FlowTwist: Efficient Context-Sensitive Inside-Out Taint Analysis for Large Codebases

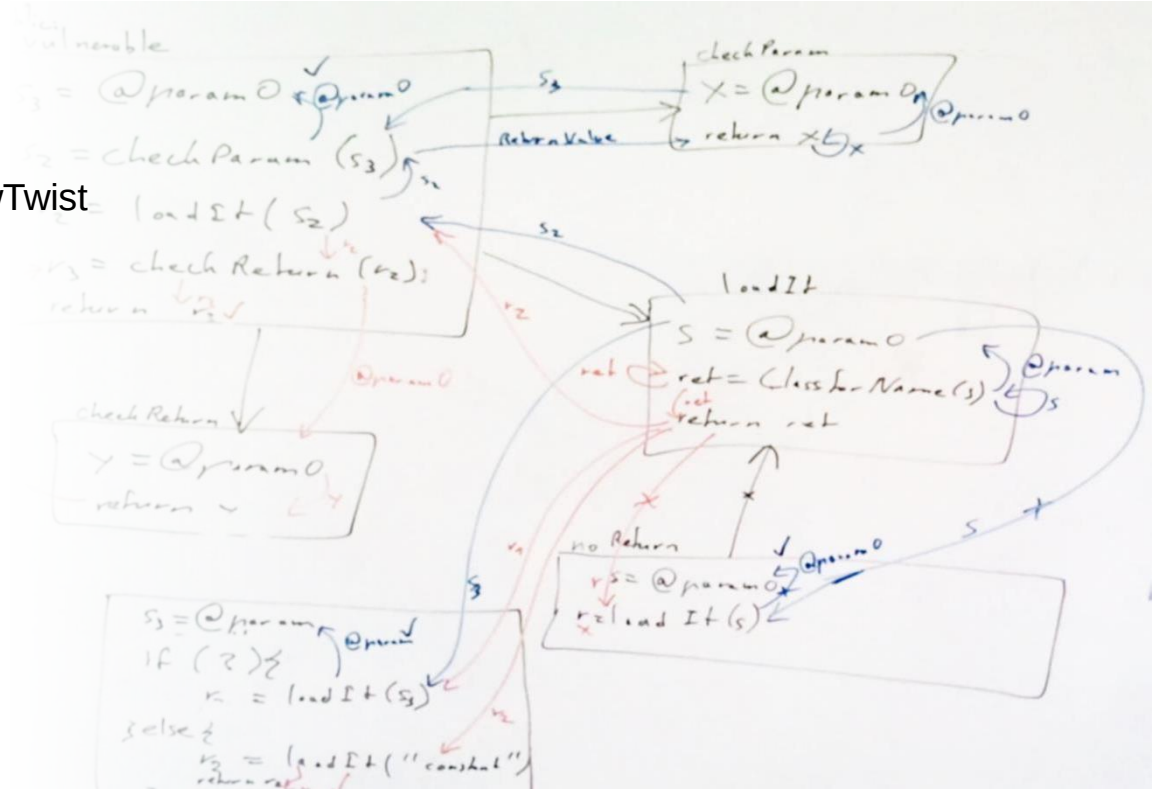
Johannes Lerch, Ben Hermann, Eric Bodden, and Mira Mezini

{lastname}@cs.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT

FLOW•TWIST

https://github.com/johanneslerch/FlowTwist



@stg_darmstadt

SOFTWARE TECHNOLOGY GROUP


Oracle pushed out new Java update to patch security holes
Released Friday, the latest critical patch update contains fixes for 50 different security flaws.
Feb 04, 2013
researchers warn that it's already being exploited "in the wild".
Jan 10, 2013
New vulnerabilities found in latest
Following ist latest updates, more
runtime.
Aug 31, 2012
Oracle issues emergency Java update to patch vulnerabilities
After hackers attack a new flaw in Java, "Oracle decided to release a fix for this vulnerability and another closely related bug as soon as possible".
Mar 04, 2013
repairs vulnerability that could allow remote attackers
to execute arbitrary
Jan 13, 2013
Homeland Security still advises disabling Java, even after update
DHS says an unpatched vulnerability may still put Web browsers using the plugin at risk of remote attack.
Jan 14, 2013
A new Java vulnerability
and therefore puts close
Sep 26, 2012

Theory: Stack-based Access Control

```
SecurityManager.checkPermission
SecurityManager.checkWrite
FileOutputStream.<init>
Attacker.doEvil
MyApplet.init
```

$$\cap = \varnothing$$

Reality: "Stack-based" Access Control

```
public static Class<?> forName(String className) throws ClassNotFoundException {
    return forName0(className,
                    true,
                    ClassLoader.getCallerClassLoader());
}
```

Implicit Permission Check

Class.forName	Privileged ClassLoader
Attacker.doEvil	Unprivileged ClassLoader
MyApplet.init	Unprivileged ClassLoader

Leak in sun.beans.finder.ClassFinder (CVE-2012-4681)

```
public static Class<?> findClass(String className)
{
    try
    {
        ClassLoader cl = ...

        ...

        return Class.forName(className, false, cl);
    }
    catch (ClassNotFoundException e)
    {

    }
    catch (SecurityException e)
    {

    }

    return Class.forName(className);
}
```

throws SecurityException

"handled" here

Leak in sun.beans.finder.ClassFinder (CVE-2012-4681)

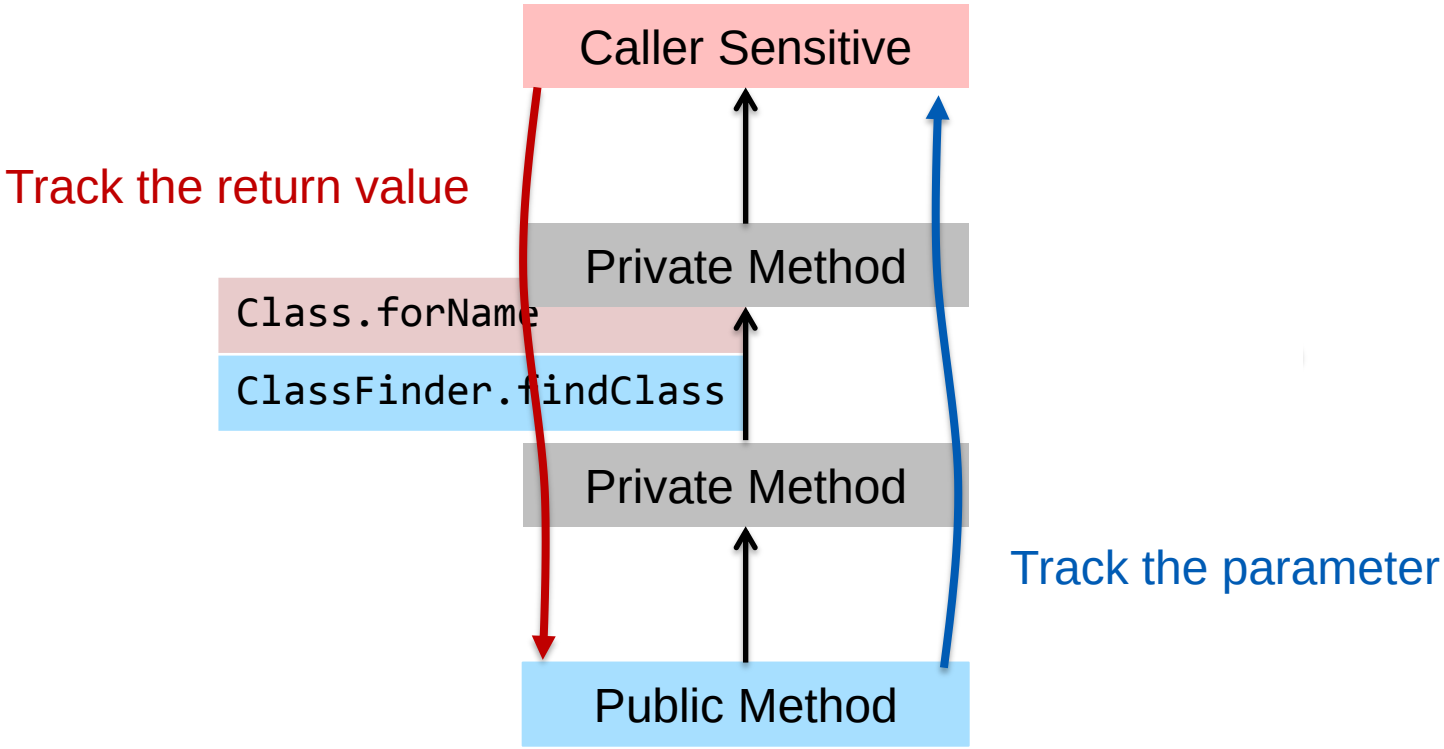
```
public static Class<?> findClass(String className)
{
    try
    {
        ClassLoader cl = ...

        ..

        re
    }
    catch
    {

    }
    catch
    {

    }

    return Class.forName(className);
}
```

Class.forName	Privileged ClassLoader
ClassFinder.findClass	Privileged ClassLoader
Attacker.doEvil	Unprivileged ClassLoader
MyApplet.init	Unprivileged ClassLoader

