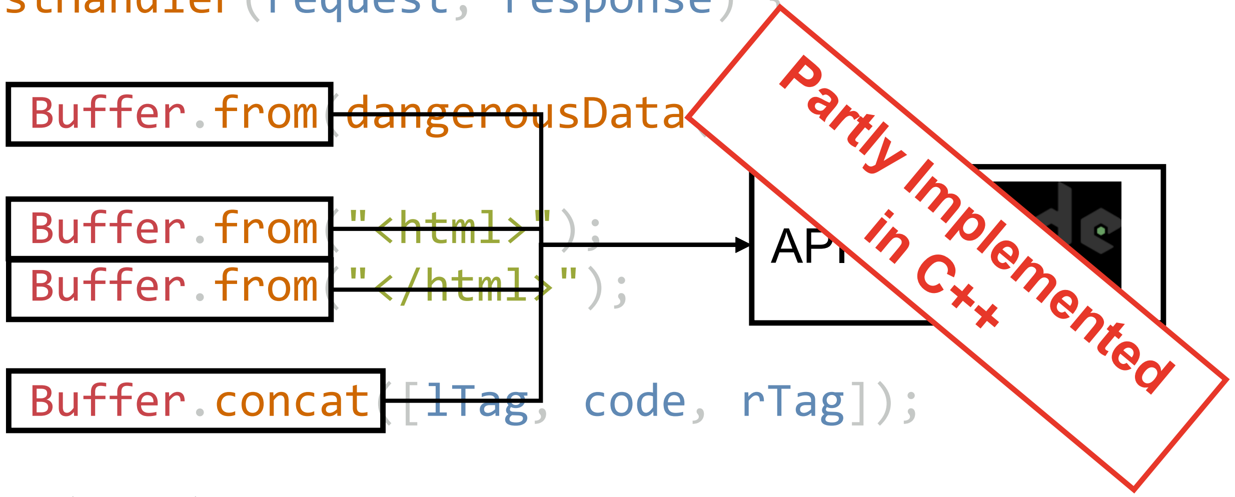
Language Embeddings

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

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

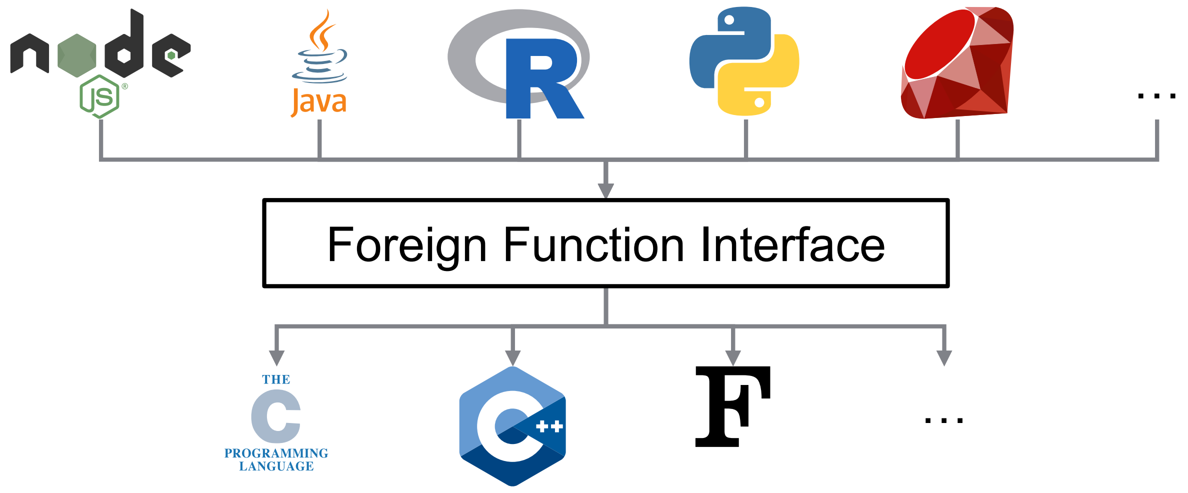


Language Embeddings

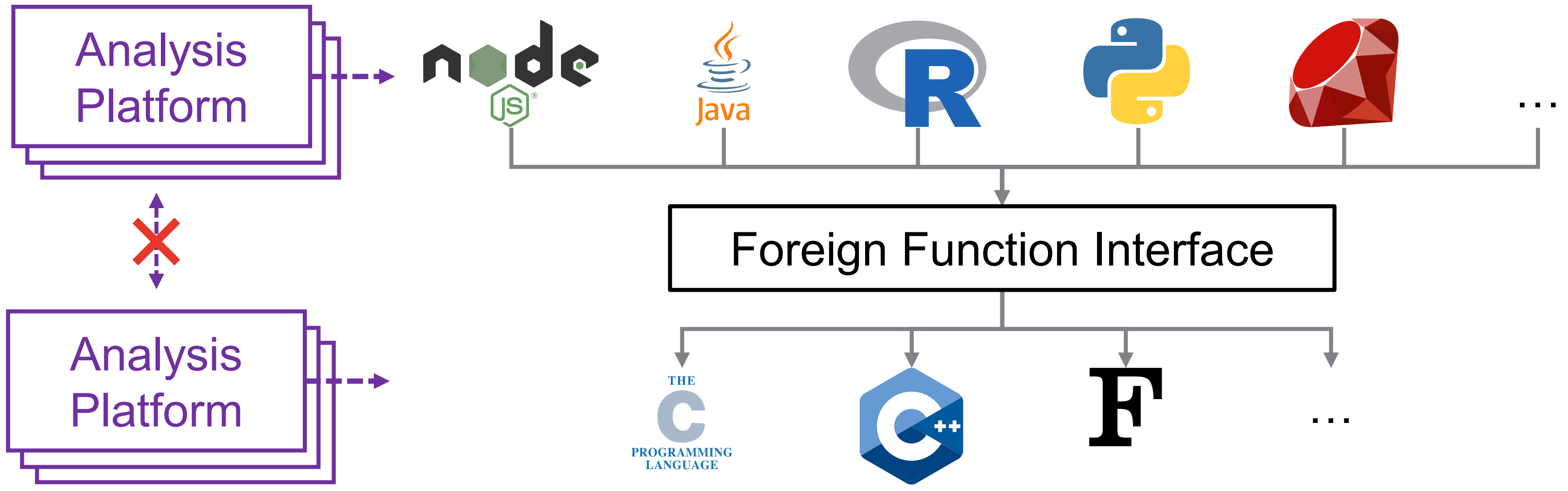
```
function requestHandler(request, response) {

    var code = Buffer.from(dangerousData);

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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



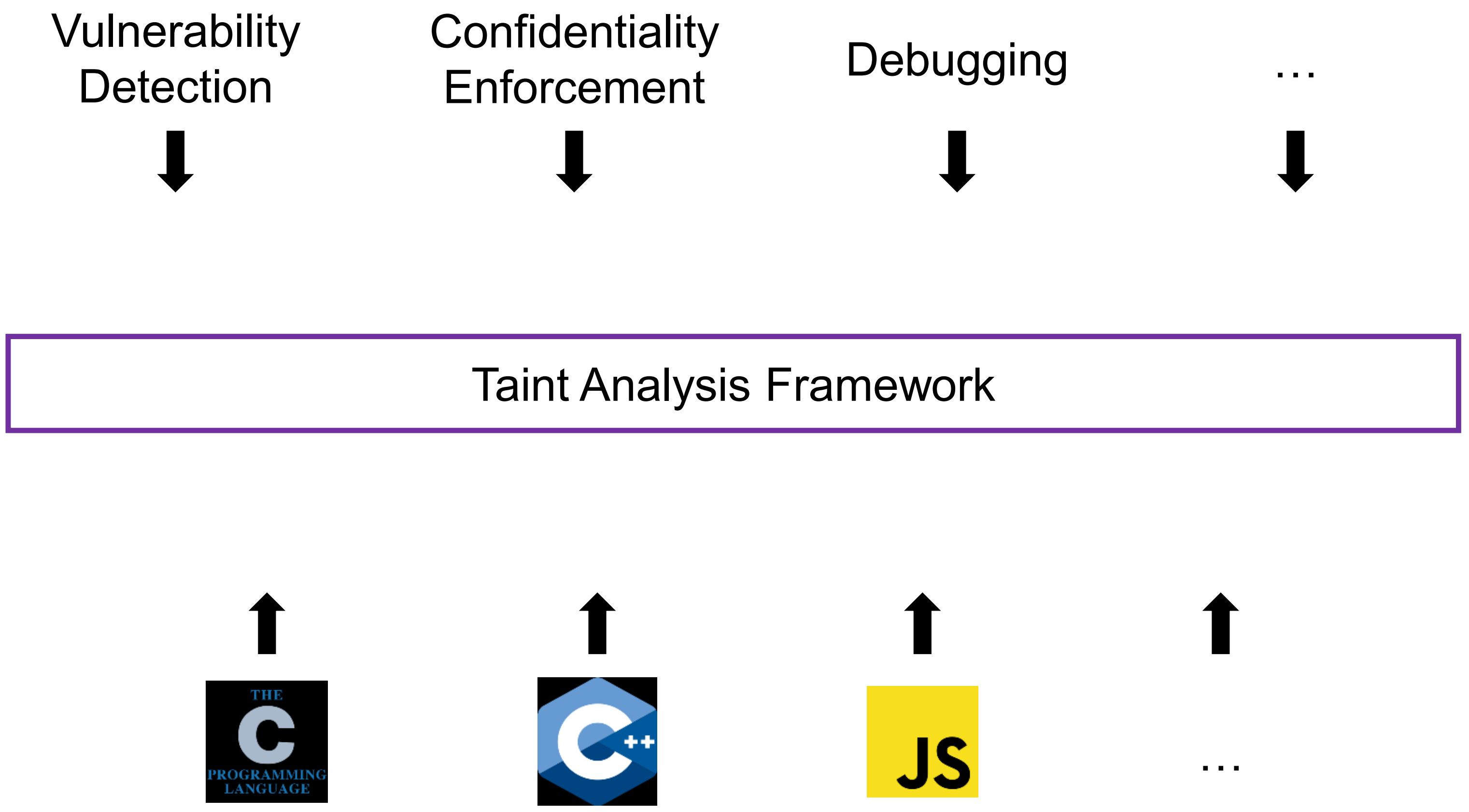
Language Interaction



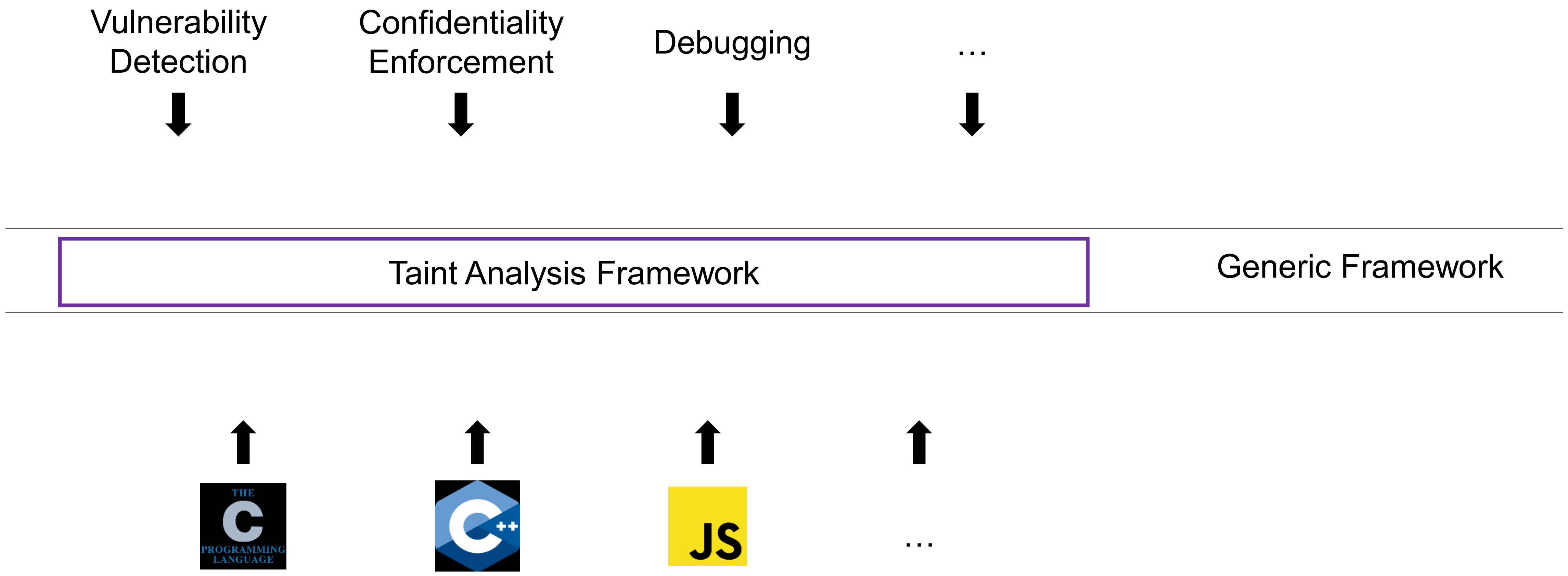
Language Interaction



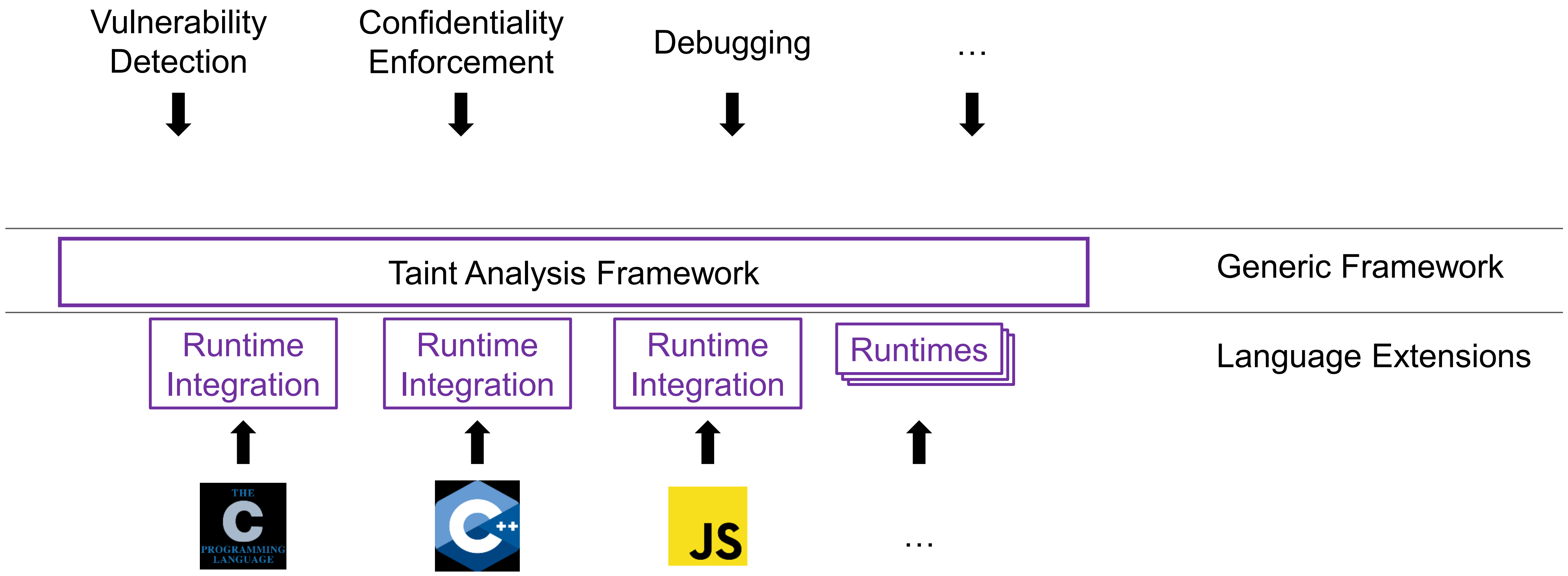
A Generic Platform for Multi-Language Dynamic Taint Analysis



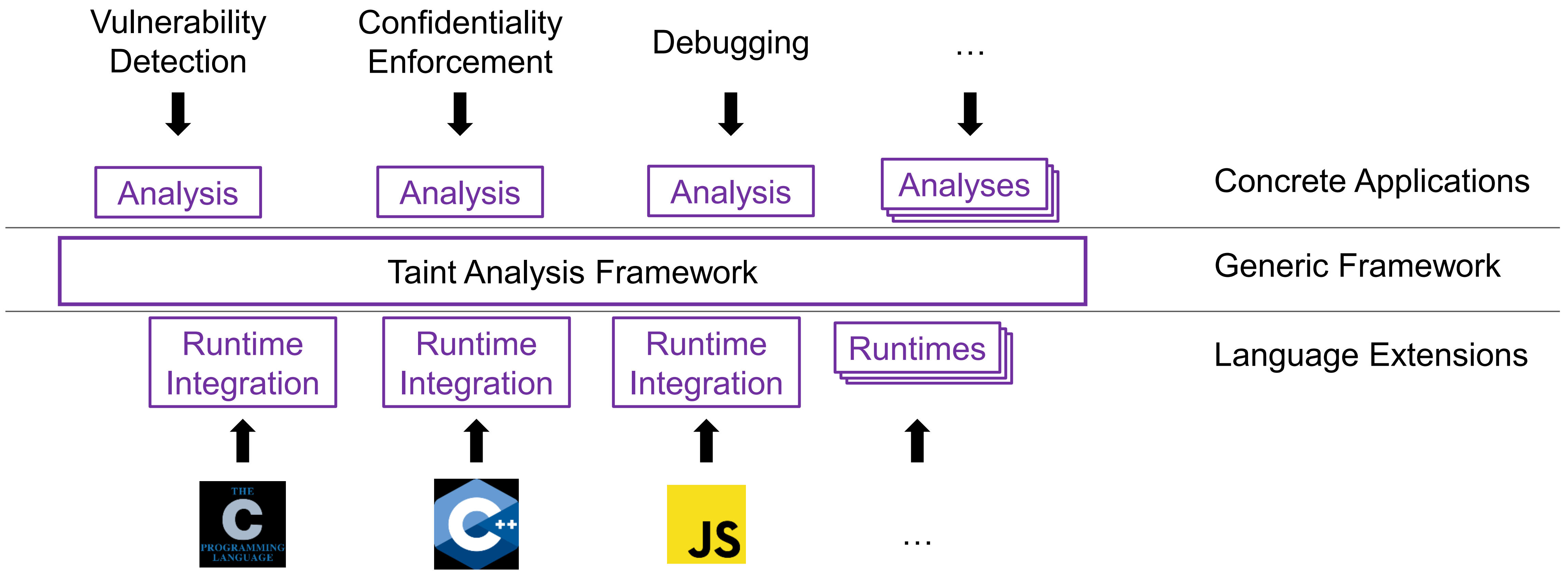
A Generic Platform for Multi-Language Dynamic Taint Analysis



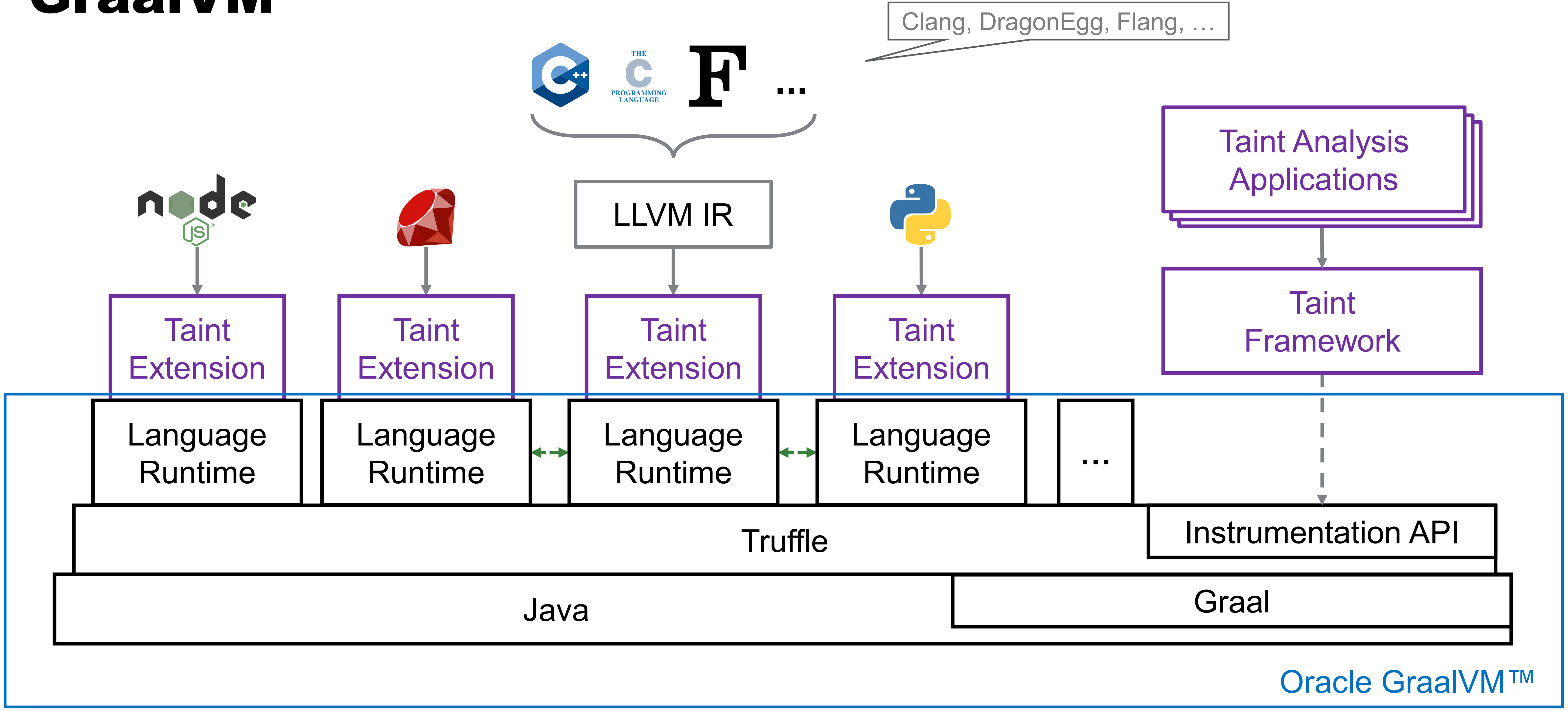
A Generic Platform for Multi-Language Dynamic Taint Analysis



A Generic Platform for Multi-Language Dynamic Taint Analysis



GraalVM


Clang, DragonEgg, Flang, …
THE C PROGRAMMING LANGUAGE
F
…
LLVM IR
Taint Analysis Applications
Taint Extension
Taint Extension
Taint Extension
Taint Extension
Taint Framework
Language Runtime
Language Runtime
Language Runtime
Language Runtime
…
Truffle
Instrumentation API
Java
Graal
Oracle GraalVM™

Truffle Instrumentation

```
function requestHandler(request, response) {

    var code = Buffer.from(dangerousData());

    var lTag = Buffer.from("<html>");
    var rTag = Buffer.from("</html>");

    var html = Buffer.concat([lTag, code, rTag]);

    response.end(html);
}

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

Truffle Instrumentation

```
function requestHandler(
    request, response) {
  response.end(
    Buffer.concat([
      Buffer.from("<script>"),
      Buffer.from(dangerousData()),
      Buffer.from("</script>")
    ])
  );
}
```

