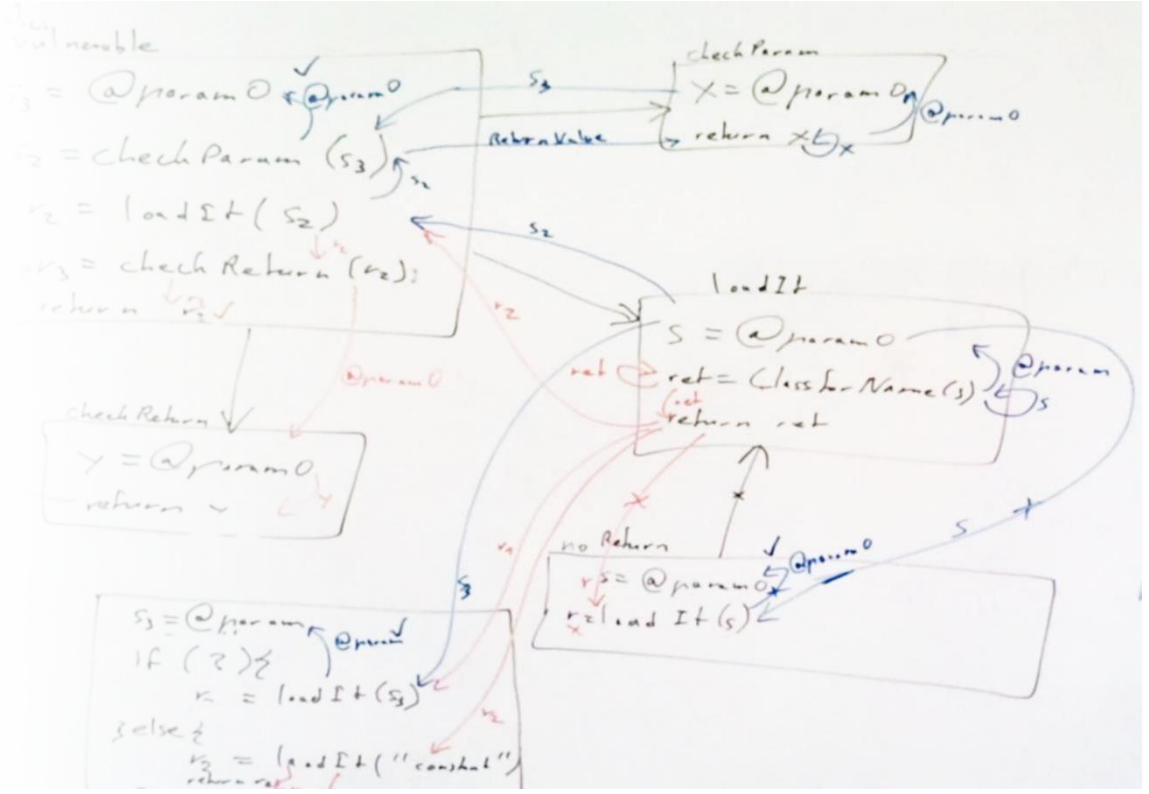
Design Your Analysis!
A Case Study on Implementation Reusability of Data-Flow Functions