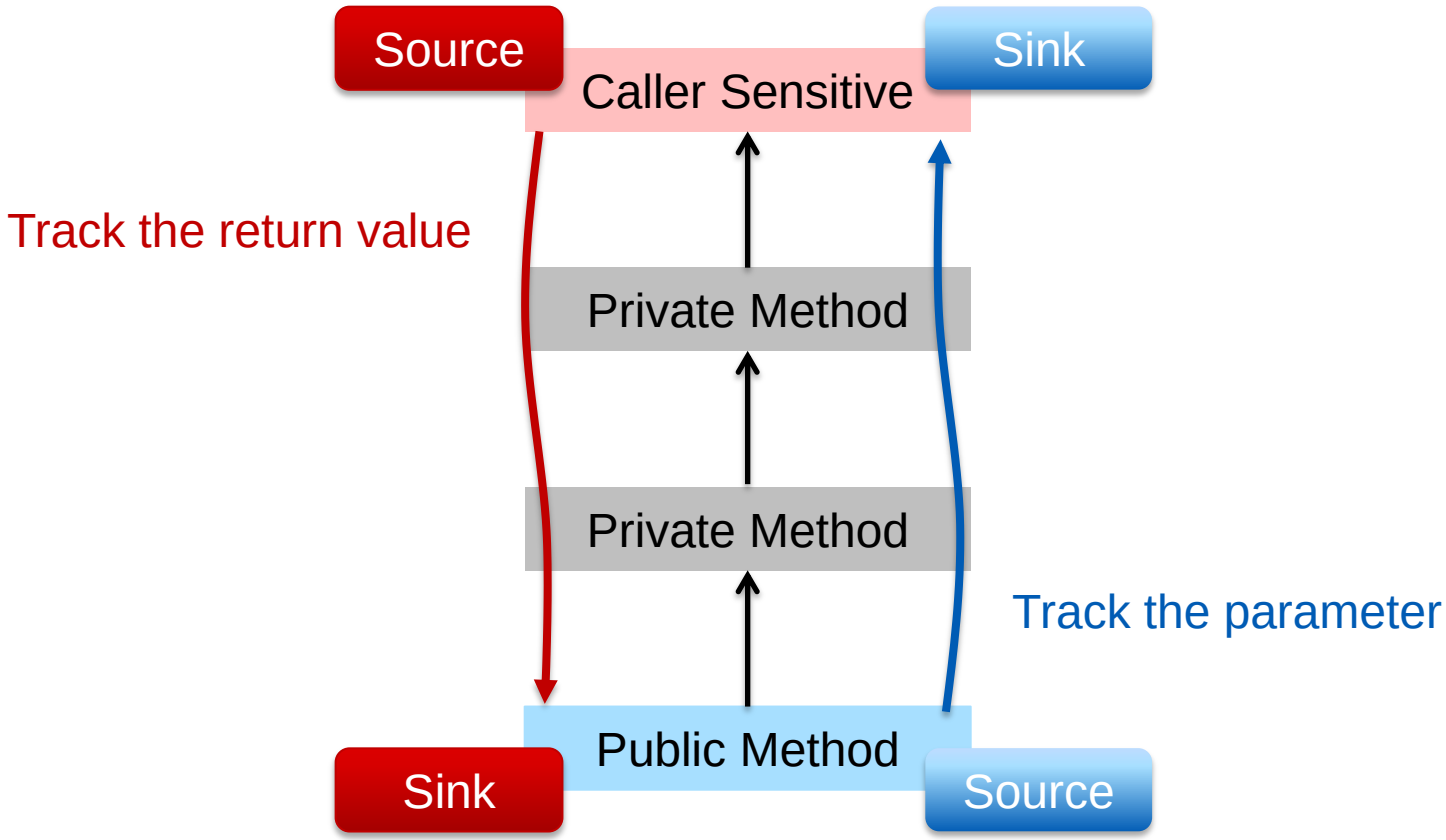
rityException

"handled" here

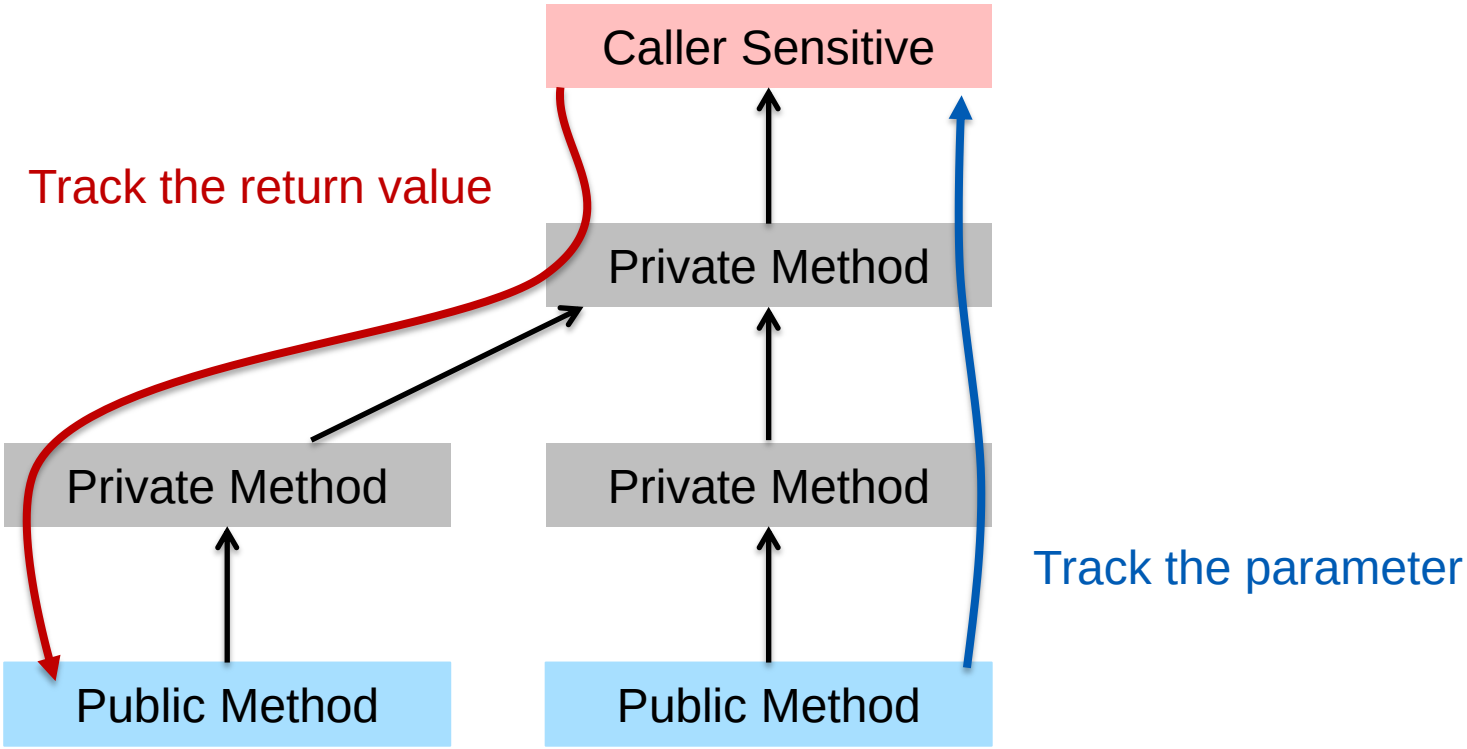
Deriving the Static Program Analysis Problem