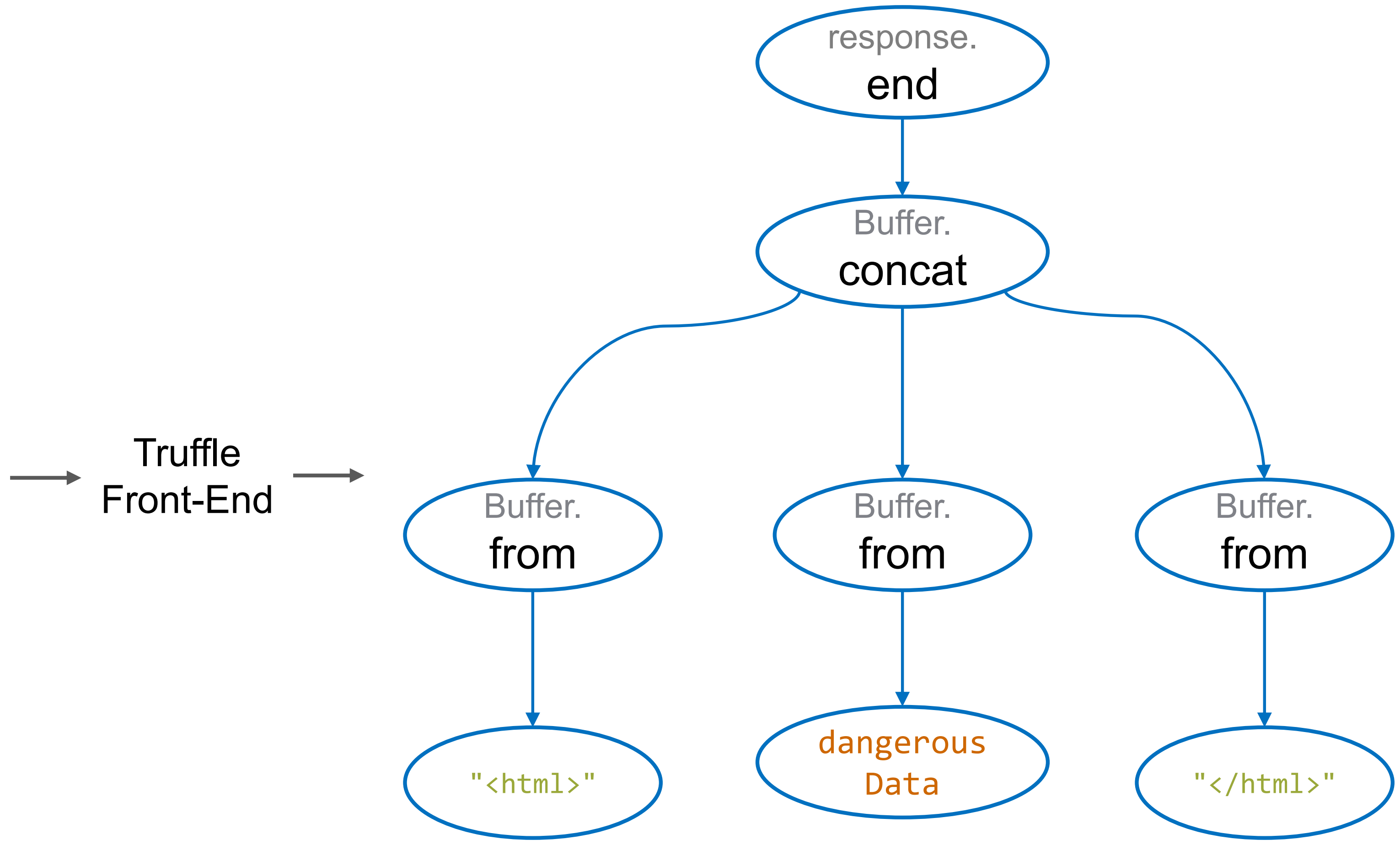
Truffle Instrumentation

```
function requestHandler(
    request, response) {
  response.end(
    Buffer.concat([
      Buffer.from("<script>"),
      Buffer.from(dangerousData()),
      Buffer.from("</script>")
    ])
  );
}
```

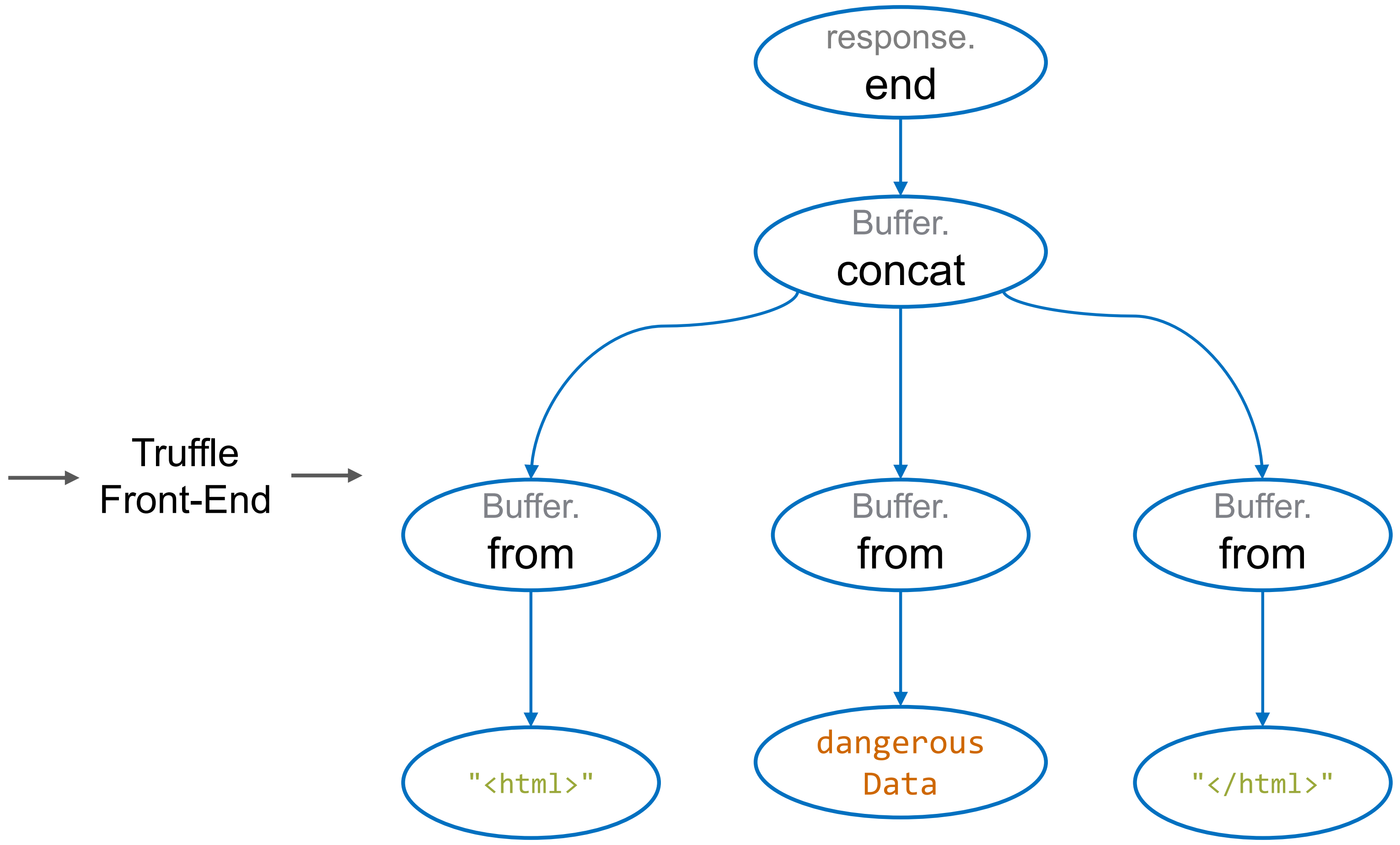
$\longrightarrow$ Truffle Front-End $\longrightarrow$

Truffle Instrumentation

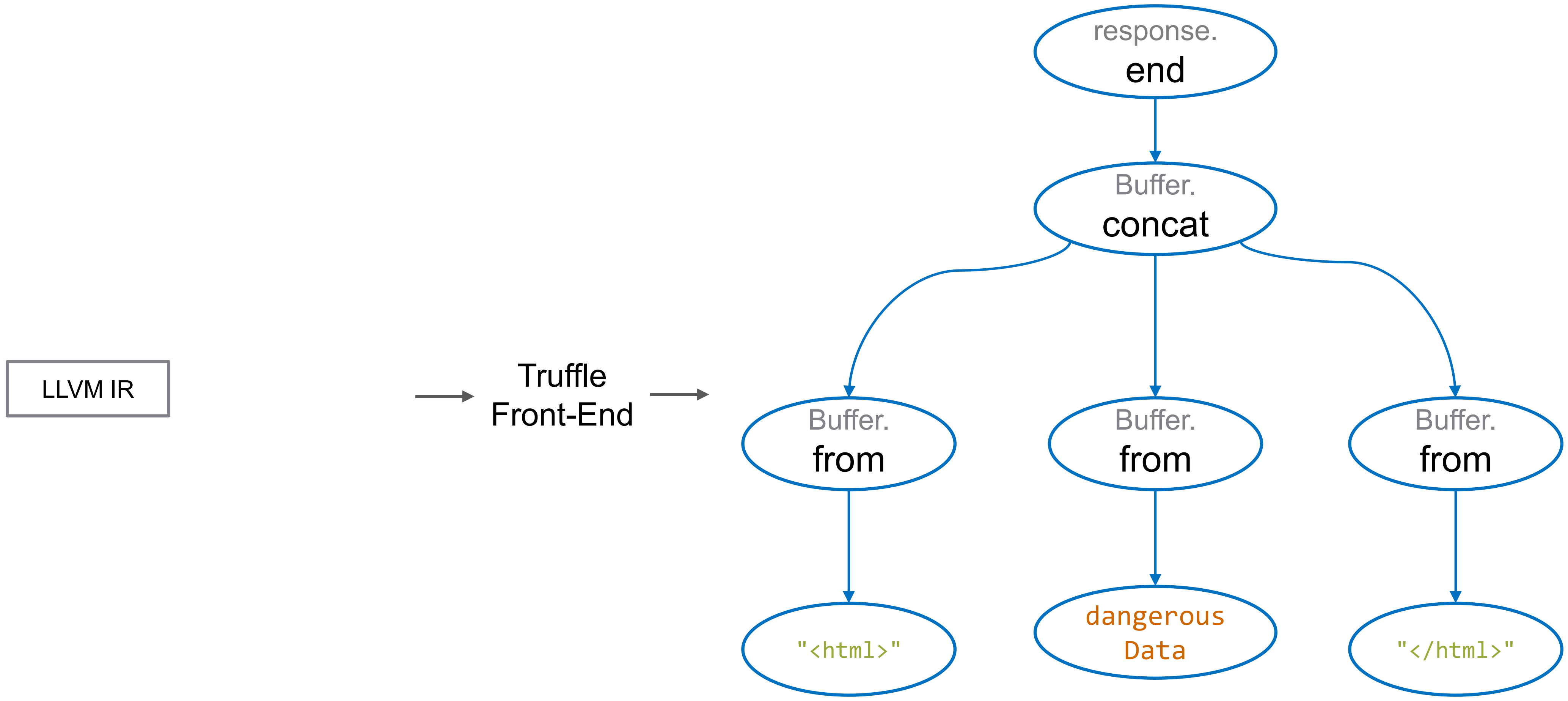
```
function requestHandler(
    request, response) {
  response.end(
    Buffer.concat([
      Buffer.from("<script>"),
      Buffer.from(dangerousData()),
      Buffer.from("</script>")
    ])
  );
}
```