Johannes Lerch and Ben Hermann

{lastname}@cs.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT



@stg_darmstadt

SOFTWARE TECHNOLOGY GROUP

Concerns of Analyses

Native Code

Cast Expressions

Exceptions

(Un-)boxing of Types

Return Values

Assignments

Reflection

Method Invocation

Static Fields

Type Conversion

Library Code

Sanitization

Instance Fields

Method Arguments

Taint Sinks

Collections

Taint Sources

Callbacks

Arrays

Analysis Frameworks

Examples: Soot, WALA, OPAL

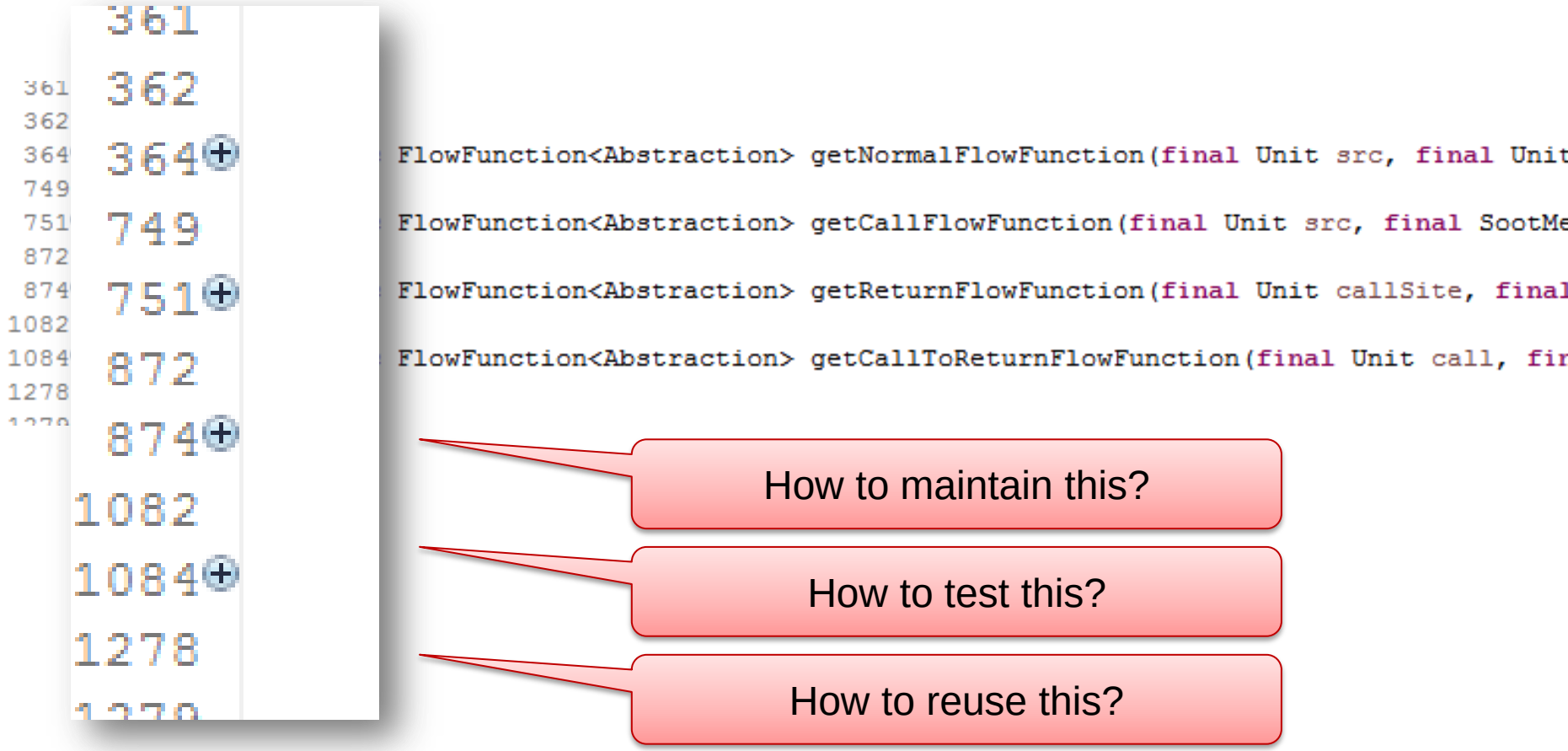
- Intermediate Representations
- Abstractions over Instructions

- Algorithms/Frameworks
  - IFDS/IDE
  - Abstract Interpretation

IFDS/IDE Framework

```
public interface FlowFunctions<N, D, M> {
   FlowFunction<D> getNormalFlowFunction(N curr, N succ);

   FlowFunction<D> getCallFlowFunction(N callStmt,
       M destinationMethod);

   FlowFunction<D> getReturnFlowFunction(N callSite,
       M calleeMethod, N exitStmt, N returnSite);

   FlowFunction<D> getCallToReturnFlowFunction(N callSite,
       N returnSite);
}

public interface FlowFunction<D> {
   Set<D> computeTargets(D source);
}
```