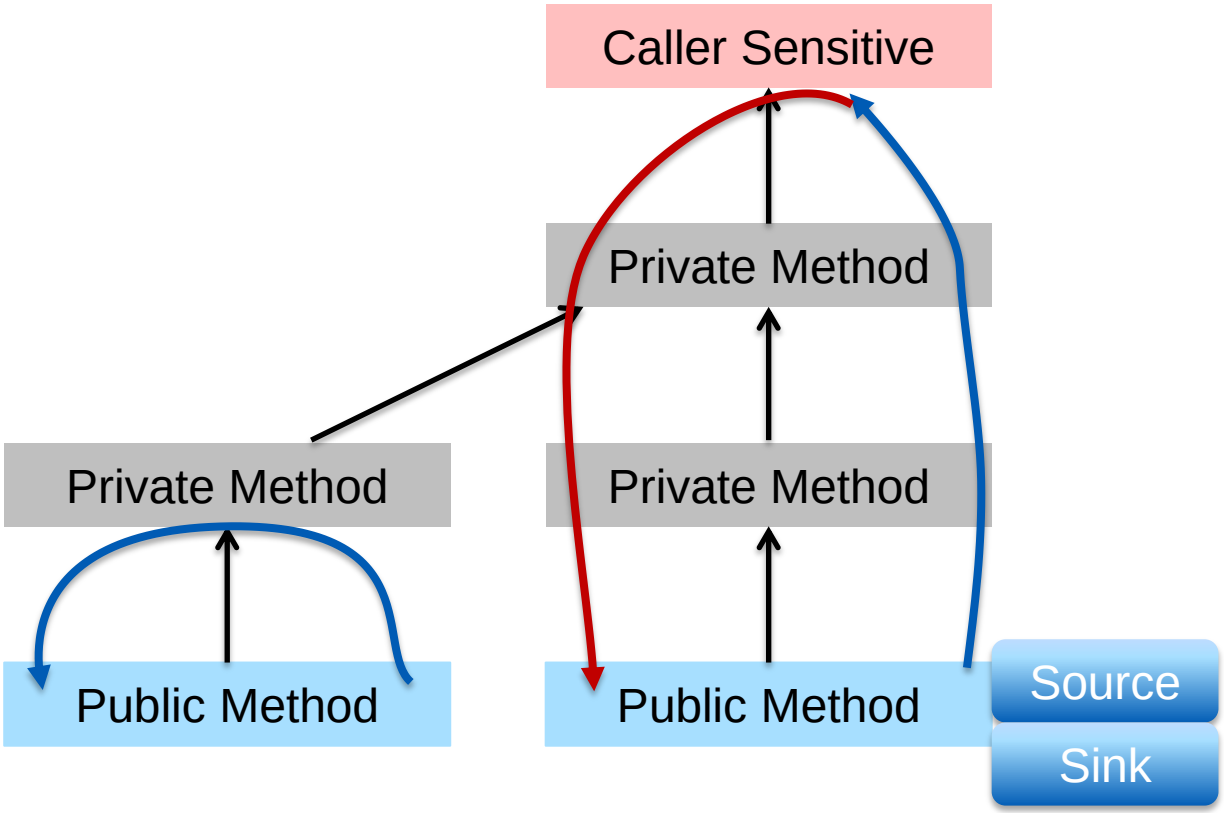
Two Independent Analyses



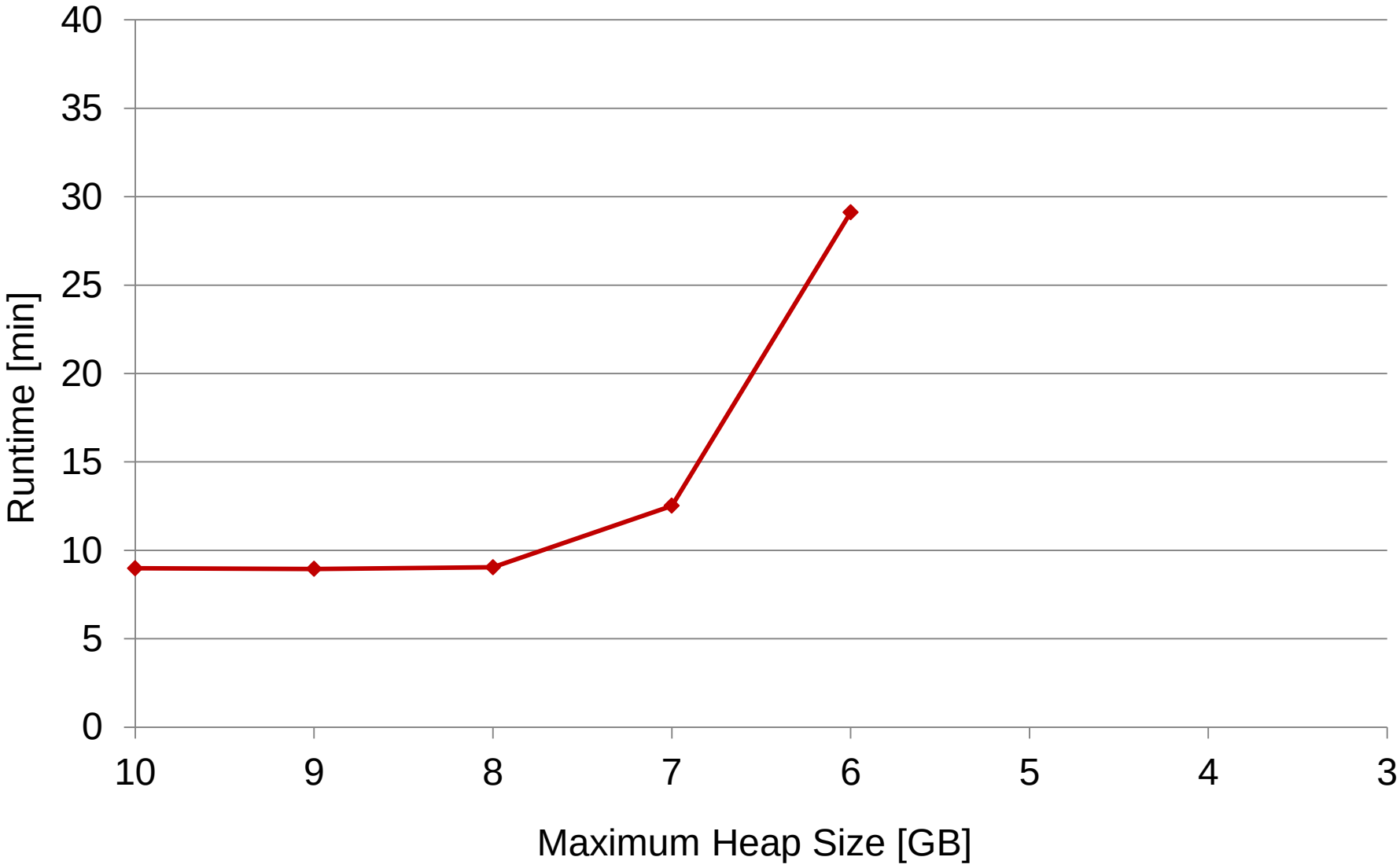
Two Independent Analyses: Not Context-Sensitive