Truffle Instrumentation

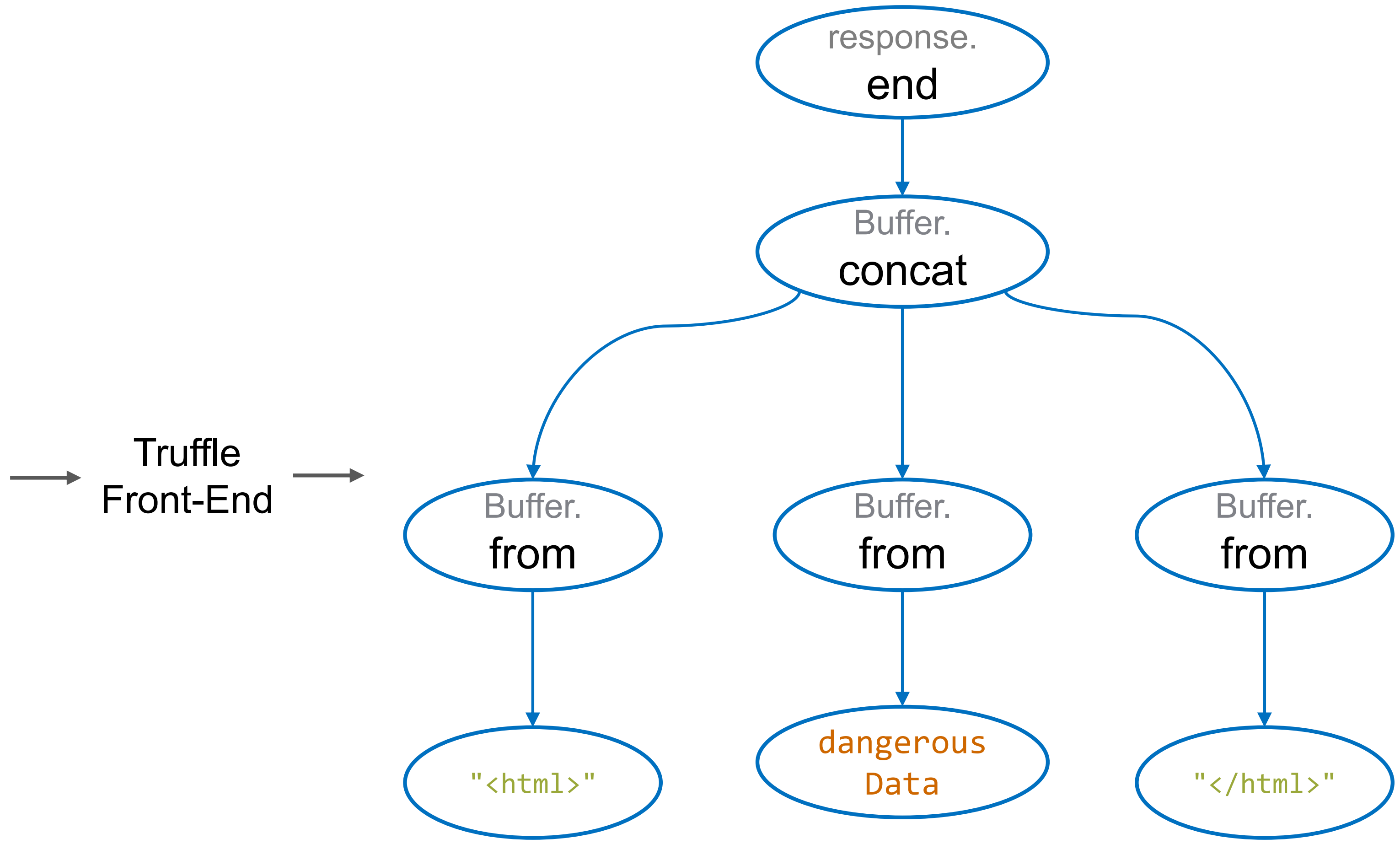


Truffle Instrumentation



Truffle Instrumentation

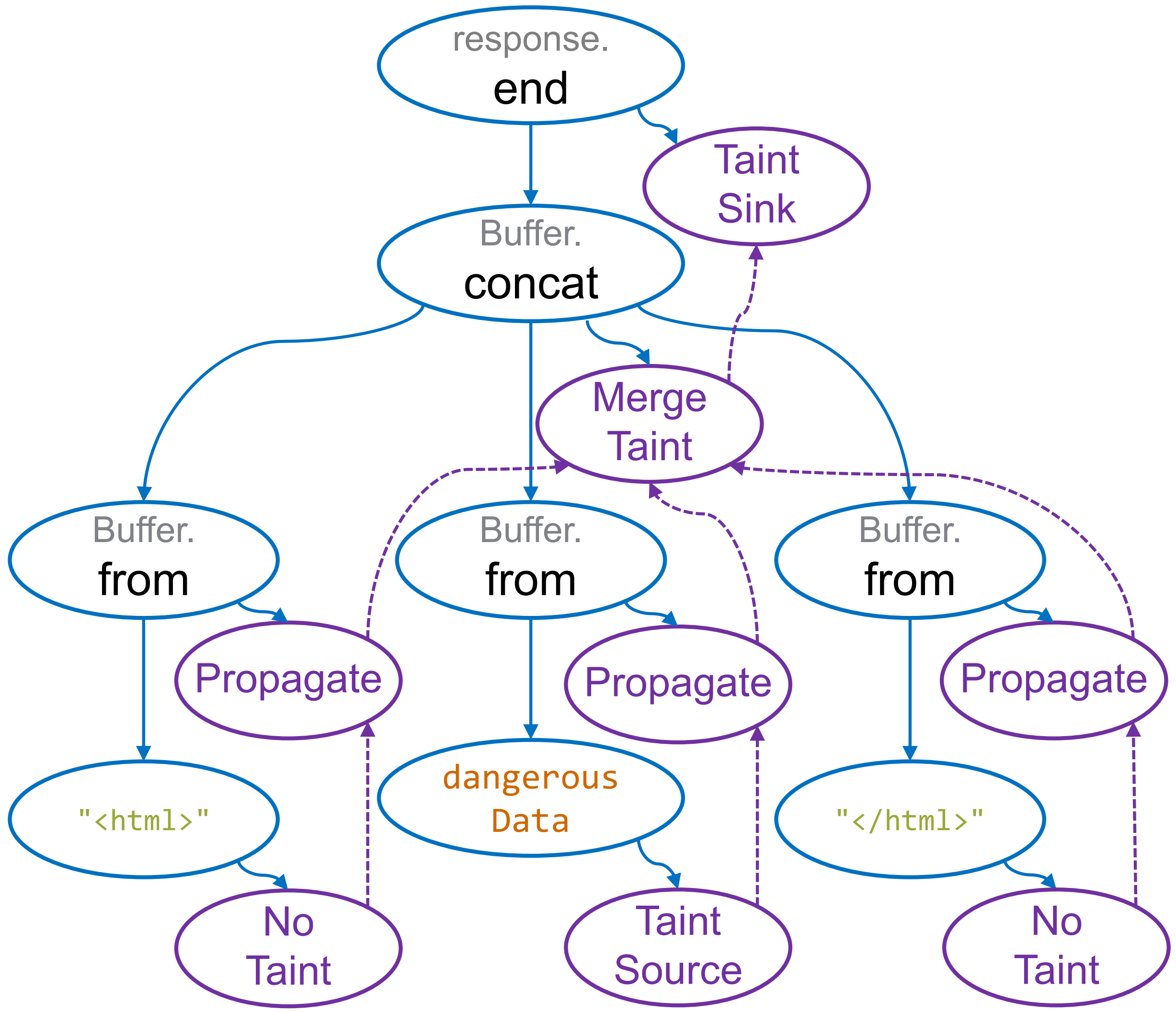
```
function requestHandler(
    request, response) {
  response.end(
    Buffer.concat([
      Buffer.from("<script>"),
      Buffer.from(dangerousData()),
      Buffer.from("</script>")
    ])
  );
}
```



Truffle Instrumentation

```
function requestHandler(
    request, response) {
  response.end(
    Buffer.concat([
      Buffer.from("<script>"),
      Buffer.from(dangerousData()),
      Buffer.from("</script>")
    ])
  );
}
```

→ Truffle Front-End →

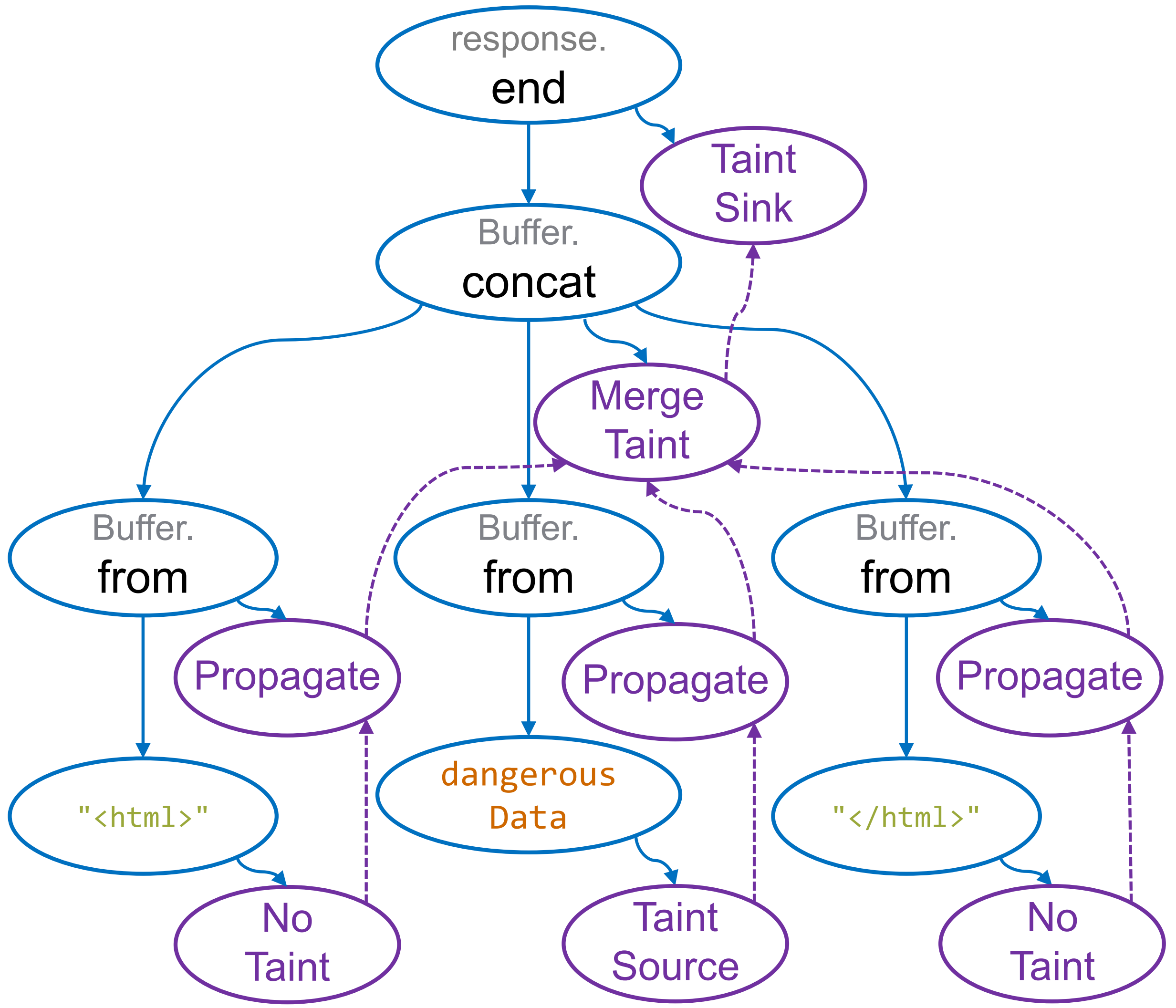


- Insert nodes to propagate taint labels

Truffle Instrumentation

```
function requestHandler(
    request, response) {
  response.end(
    Buffer.concat([
      Buffer.from("<script>"),
      Buffer.from(dangerousData()),
      Buffer.from("</script>")
    ])
  );
}
```

→ Truffle Front-End →

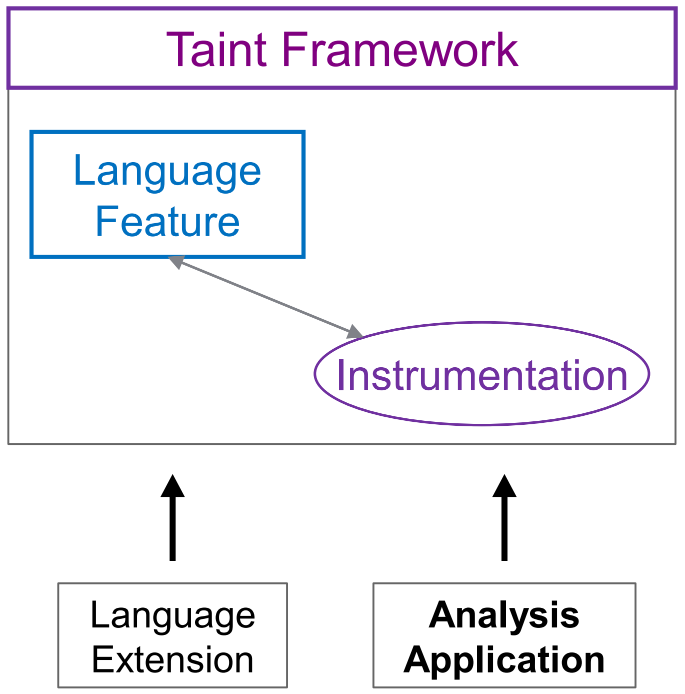


- Insert nodes to propagate taint labels
- Language-level instrumentation
    - Footprint

