FlowDist: Multi-Staged Refinement-Based Dynamic Information Flow Analysis for Distributed Software Systems

Xiaoqin Fu and Haipeng Cai

School of Electrical Engineering and Computer Science

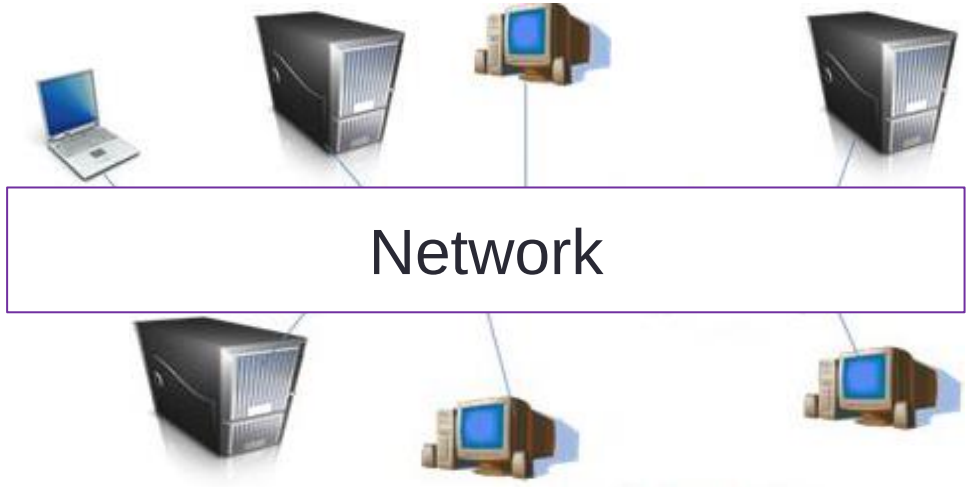
Washington State University

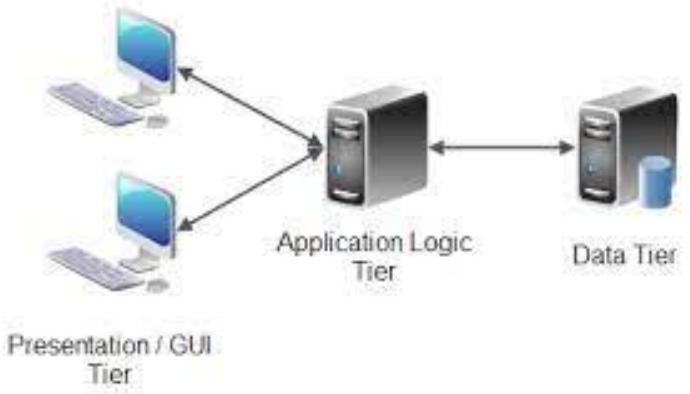
30TH USENIX Security Symposium

Contributions

- The first purely application-level dynamic information flow analysis for common distributed systems, FLOWDIST.
- Alternative designs of FLOWDIST exploring the design methodology.
- An open-source implementation of FLOWDIST for Java working with real-world, large-scale distributed software systems.
- Evaluations of FLOWDIST showing its practical effectiveness, scalability, and capabilities in vulnerability discovery.

Distributed systems





Our Technical Aims

- Scalability
- Effectiveness

An information flow security vulnerability

```
// Executed in a Client process
39 public class ClientCnxnSocketNIO extends ClientCnxnSocket { . . .
61        public void doIO(java.utils.list,···) { . . .
63           SocketChannel sock = (SocketChannel) sockKey.channel();
68           int rc = sock.read(incomingBuffer);  // Source
                  ·····
103          Packet p = findSendablePacket(outgoingQueue,...
                        ·····
107          sock.write(p.bb); . . . }}

// Executed in a Container process
247 public class InstanceContainer implements Watcher,  . . . {
391     public void run() throws IOException,  . . . {
392        zk = new ZooKeeper(zkHostPort, sessTimeout, this);
393        mknod(assignmentsNode, CreateMode.PERSISTENT);
                     ·····
397        zk.getChildren(assignmentsNode, true, this, nul . . . }  . . . }

// Executed in a Server process
432 public class BinaryOutputArchive implements OutputArchive {
437     public getArchive(java.io.OutputStream strm) {
438        return new BinaryOutputArchive(new DataOutputStream(strm)); }
          ·····
442     public BinaryOutputArchive(DataOutput out) {
443        this.out = out;
            ·····
         . . . }
454     public void writeInt(int i, String tag) throws IOException {...
455       out.writeInt(i);  // Sink          . . . }    . . . }
```