Pure Forward Context-Sensitive Approach

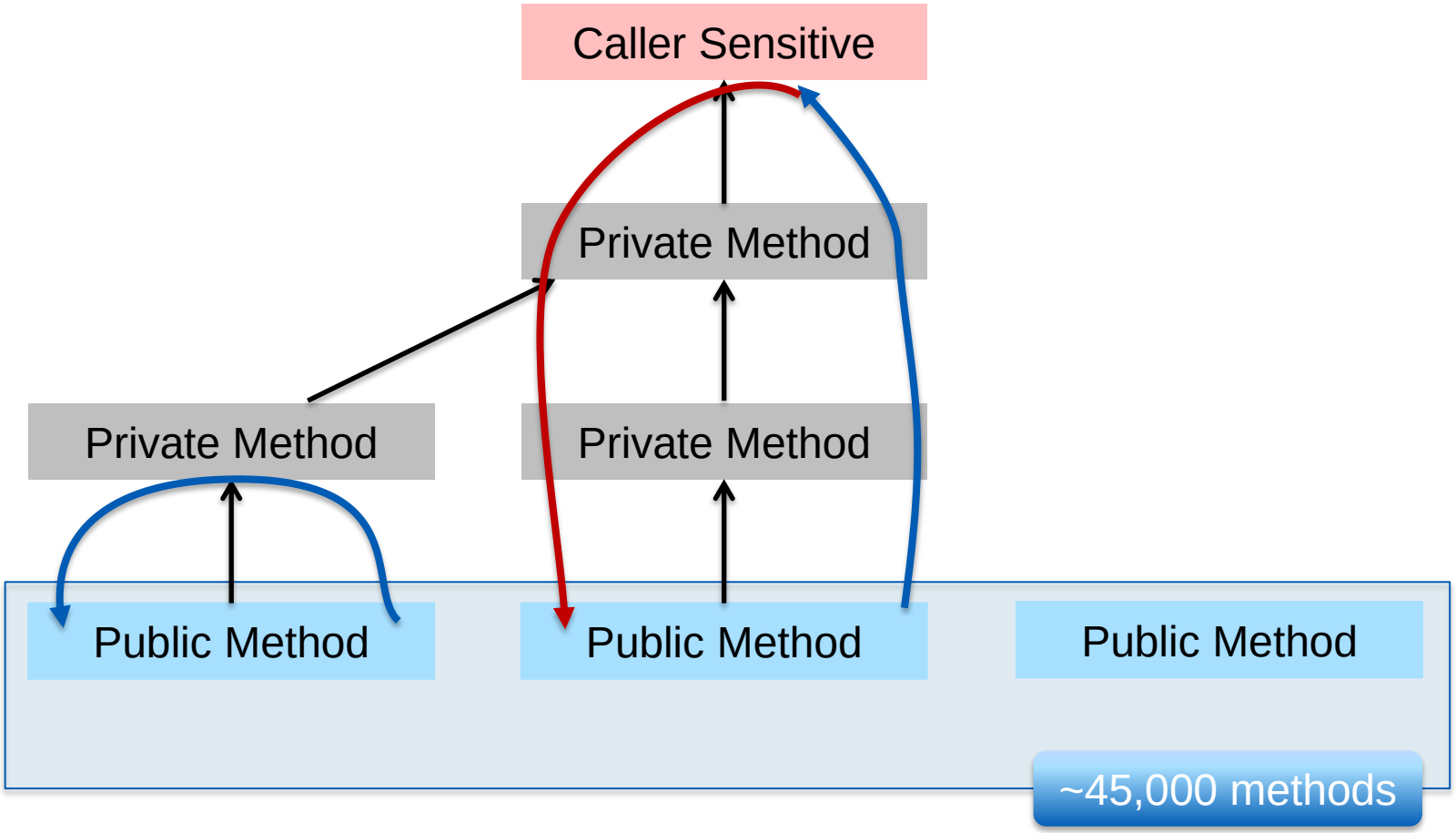


Results – only `Class.forName`

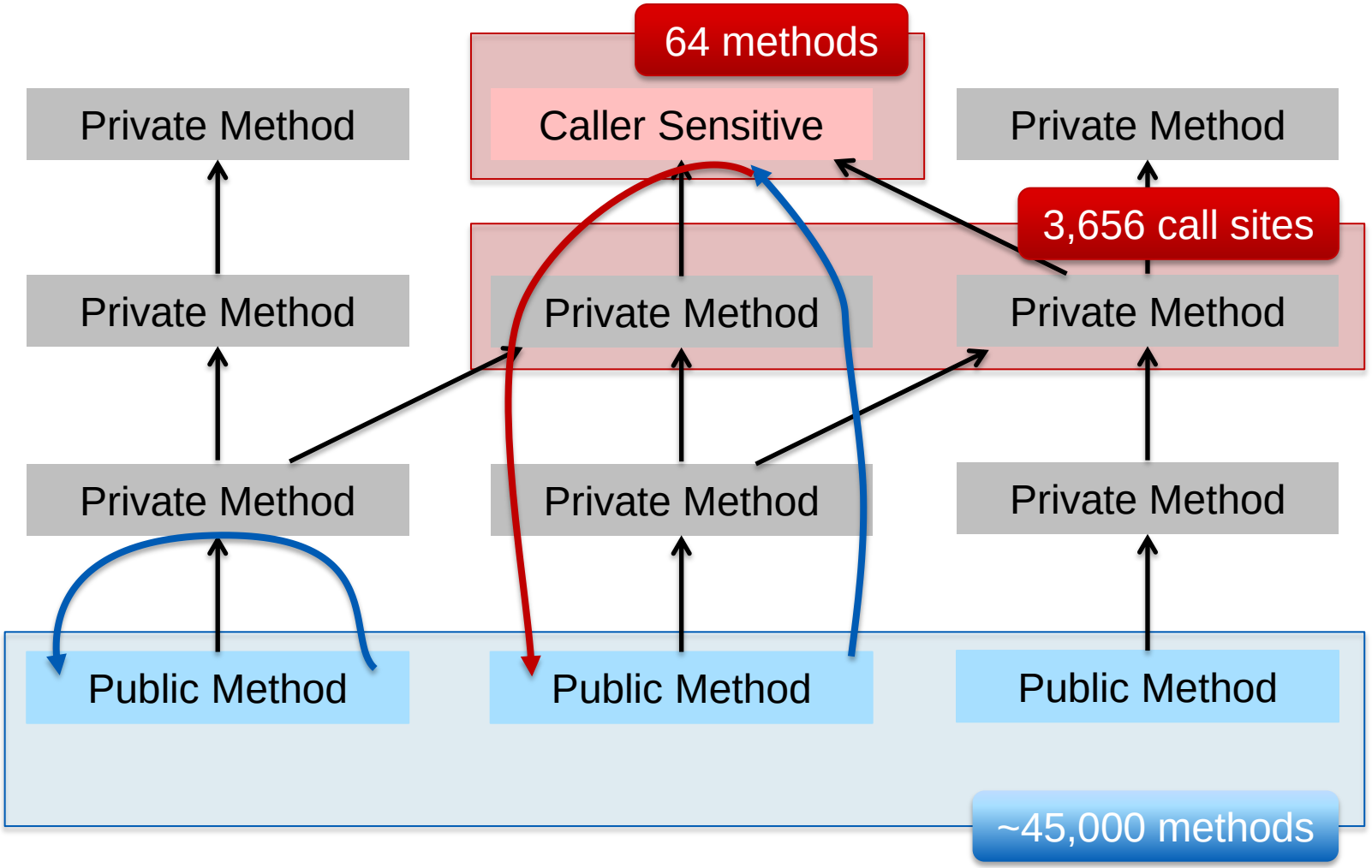


Results – all Caller Sensitive Methods

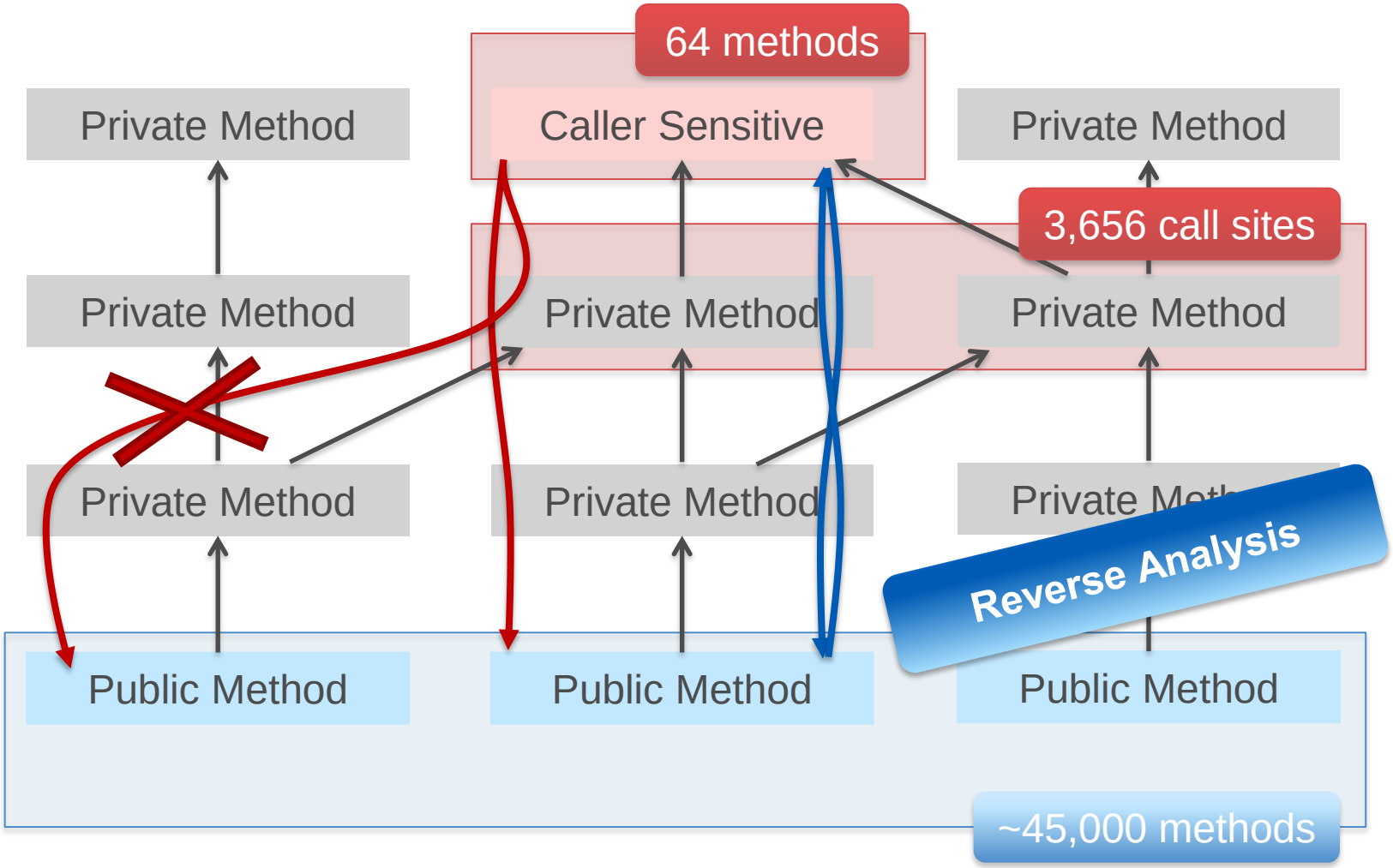
Scale of the Problem



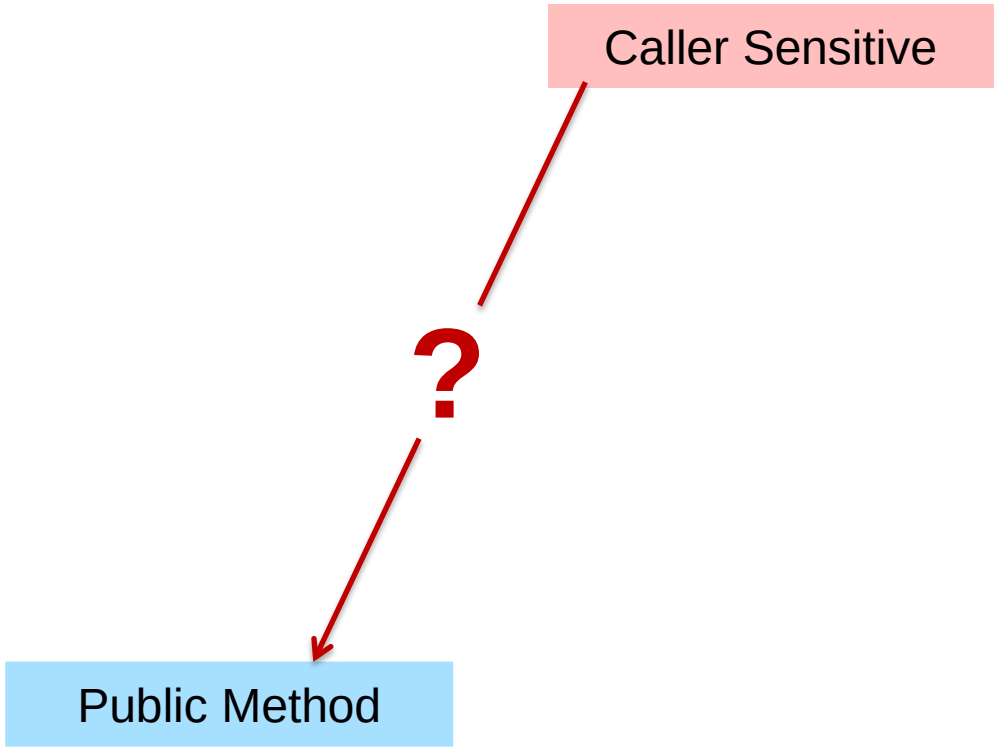
Scale of the Problem



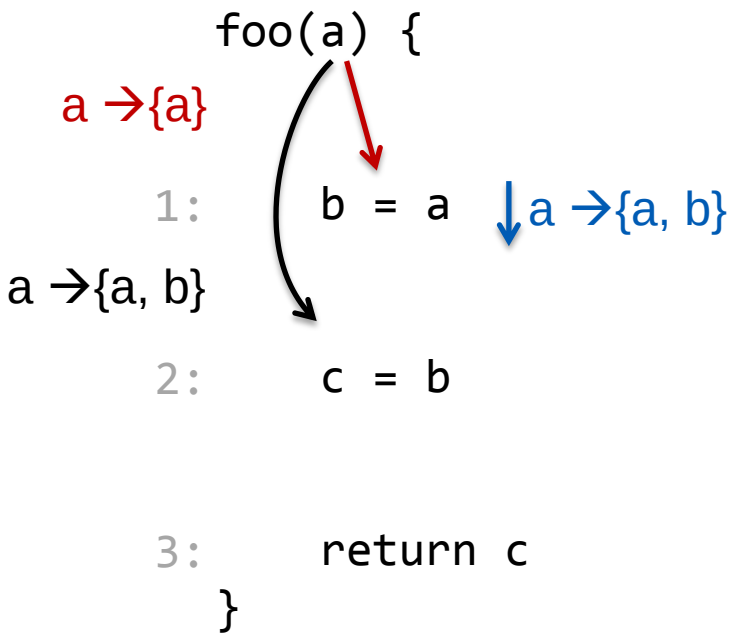
Exploit Imbalance to Improve Scalability