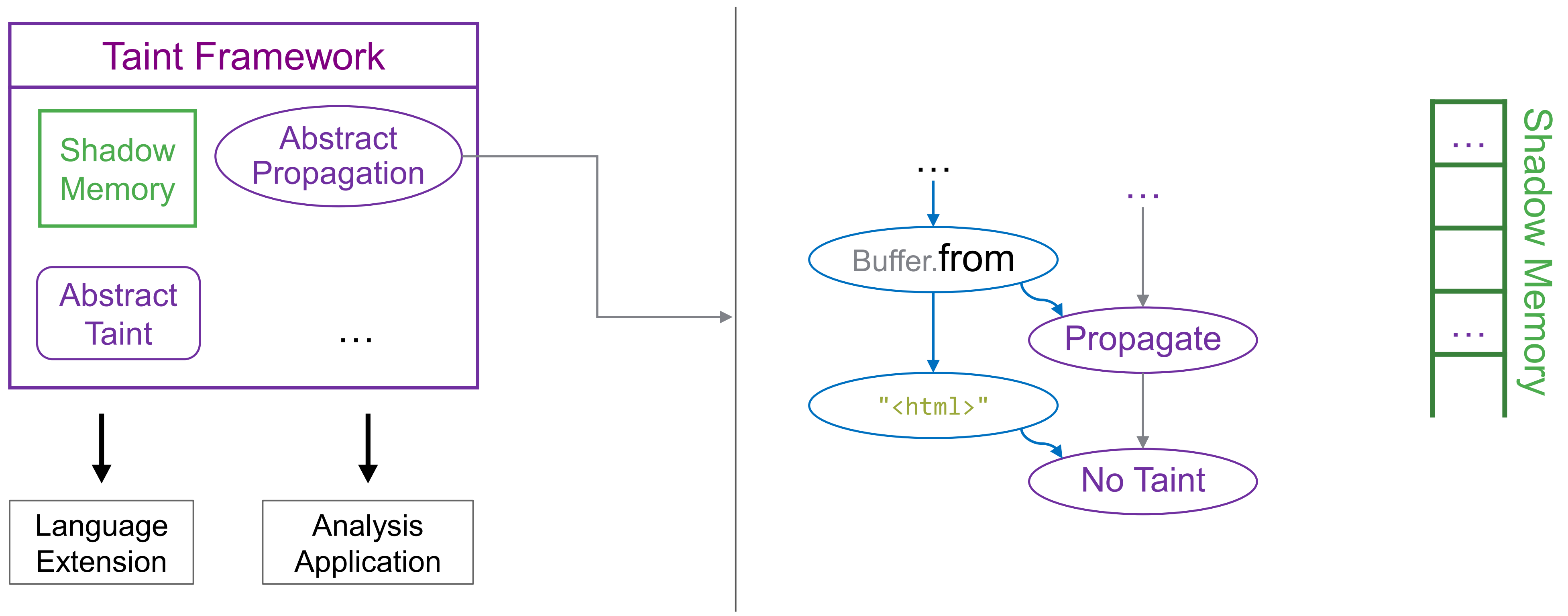
Instrumentation Strategy

- Adaptable instrumentation
  - Language-Level
  - Taint Sources and Taint Sinks
  - Default propgation strategy
  - Full control to analysis applications
  - Code reuse

- Language-agnostic taint analysis
  - Abstractions for common language features

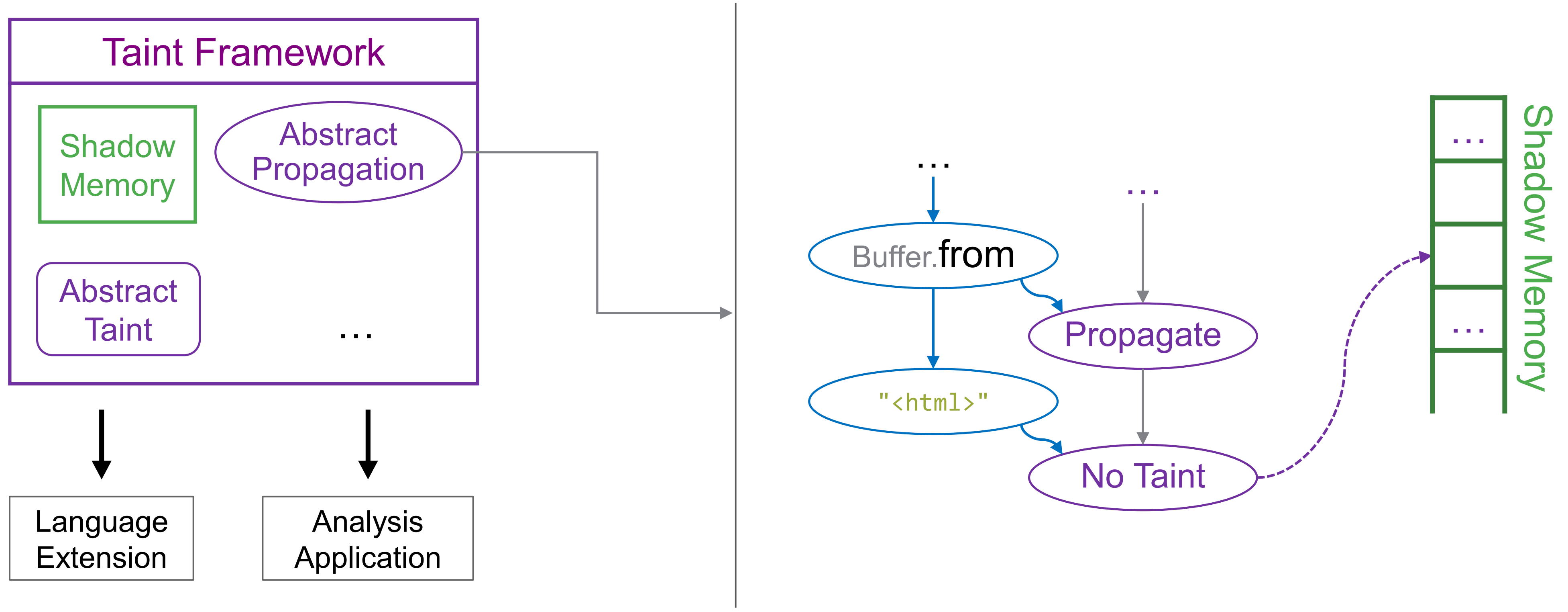


Reusable Functionality



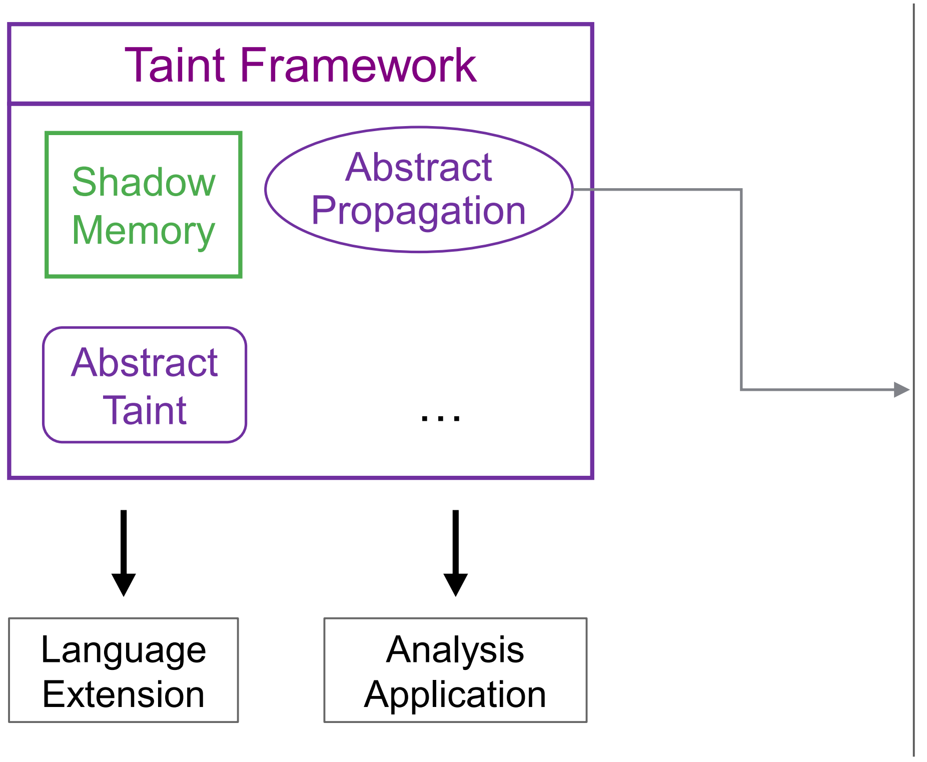
- Support multiple languages and applications
- Efficient implementation