Large Clients of the IFDS Framework

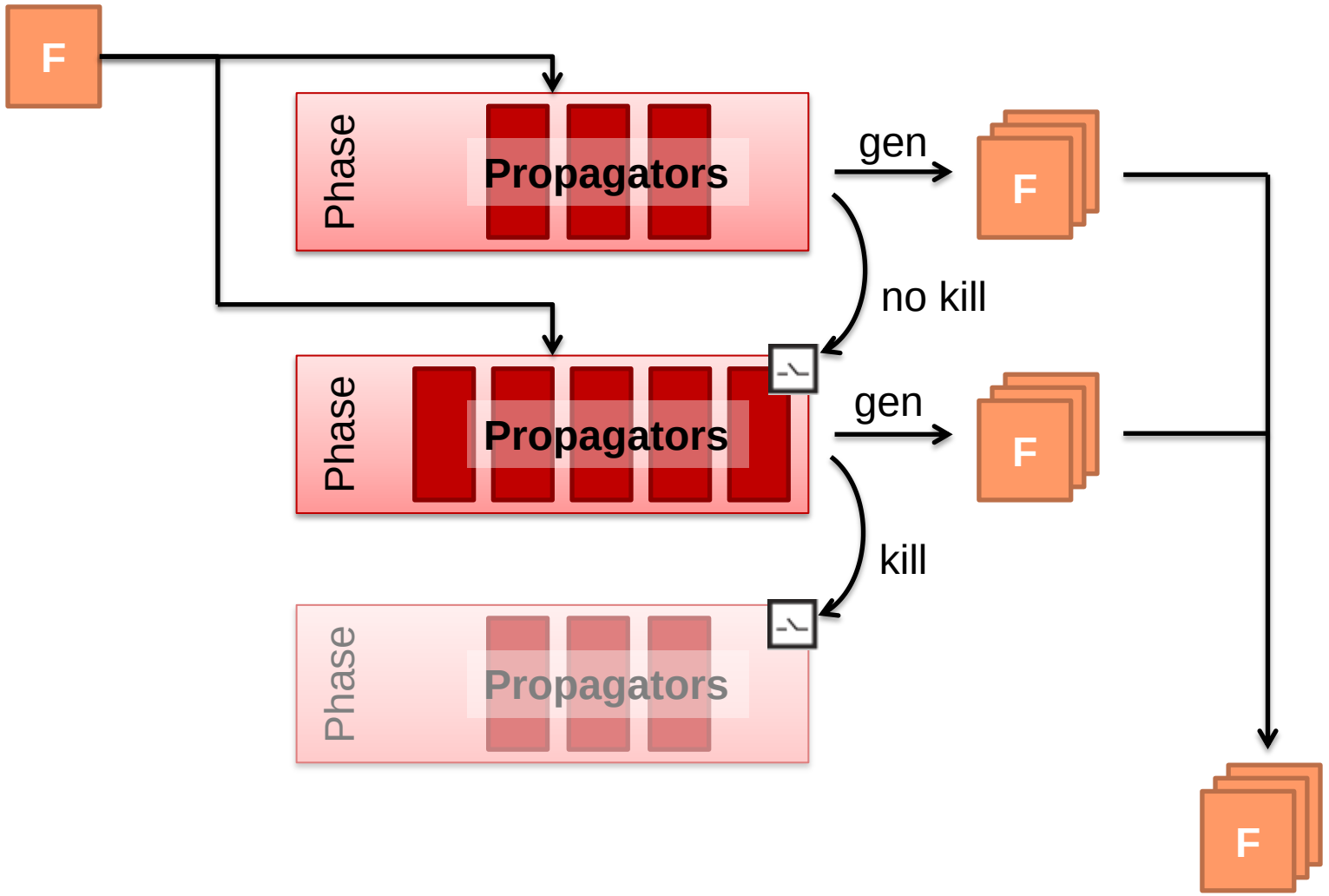


Propagator Interface

```
public interface Propagator<N, D, M> {
  boolean canHandle(D fact);

  KillGenInfo<D> propagateNormalFlow(D source, N curr,
      N succ);

  KillGenInfo<D> pr
      M destinatio

  KillGenInfo<D> pr
      M calleeMeth

  KillGenInfo<D> propagateCallToReturnFlow(D source,
      N callSite);
}
```

```
FlowFunction<D> getNormalFlowFunction(
   N curr, N succ);

public interface FlowFunction<D> {
   Set<D> computeTargets(D source);
}
```

Phase Processing



Implementation

```
Set<D> computeTargets(D source) {
   boolean killed = false;
   Set<D> gens = new HashS
   for(Propagator<D>[] pha
      for(Propagator<D> pr
         if(propagator.car
            KillGenInfo kg
            killed |= kgi.
            gens.addAll(kg
         }
      }
      if(killed)
         break;
   }
   return gens;
}
```

```
phases = new Propagator[][] {
   {
      new PrimitiveTypesKiller(),
      new PermissionCheckPropagator(),
      /* ... */
   },
   {
      new AssignmentPropagator(),
      new FieldAccessPropagator(),
      new StringBuilderPropagator(),
      /* ... */
   },
   {
      new SinkHandler(),
      /* ... */
   }
};
```

Discussion

- Separation of concerns
  - Easier to **maintain**
  - Easier to **test**
  - Easier to **reuse**
- Case Study
  - Implemented SQL-Injection, Path Traversal, Unchecked Redirect, … vulnerability detection
  - **Reused** FlowTwists implementations, only source, sink, and sanitization specific Propagators implemented