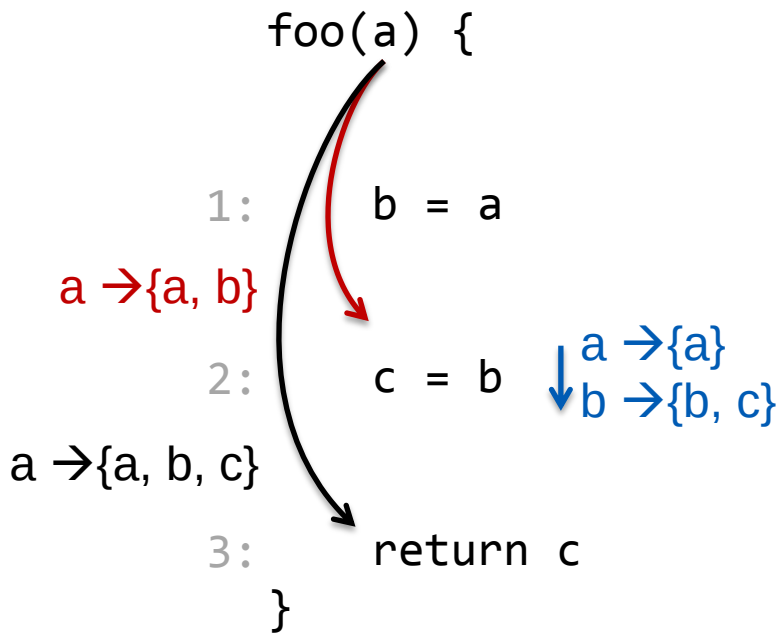
IFDS Algorithm [17,19] Reports Leaks



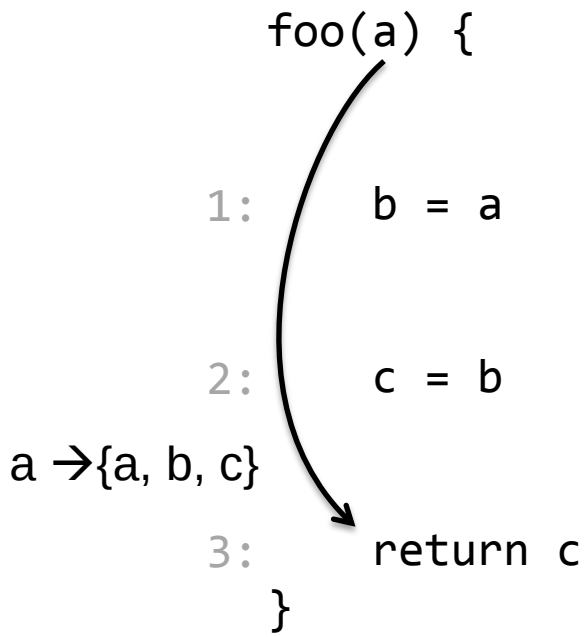
IFDS Algorithm: Computing Summaries



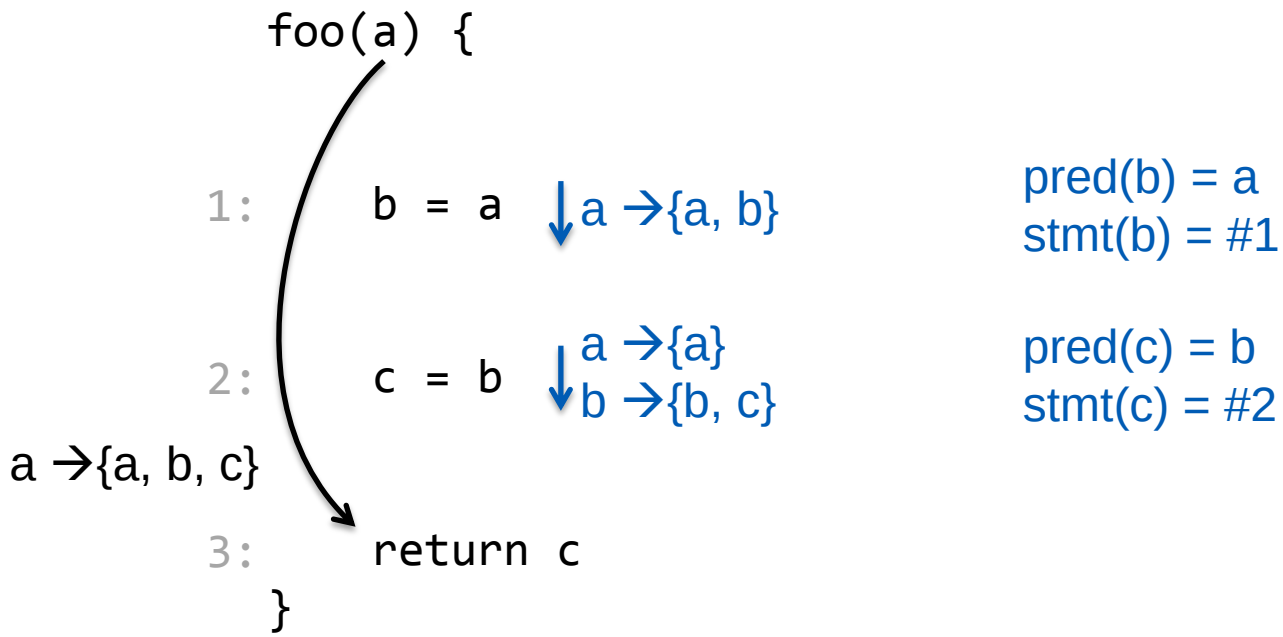
IFDS Algorithm: Computing Summaries



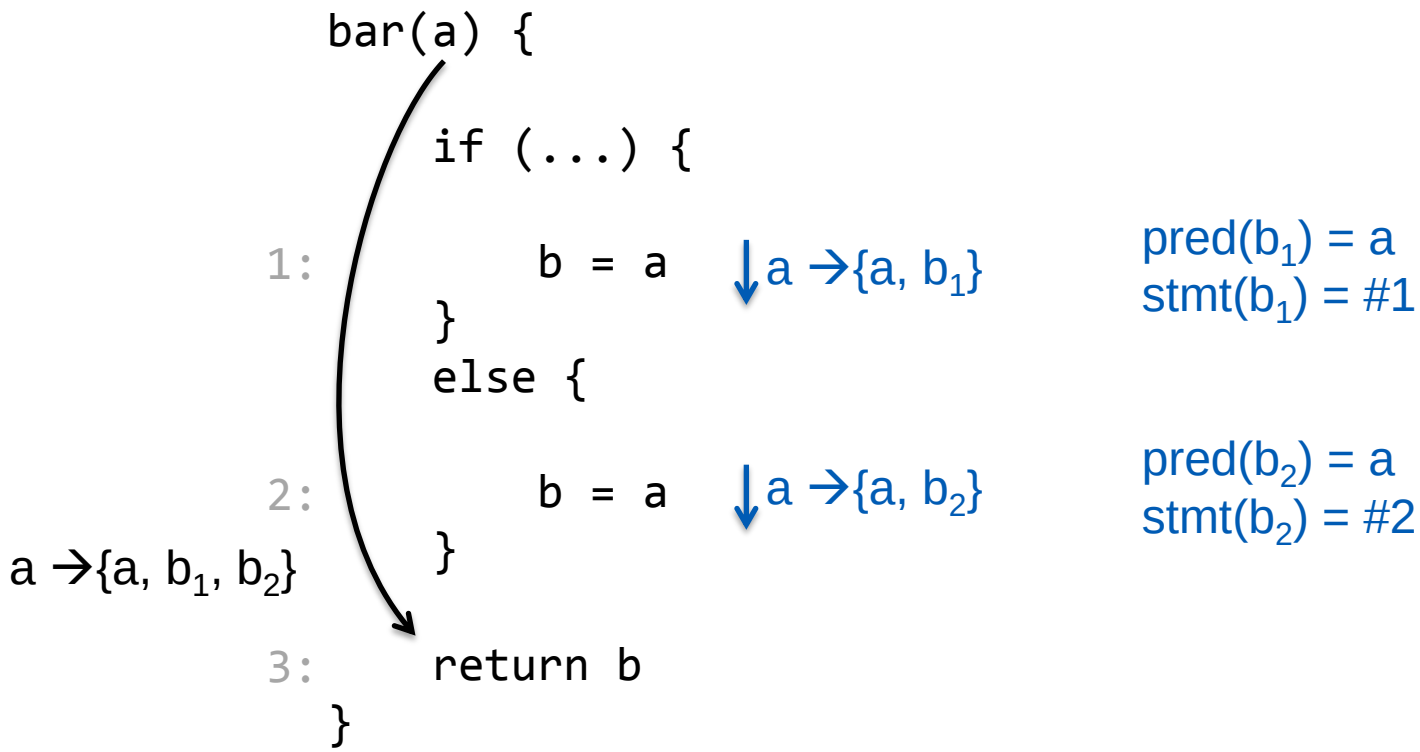
IFDS Algorithm: Computing Summaries



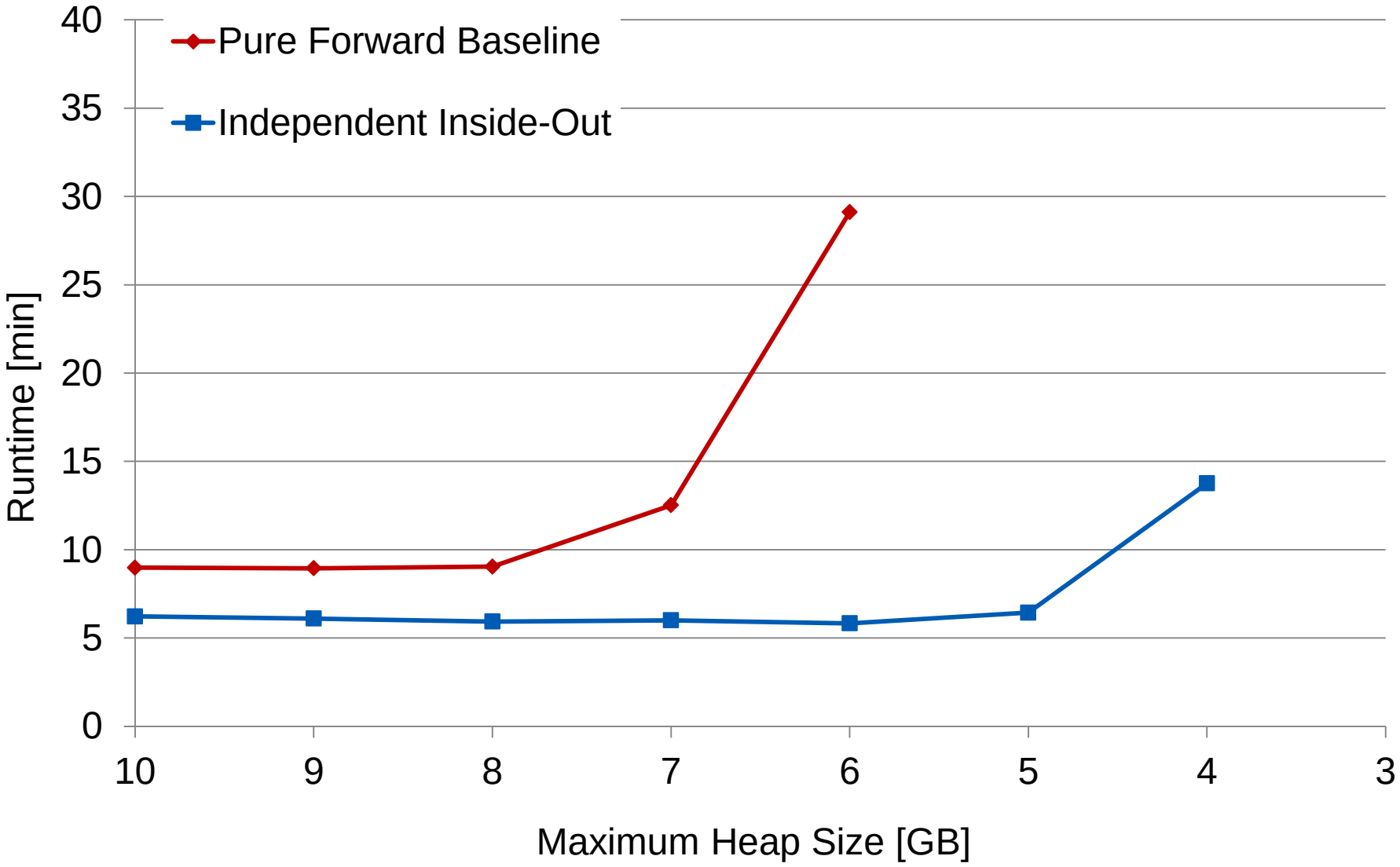
Path Construction



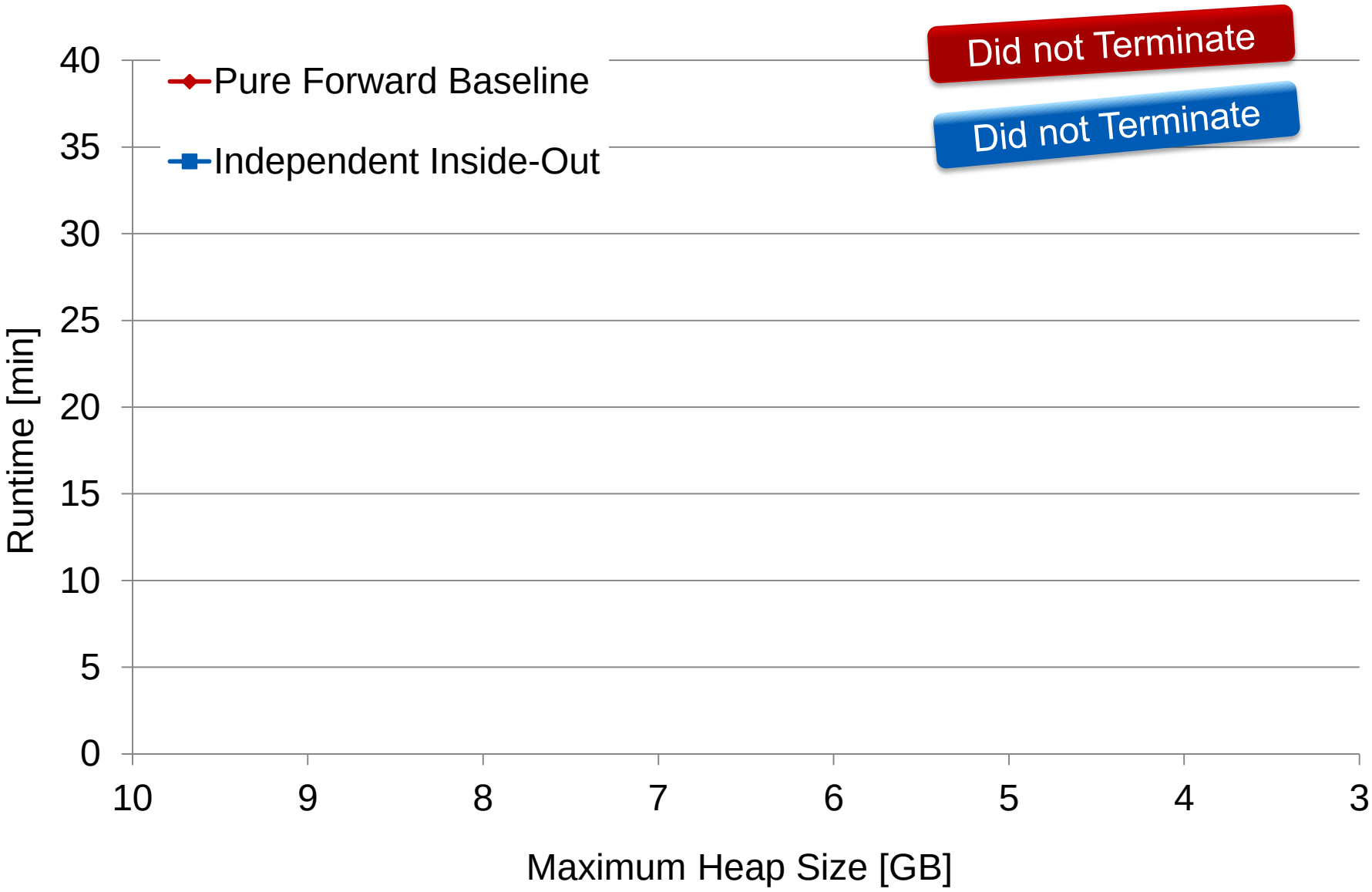
Path Construction: Merge at Branches



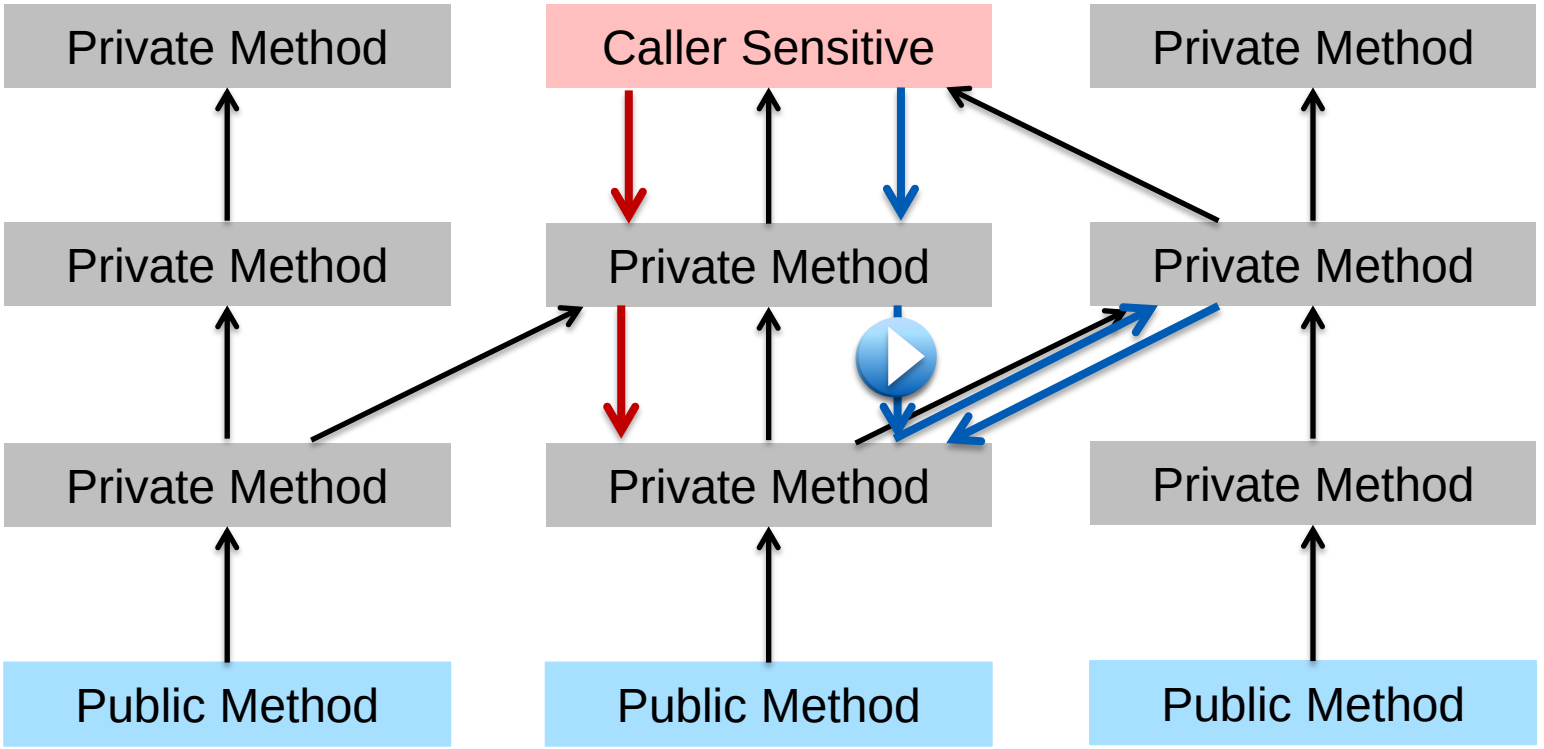