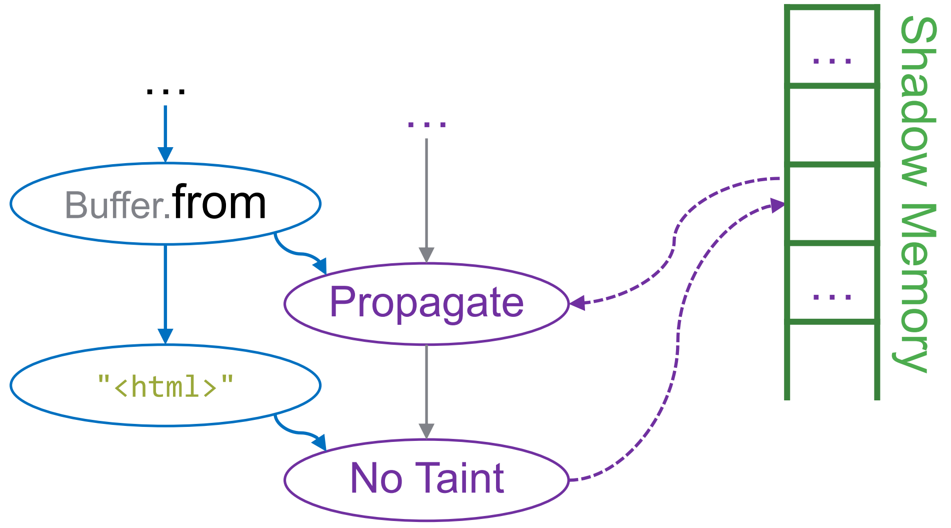
Reusable Functionality



- Support multiple languages and applications
- Efficient implementation

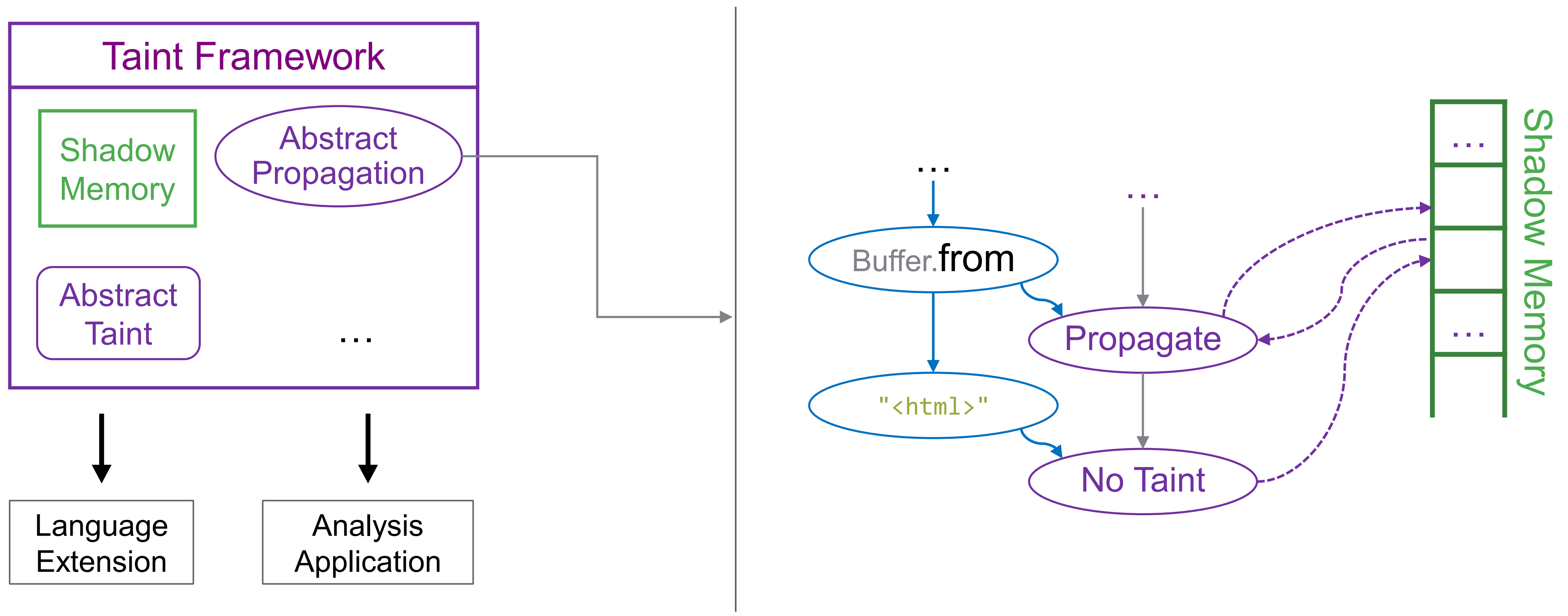
Reusable Functionality





- Support multiple languages and applications
- Efficient implementation

Reusable Functionality

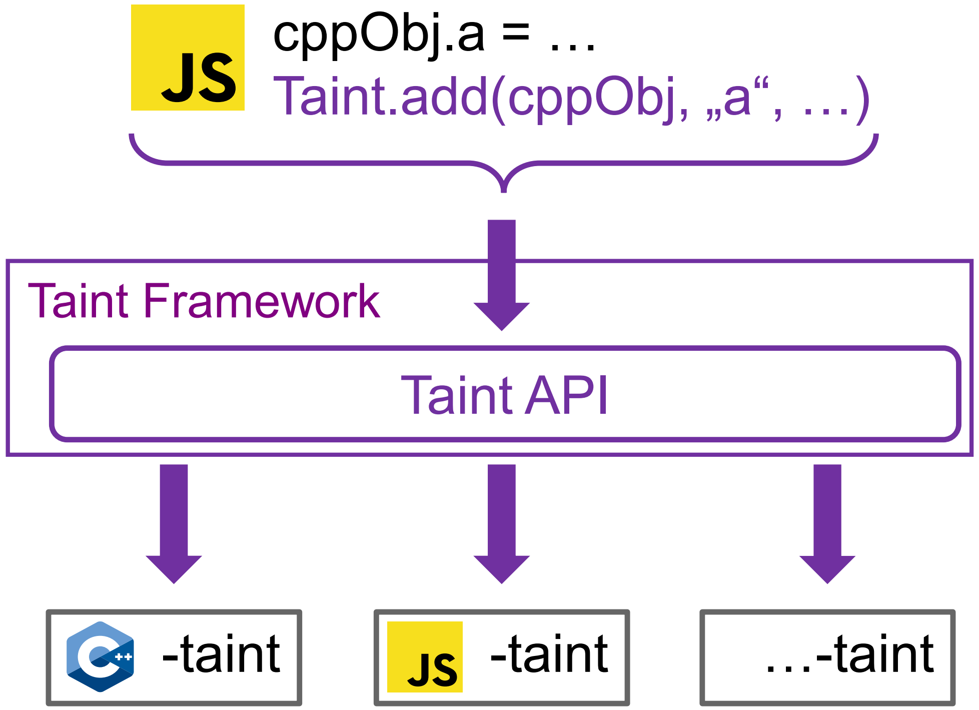


- Support multiple languages and applications
- Efficient implementation

Language Integration

- Truffle Interoperability

- Language-specific storage of member taint
  - Distinct objects vs. memory pointers
  - Granularity

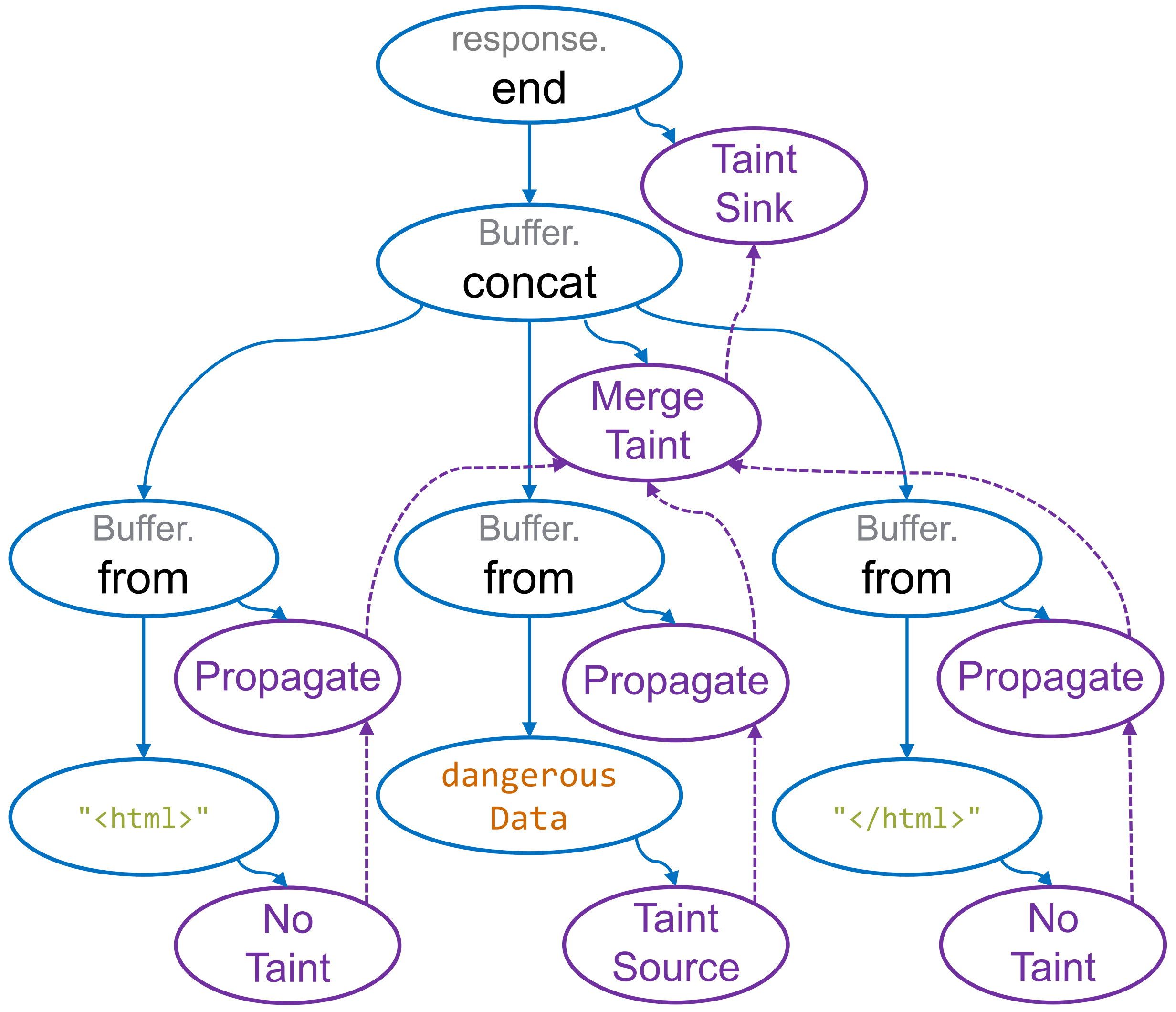


Performance

- Can we just leave it to the compiler?
  - Inlining
  - Partial Evaluation
  - Escape Analysis
  - …

- Optimize code and instrumentation together

- Implement common functionality amenable to Graal compilation

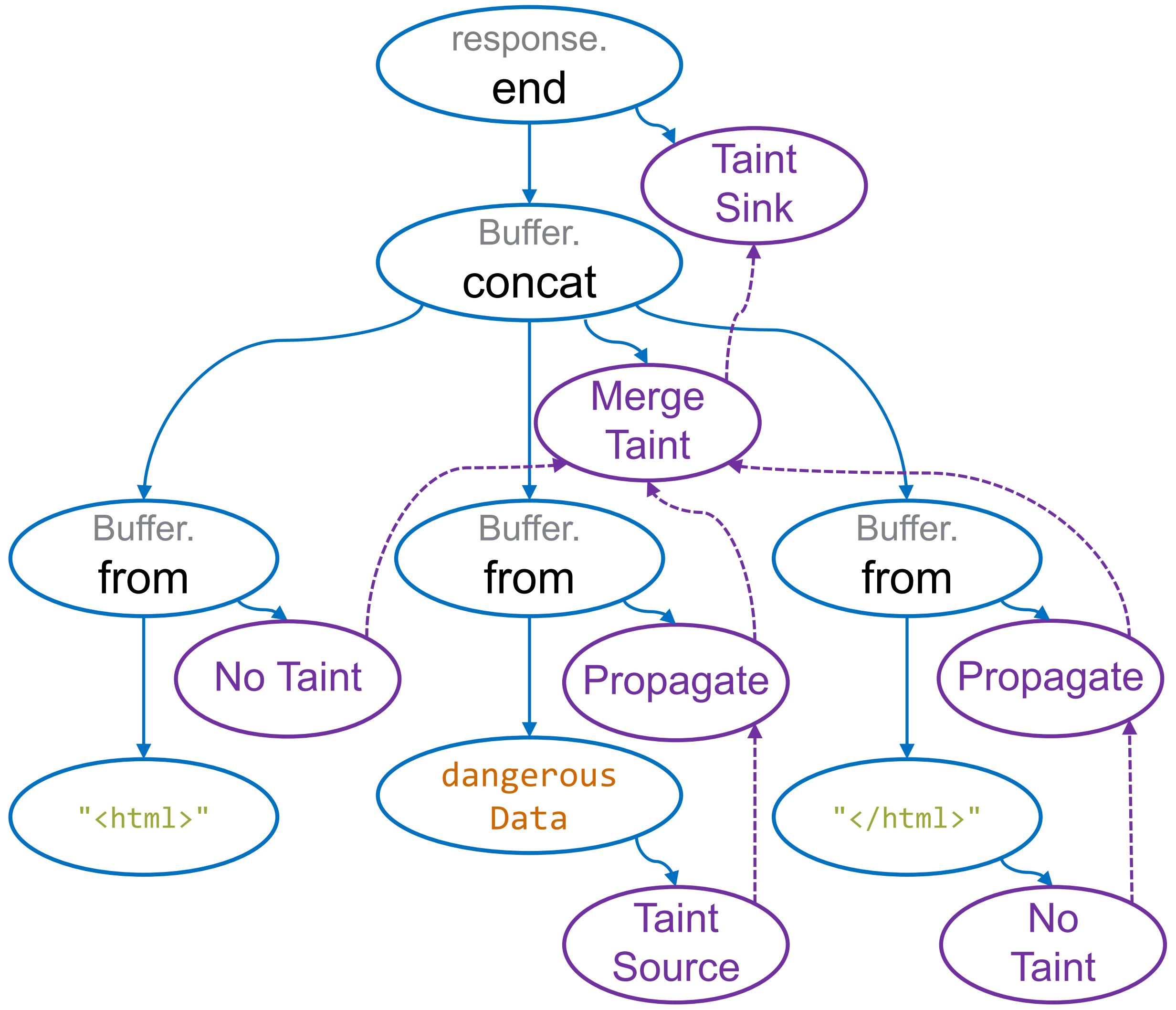


Performance

- Can we just leave it to the compiler?
  - Inlining
  - Partial Evaluation
  - Escape Analysis
  - …