Blue line: source information flow path segment (*SOFPS*)
Green line: remote information flow path segment (*REFPS*)
Red line: sink information flow path segment (*SIFPS*)

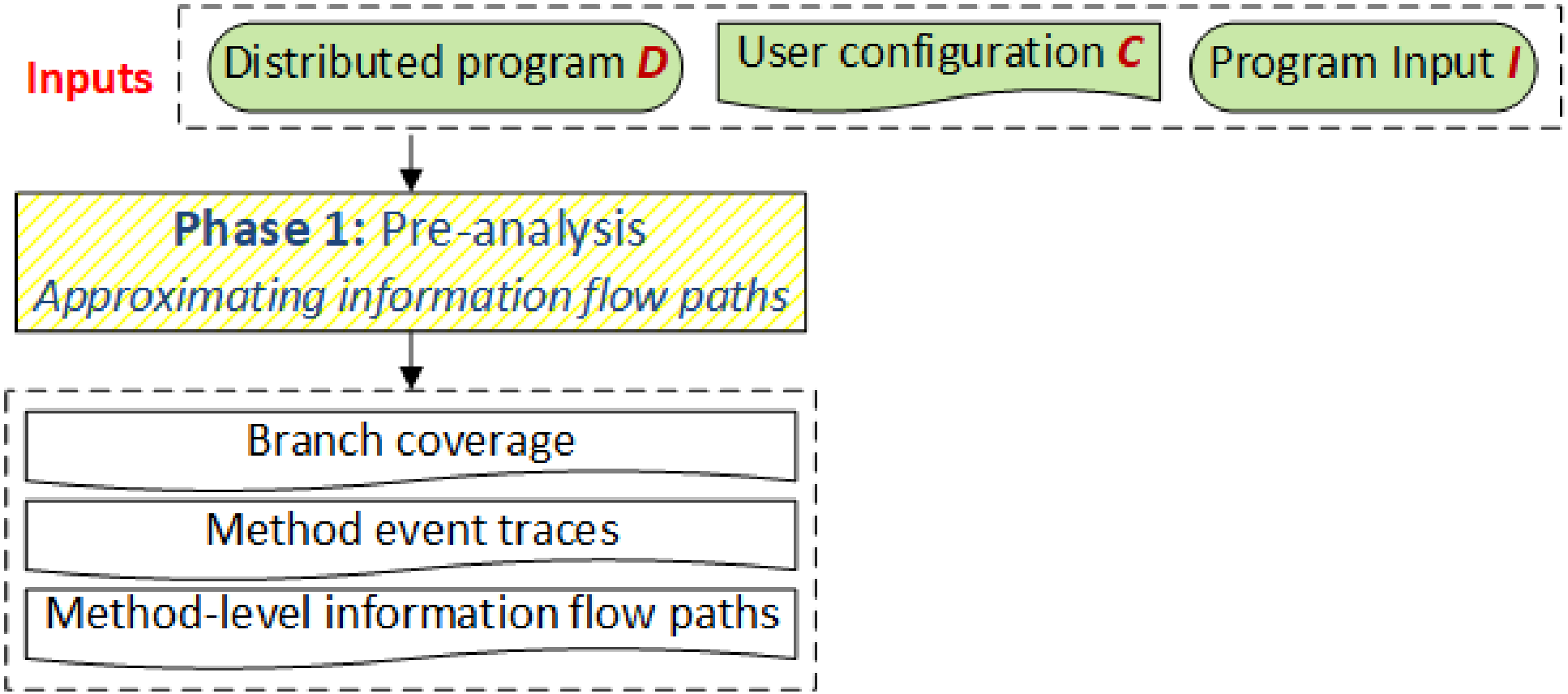
Solid line: intraprocess flow
Dashed line: interprocess flow