Results – only `Class.forName`



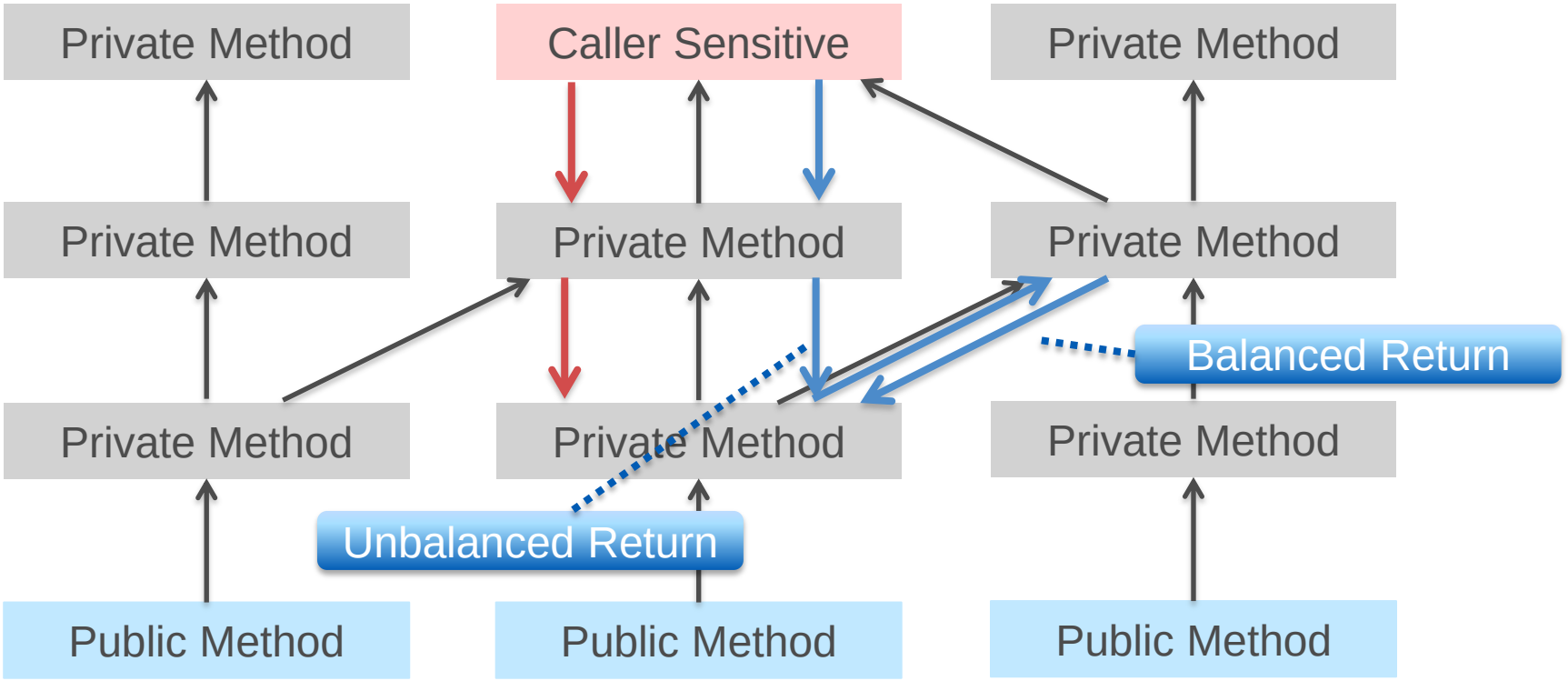
Results – all Caller Sensitive Methods



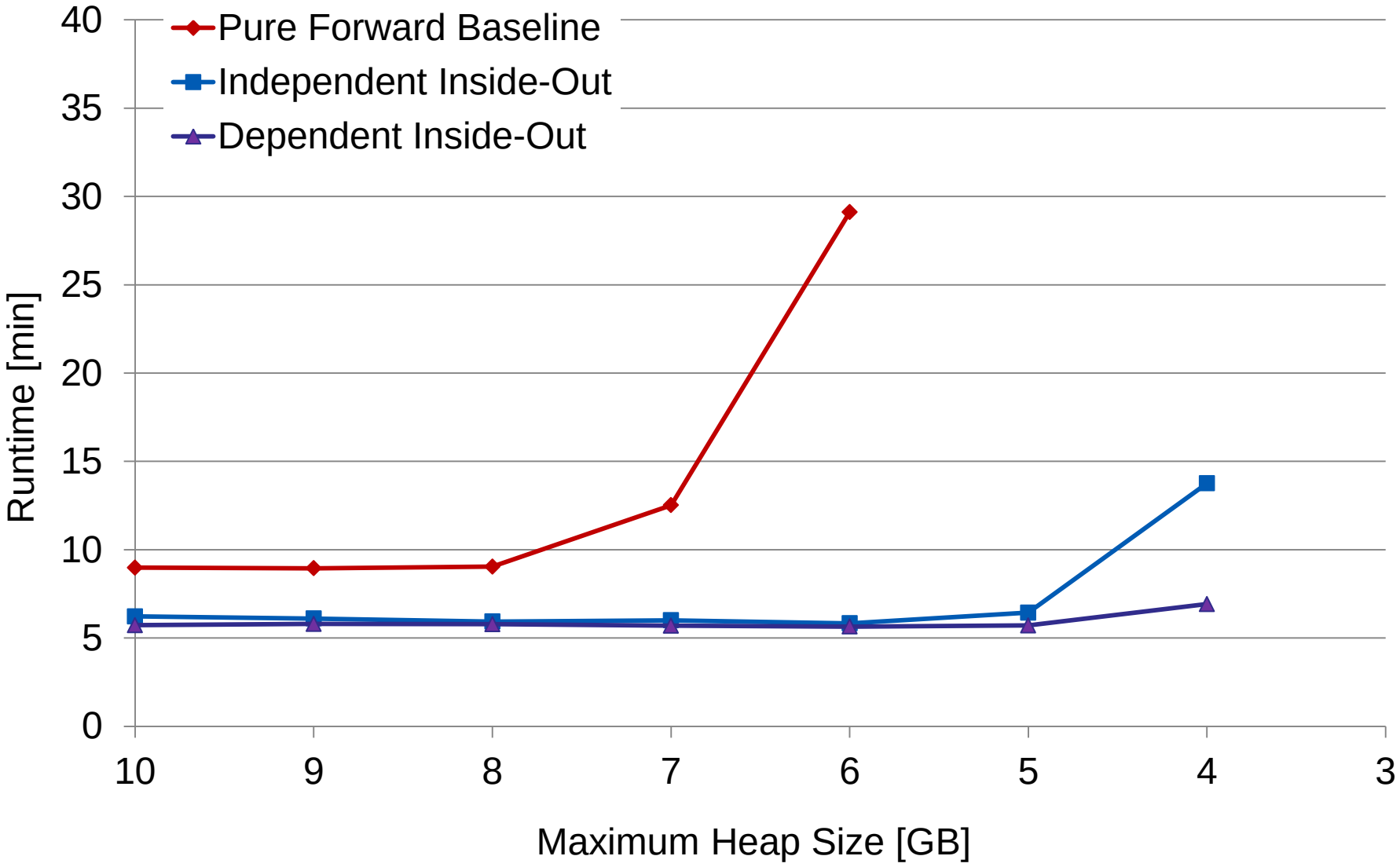
Two Synchronized/Dependent Analyses



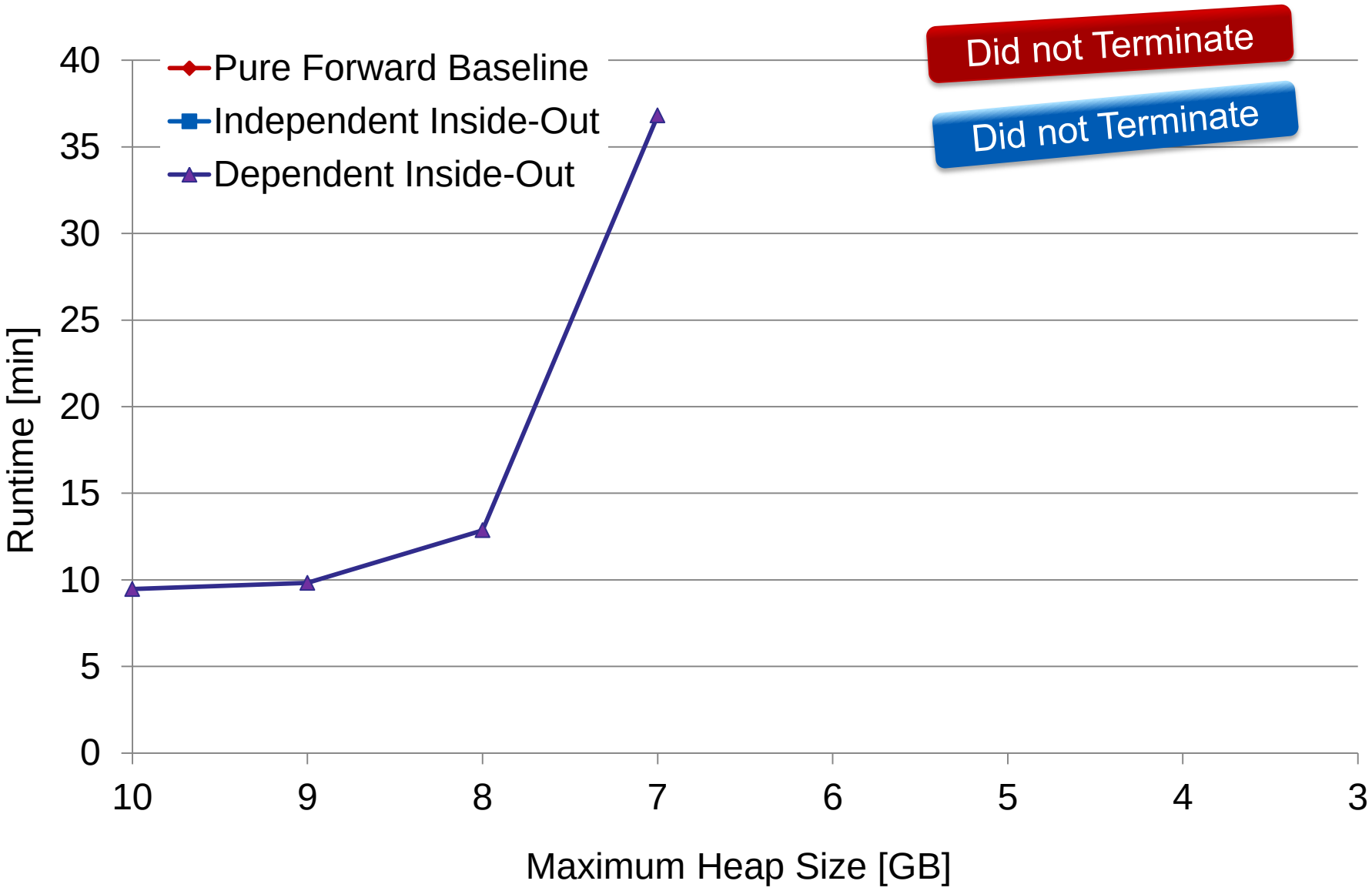
Two Synchronized/Dependent Analyses



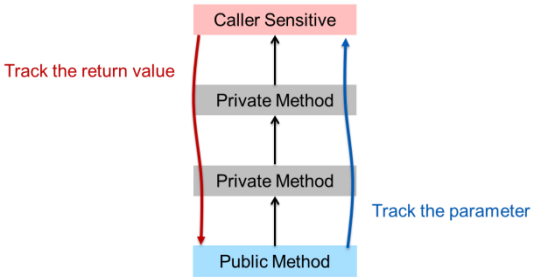
Results – only `Class.forName`

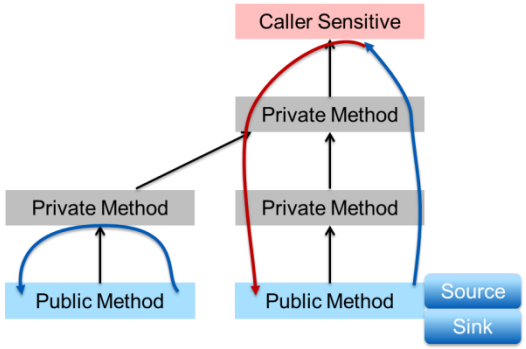


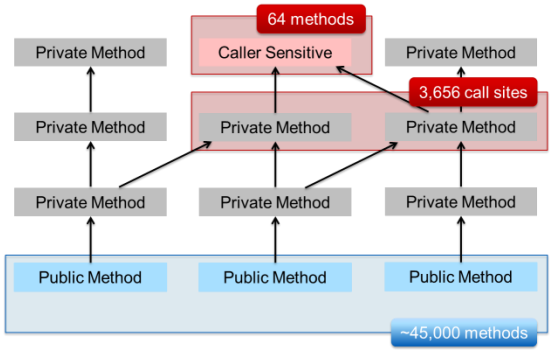
Results – all Caller Sensitive Methods