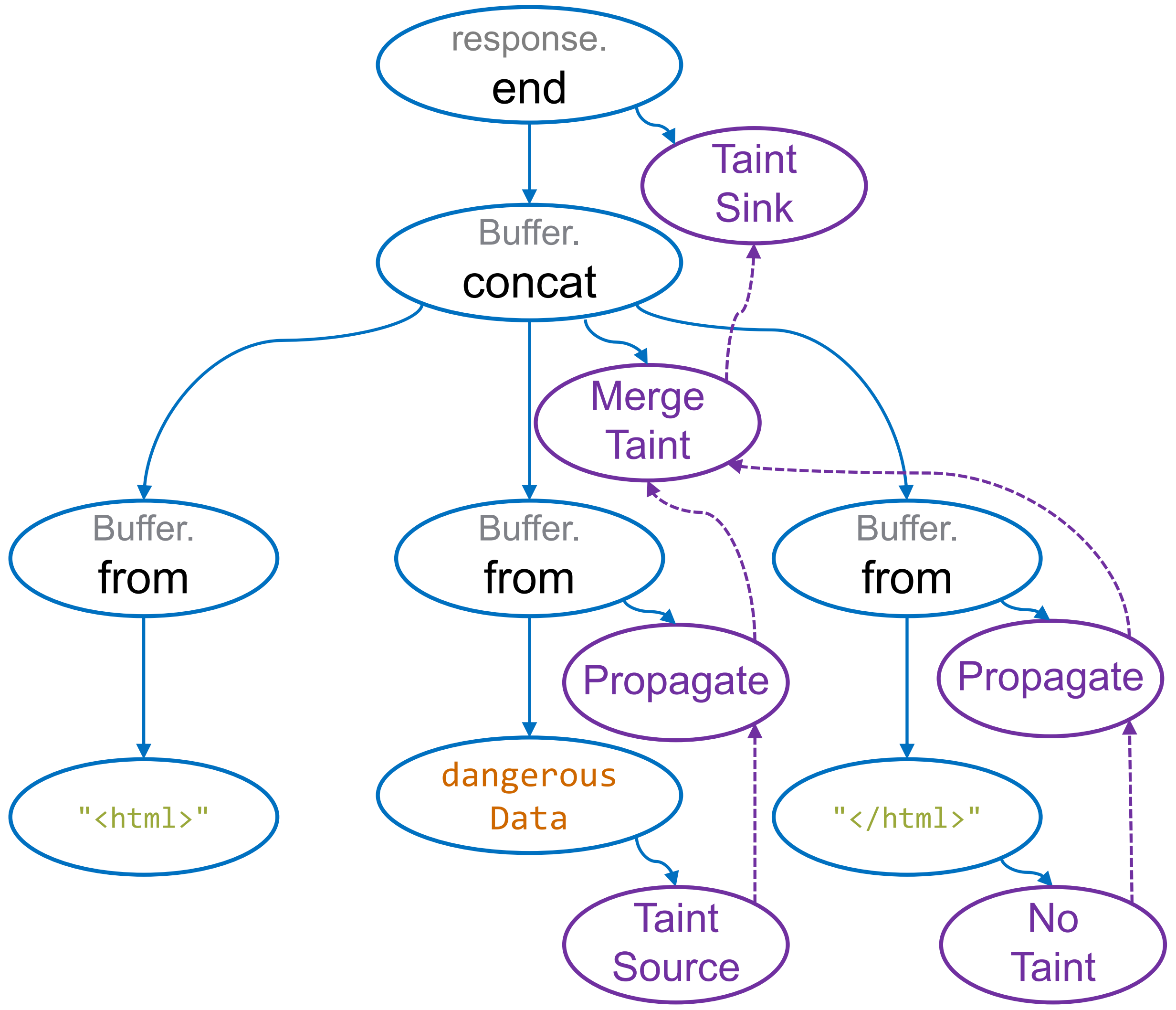
- Optimize code and instrumentation together

- Implement common functionality amenable to Graal compilation



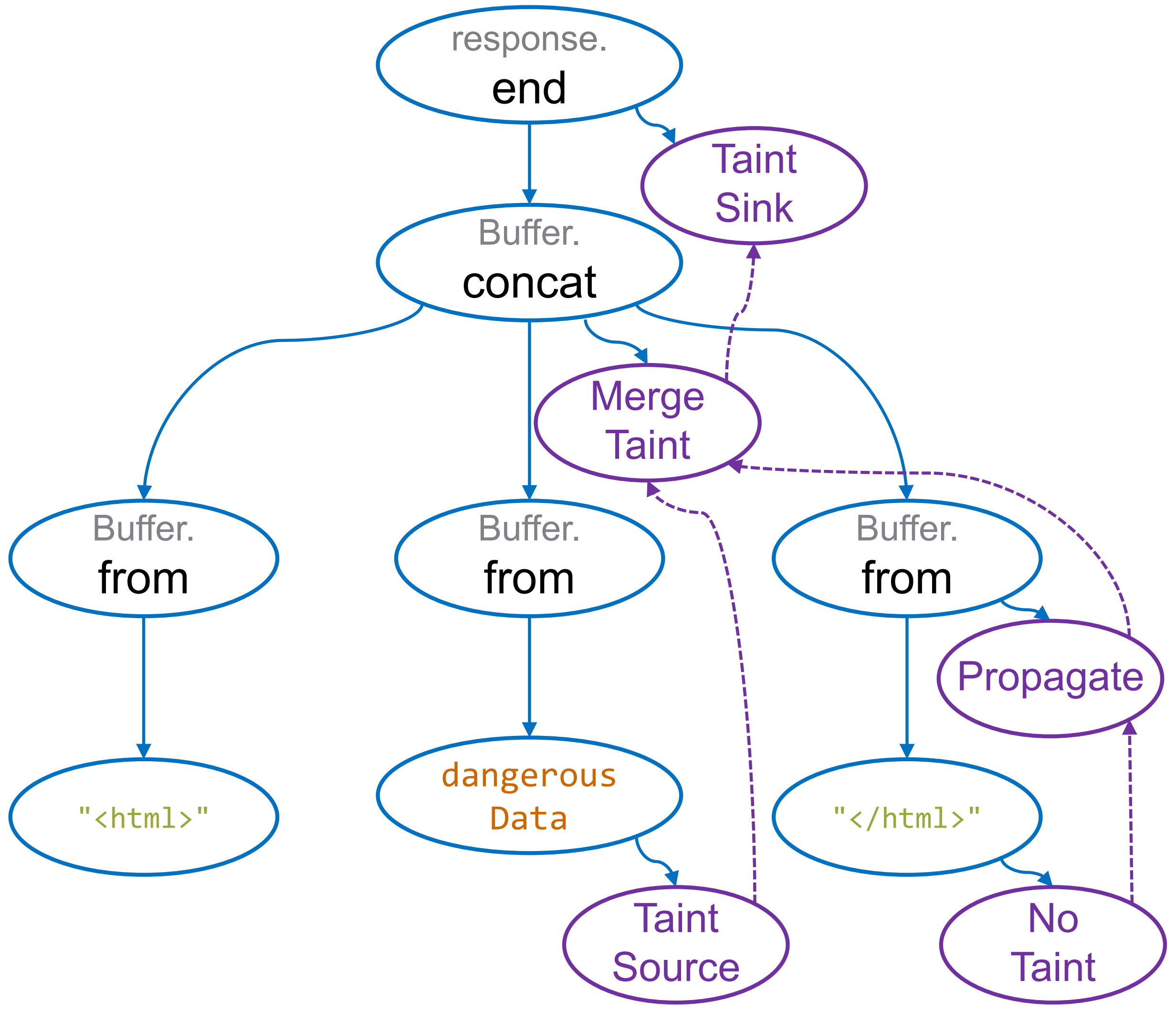
Performance

- Can we just leave it to the compiler?
  - Inlining
  - Partial Evaluation
  - Escape Analysis
  - …
- Optimize code and instrumentation together
- Implement common functionality amenable to Graal compilation



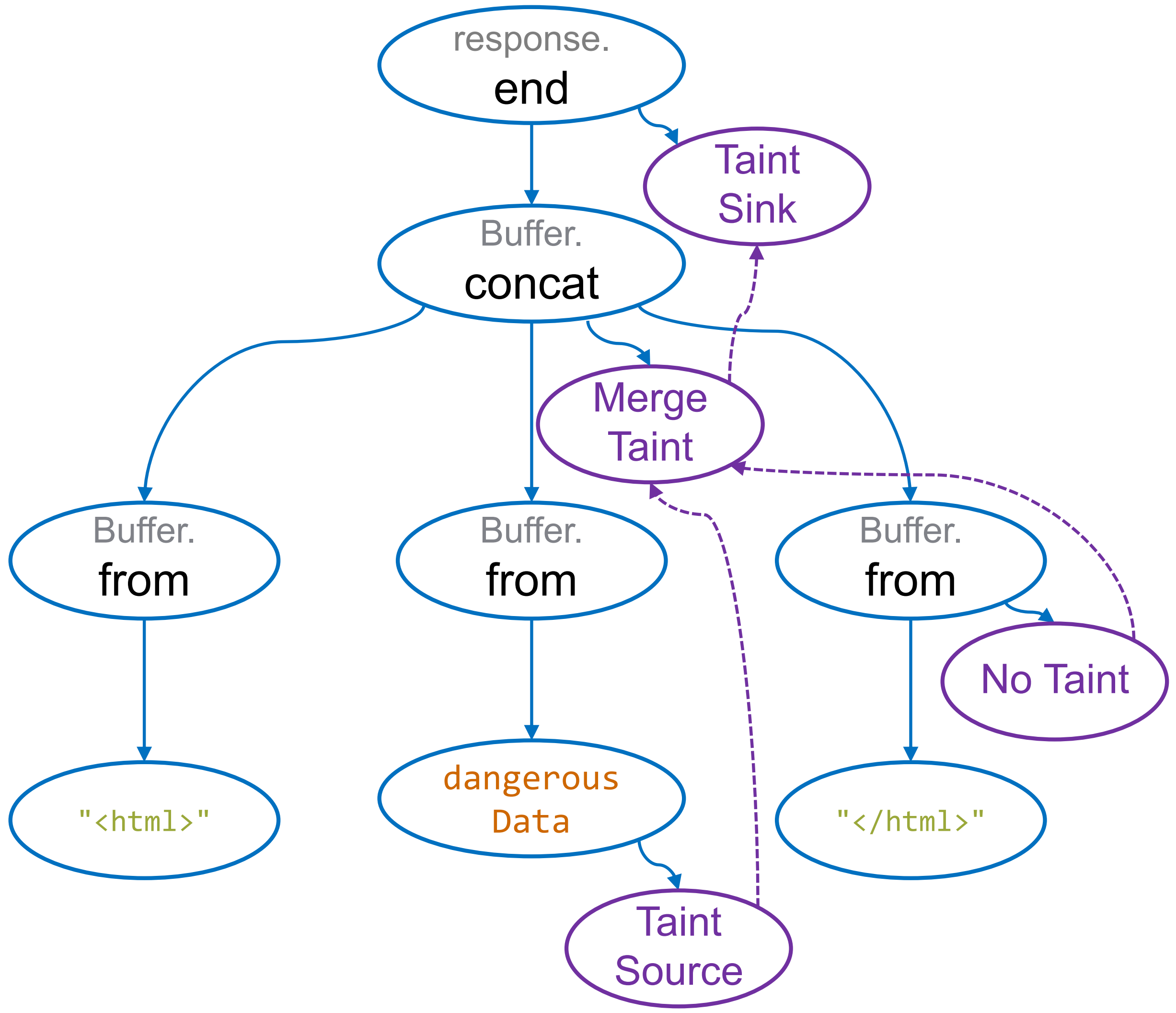
Performance

- Can we just leave it to the compiler?
  - Inlining
  - Partial Evaluation
  - Escape Analysis
  - …
- Optimize code and instrumentation together
- Implement common functionality amenable to Graal compilation