Challenges and the solution

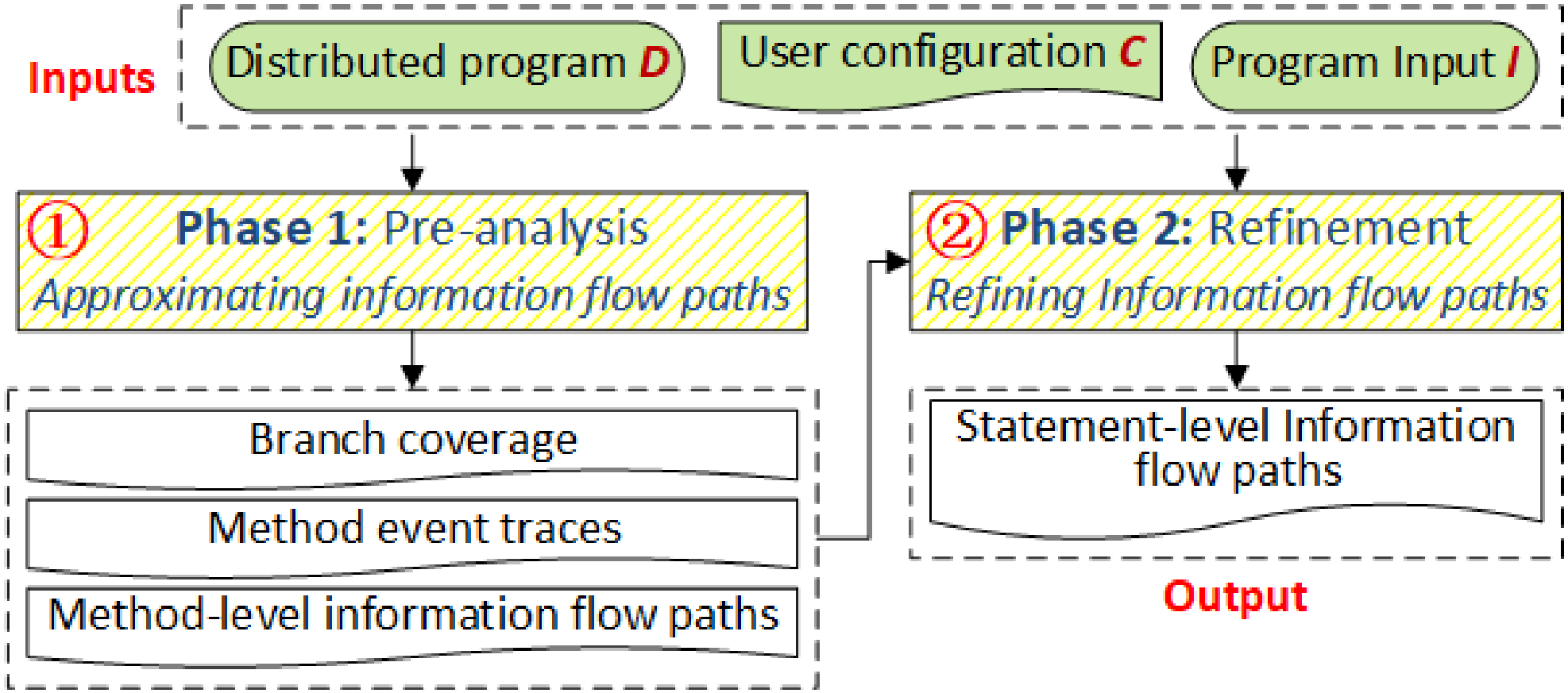
- Applicability challenge: Part-ordering of method executed events
- Portability challenge: Application-level, no platform modification
- Scalability challenge: Multi-phased, refinement-based analysis