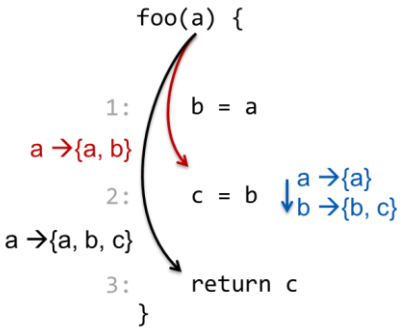


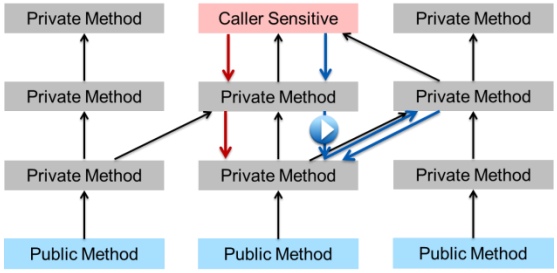
Summary


Caller Sensitive
Track the return value
Private Method
Private Method
Track the parameter
Public Method


Caller Sensitive
Private Method
Private Method
Private Method
Public Method
Public Method
Source
Sink


64 methods
Private Method
Caller Sensitive
Private Method
3,656 call sites
Private Method
Private Method
Private Method
Private Method
Private Method
Private Method
Public Method
Public Method
Public Method
~45,000 methods


foo(a) {
1:
b = a
a →{a, b}
2:
c = b
a →{a}
b →{b, c}
a →{a, b, c}
3:
return c
}


Private Method
Caller Sensitive
Private Method
Private Method
Private Method
Private Method
Private Method
Private Method
Private Method
Public Method
Public Method
Public Method


Pure Forward Baseline
Independent Inside-Out
Dependent Inside-Out
Did not Terminate
Did not Terminate
Runtime [min]
40
35
30
25
20
15
10
5
0
10
9
8
7
6
5
4
3
Maximum Heap Size [GB]