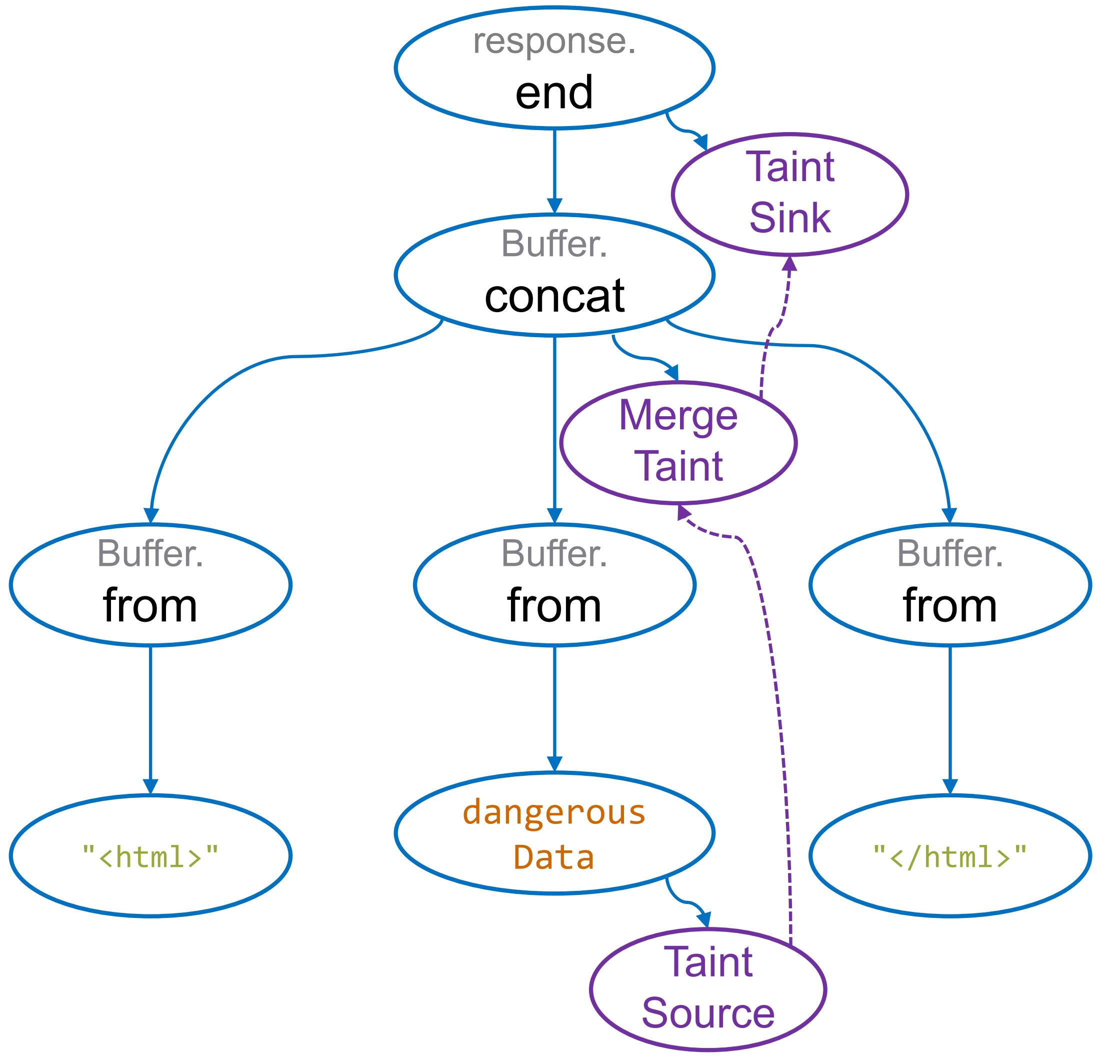
Performance

- Can we just leave it to the compiler?
  - Inlining
  - Partial Evaluation
  - Escape Analysis
  - …
- Optimize code and instrumentation together
- Implement common functionality amenable to Graal compilation



Performance

- Can we just leave it to the compiler?
  - Inlining
  - Partial Evaluation
  - Escape Analysis
  - …
- Optimize code and instrumentation together
- Implement common functionality amenable to Graal compilation

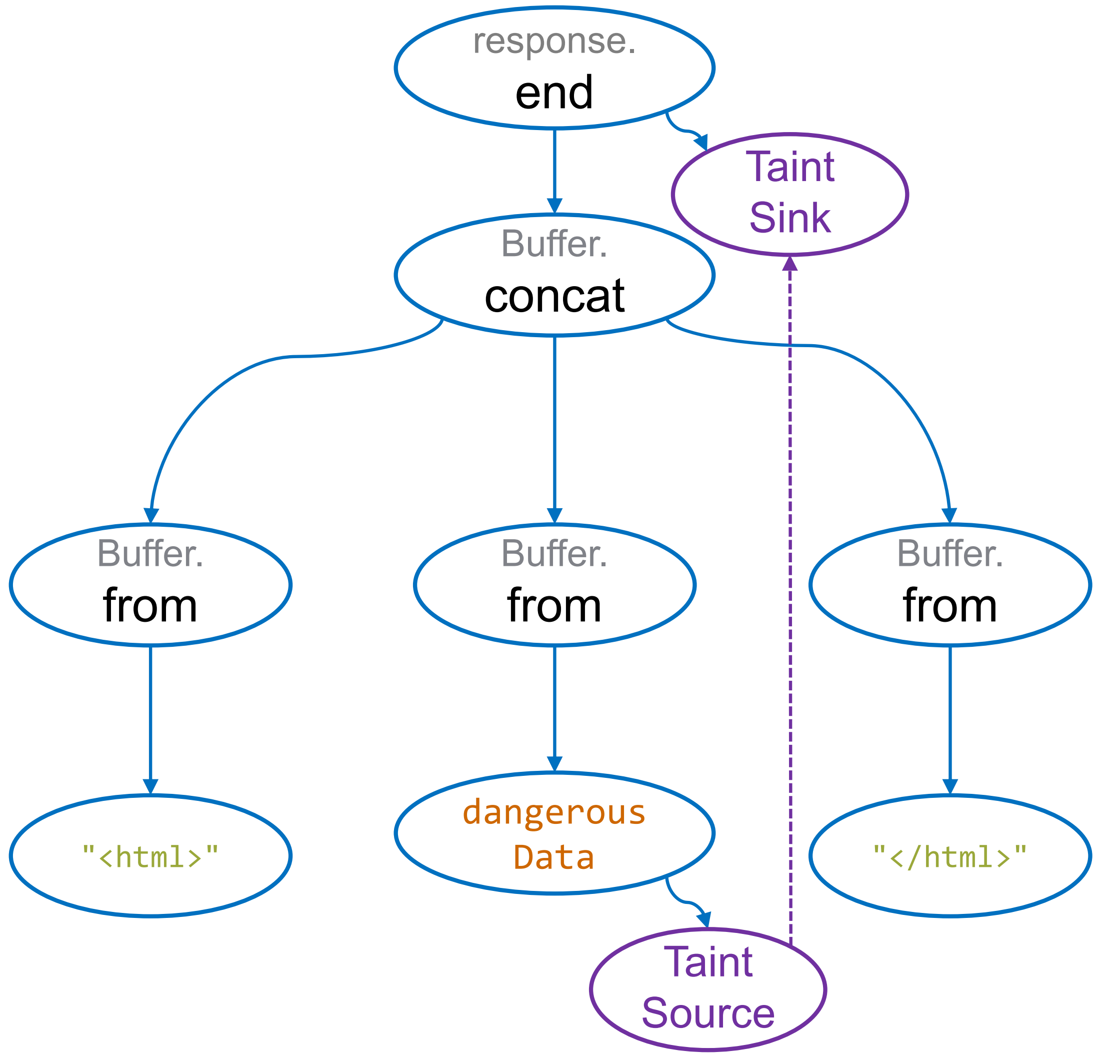


Performance

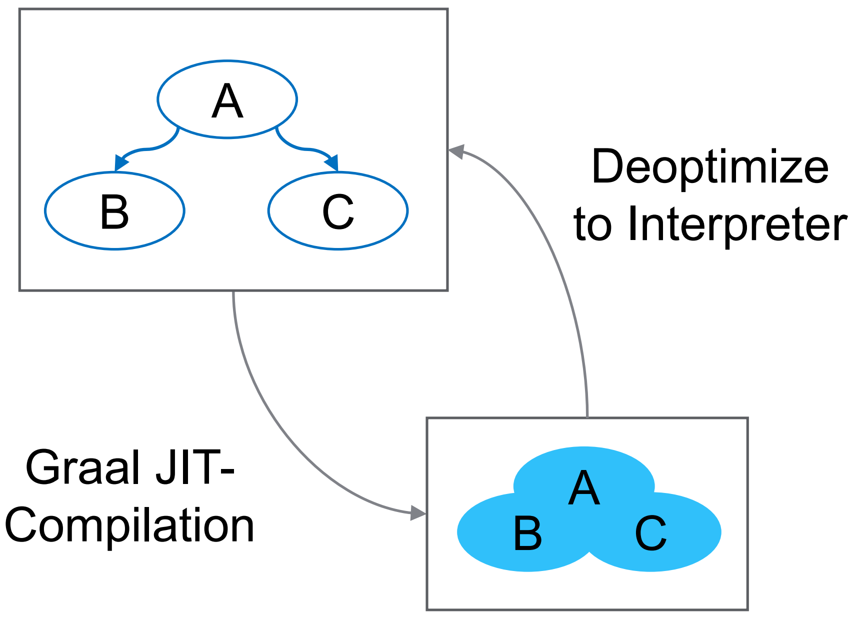
- Can we just leave it to the compiler?
  - Inlining
  - Partial Evaluation
  - Escape Analysis
  - …

- Optimize code and instrumentation together

- Implement common functionality amenable to Graal compilation



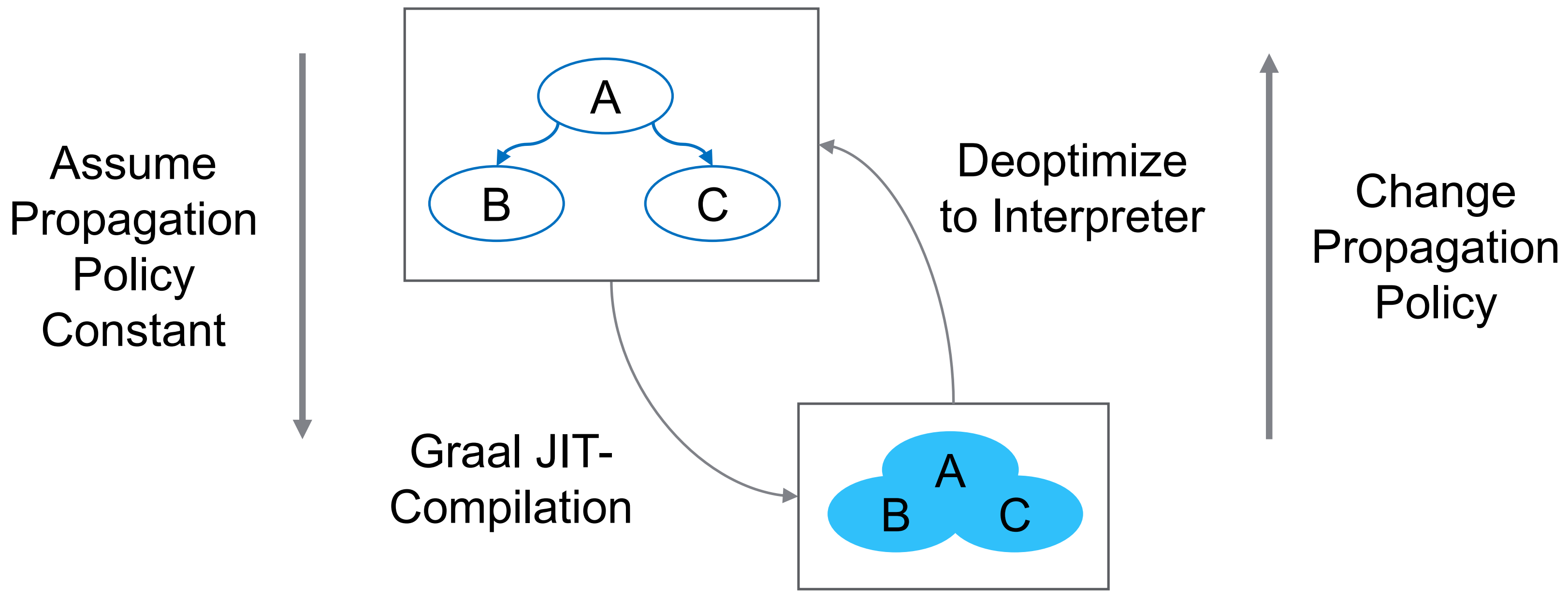
Speculative Optimization and Dynamic Compilation



- Dynamically change instrumentation
  - Assume instrumentation constant for JIT
  - Deoptimize on changed instrumentation

- Runtime-configurable propagation policies
- Assume value properties
- And more to come

Speculative Optimization and Dynamic Compilation


Assume Propagation Policy Constant
A
B
C
Deoptimize to Interpreter
Change Propagation Policy
Graal JIT-Compilation
A
B
C

- Dynamically change instrumentation
  - Assume instrumentation constant for JIT
  - Deoptimize on changed instrumentation

- Runtime-configurable propagation policies
- Assume value properties
- And more to come

Summary

- Programs often use **code of multiple languages**
  - But current taint tracking systems are limited to only one

- In progress: **multi-language taint analysis platform**
  - Choosing suitable abstractions
  - Integrating multiple languages
  - Language-agnostic taint analysis

- In progress: **reduce overhead** by dynamic compilation and speculative optimization
  - Efficient strategies for storing and propagating taint
  - Make use of existing compiler