Overall Workflow

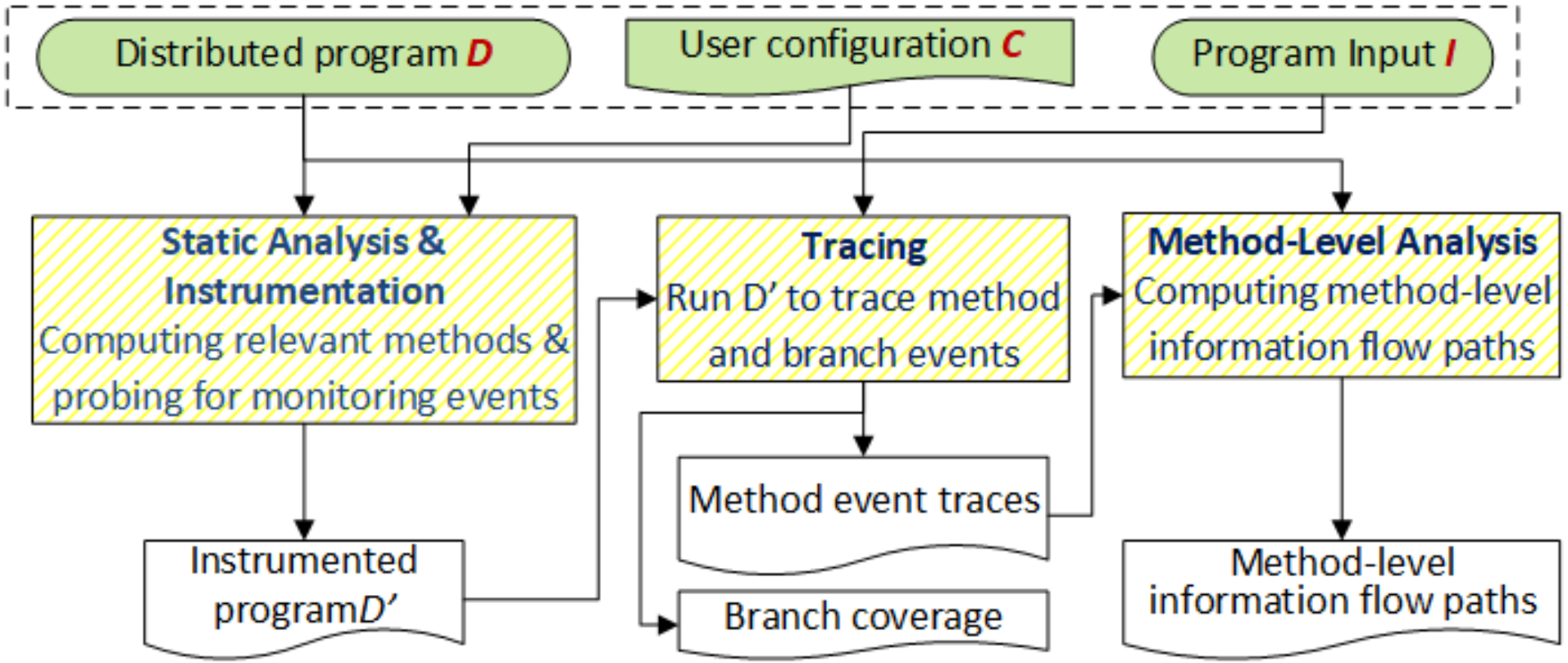
Overall Workflow



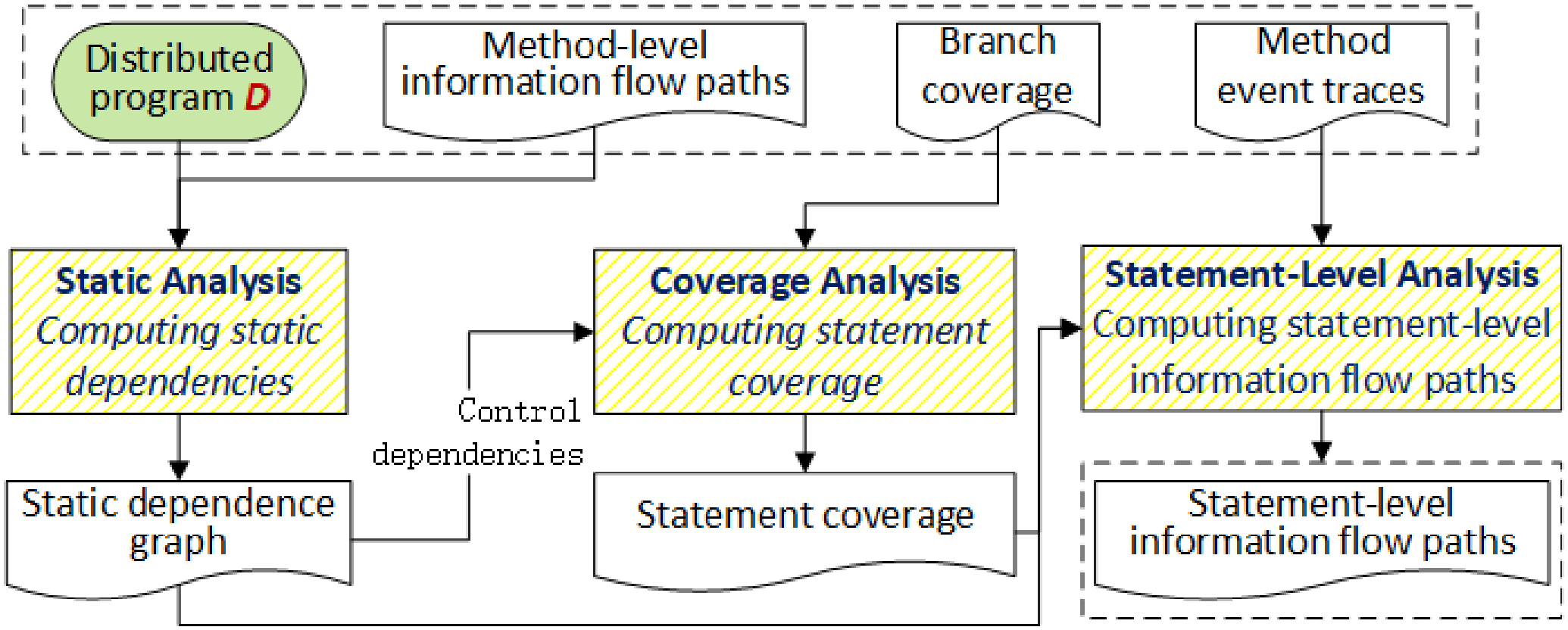
Overall Workflow



Workflow: phase1



Workflow: phase2



Alternative designs:

FLOWDISTsim

- In the first step, FLOWDISTsim skips the static analysis, and simply instruments all methods and branches.

FLOWDISTmul

- The Phase 1 of FLOWDISTmul only probes for and traces the first entry and last returned-into events of each method, and computes method-level flow paths.
- The new intermediate phase then probes for and traces the coverage of branches in, and all instances of both kinds of events of, methods on such paths.
- Lastly, the second step is removed from Phase 2.

Evaluation subjects

Subject	#SLOC	Scenario	Tests
NIOEcho	412	Client/server	Integration
MultiChat	470	Peer To peer	Integration
ADEN	4,385	Peer To peer	Integration
Raining Sockets	6,711	Client/server	Integration
OpenChord	9,244	Peer To peer	Integration
Thrift	14,510	Client/server	Integration
xSocket	15,760	Peer To peer	Integration
ZooKeeper	62,194	Client/server, N-tier, N-tier	Integration, Load, System
RocketMQ	105,444	N-tier, N-tier	Integration, System
Voldemort	115,310	Client/server, N-tier, N-tier	Integration, Load, System
Netty	167,961	N-tier	Integration
Hsqldb	326,678	Client/server, N-tier	Integration, System

Information flow paths found by FLOWDIST

Subject	Test	#Paths
NioEcho	Integration	6
Raining Sockets	Integration	0
Thrift	Integration	3
xSocket	Integration	2
Zookeeper	Load	64
	System	46
RocketMQ	Integration	17
	System	50
Voldemort	Integration	138
	System	42
Netty	Integration	2
HSQLDB System	System	4
Total:		**374**

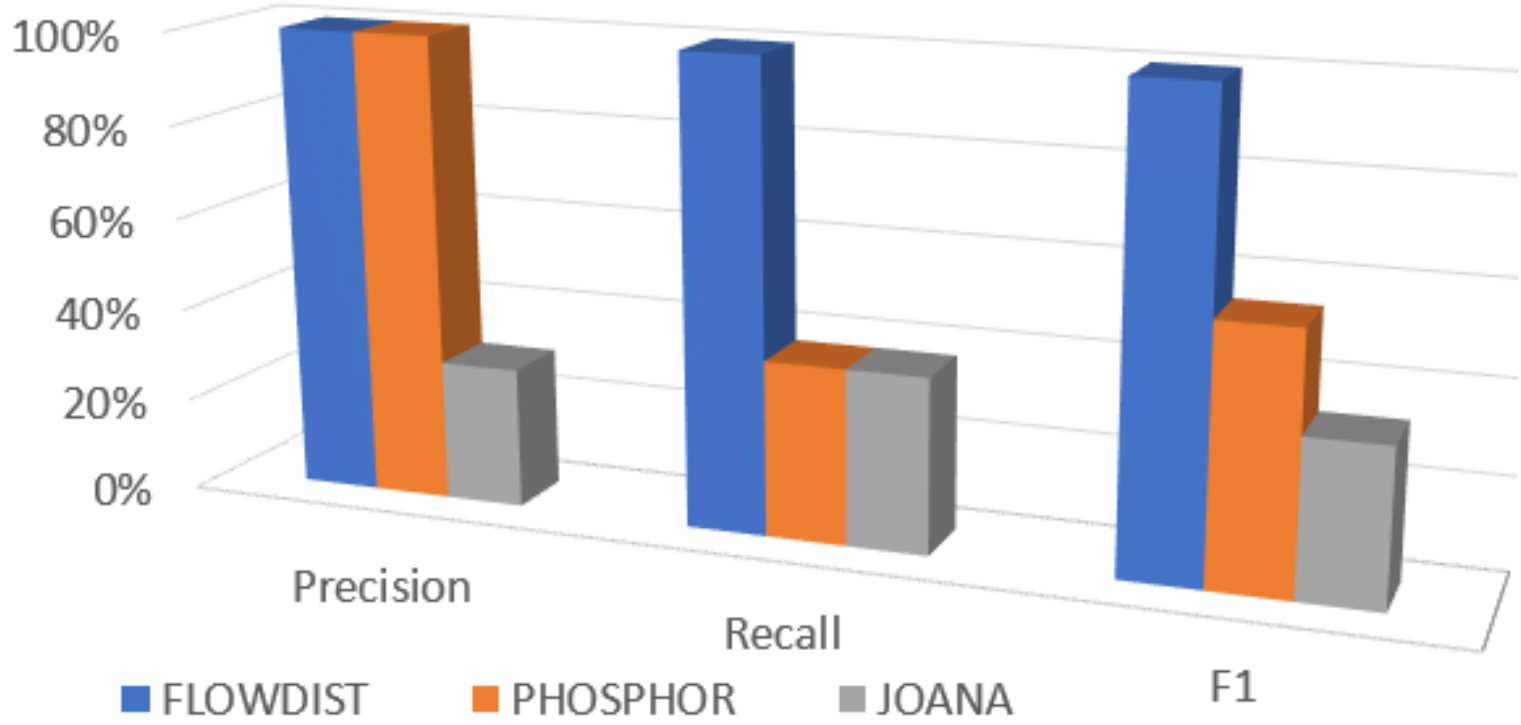
Existing vulnerabilities detected by FLOWDIST

Subject	# Vulnerability	# Detected	# False Positive
HSQLDB	1	1	0
Netty	10	5	5
RocketMQ	1	1	0
Thrift	1	1	0
Voldemort	6	5	1
xSocket	1	1	0
ZooKeeper	4	4	0
Total:	**24**	**18**	**6**

New vulnerabilities detected by FLOWDIST

Subject	#Fixed	#Confirmed	#Pending
HSQLDB	0	5	2
Netty	1	1	0
RainingSockets	0	1	0
RocketMQ	0	4	0
Thrift	0	5	0
Voldemort	0	0	4
xSocket	0	0	1
Zookeeper	1	1	0
Total:	**2**	**17**	**7**

Compared with baselines



- Low recall: JOANA, Phosphor
- Low precision: JOANA
- Low F1: JOANA, Phosphor

Recommendations on the selection among FLOWDIST alternative designs and peer tools

<table>
<tr><th colspan="3" rowspan="2">System type</th><th colspan="2">With non-deterministic executions?</th></tr>
<tr><th>Yes</th><th>No</th></tr>
<tr><td rowspan="4">Distributed (multi-process)</td><td rowspan="3">Common</td><td rowspan="2">Small</td><td rowspan="2">FLOWDISTsim</td><td>FLOWDISTsim</td></tr>
<tr><td>or FLOWDISTmul</td></tr>
<tr><td>Large</td><td>FLOWDIST</td><td>FLOWDIST</td></tr>
<tr><td colspan="2" rowspan="2">Specialized</td><td colspan="2">Kakute (for Spark)</td></tr>
<tr><td colspan="2">Pileus (for OpenStack), ...</td></tr>
<tr><td colspan="3">Single-process</td><td colspan="2">Phosphor, Joana, ...</td></tr>
</table>

Conclusion

- Application-level
- Multi-staged
- Refinement-based

More in paper

FLOWDIST: Multi-Staged Refinement-Based Dynamic Information Flow Analysis for Distributed Software Systems. *In Proceedings of the 30th USENIX Security Symposium (USENIX)*, 2021

xiaoqin.fu@wsu.edu haipeng.cai@wsu.edu