Cooperative Program
Analysis for Reliable
Mobile Applications

Bor-Yuh Evan Chang
University of Colorado Boulder

Huawei Vision Forum
October 21, 2015
A program analysis story ...
Software is everywhere and varying more and more
Software is everywhere and varying more and more
Software is everywhere and varying more and more

Software is getting more and more complex
Software is everywhere and varying more and more

Software is getting more and more complex
1980s  Bug in Therac-25 kills 6
1980s  Bug in Therac-25 kills 6

2000s: Conficker worm costs $9.1 billion in damages
1980s: Bug in Therac-25 kills 6

2000s: Conficker worm costs $9.1 billion in damages

Today: “Don’t buy this app, it crashes.”
How does program analysis save the day?

Program Analysis for Formal Verification

Systematically examine the program to “simulate” running it on “all inputs”
How does program analysis save the day?

Program Analysis for Formal Verification

Systematically examine the program to “simulate” running it on “all inputs”
Systematically examine the program to “simulate” running it on “all inputs”
Program Analysis for Formal Verification

Systematically examine the program to “simulate” running it on “all inputs”
The End?
Program Analysis for Formal Verification

Systematically examine the program to "simulate" running it on "all inputs"
The Ugly, Hidden Truth

Program Analysis for Formal Verification

Systematically examine the program to “simulate” running it on “all inputs”
Program Analysis for Formal Verification

Undecidability necessitates the possibility of false alarms. We hope not too many.
Uncooperative Program Analysis?
Oh Verifier, help me prove my program has no bugs
Uncooperative Program Analysis?

Oh Verifier, help me prove my program has no bugs

On line 142, there may be a bug
Oh
Verifier, help me prove my program has no bugs

On line 142, there may be a bug

Isn’t it obvious this can’t happen!?!?
Oh
Verifier, help me prove my program has no bugs

On line 142, there may be a bug

Isn’t it obvious this can’t happen!?!?

And noisily repeated over and over!
Uncooperative Program Analysis?

Oh Verifier, help me prove my program has no bugs.

On line 142, there may be a bug.

Isn’t it obvious this can’t happen!?!?

And noisily repeated over and over!

The well-known false alarm problem.
Uncooperative Program Analysis?

Oh Verifier, help me prove my program has no bugs!

On line 142, there may be a bug.

Isn't it obvious this can't happen!??!

The well-known false alarm problem.

And noisily repeated over and over!
"[M]ore than a 30% [false alarm rate] easily causes problems. True bugs get lost in the false. A vicious cycle starts where low trust causes complex [true] bugs to be labeled false [alarms], leading to yet lower trust."

"A stupid false [alarm] implies the tool is stupid."

The traditional approach to the false alarm problem focuses on improving the verifier.
The traditional approach to the false alarm problem focuses on improving the verifier.

Redesign the verifier with more magic to hopefully reduce the number of false alarms.
The traditional approach to the false alarm problem focuses on improving the verifier.

Redesign the verifier with more magic to hopefully reduce the number of false alarms.

But it can never be perfect (undecidability).
The traditional approach to the false alarm problem focuses on improving the verifier.

Redesign the verifier with more magic to hopefully reduce the number of false alarms.

But it can never be perfect (undecidability).

Also not a sufficient "excuse"
Research Agenda: The cooperative approach addresses the whole bug mitigation process.
Research Agenda: The cooperative approach addresses the whole bug mitigation process.
Research Agenda: The cooperative approach addresses the whole bug mitigation process.
Research Agenda: The cooperative approach addresses the whole bug mitigation process.
Research Agenda: The cooperative approach addresses the whole bug mitigation process.

Verifier

Thresher/Hopper: Goal-Directed Refutation Analysis
[Blackshear+ SAS’11, Blackshear+ PLDI’13, Blackshear+ OOPSLA’15]

Program

Manual Triaging

proof of no bug

Alarm Report

proof of no bug
Research Agenda: The cooperative approach addresses the whole bug mitigation process.
Research Agenda: The cooperative approach addresses the whole bug mitigation process.

Verifier

Program

Spec-
ification

Thresher/Hopper: Goal-Directed Refutation Analysis
[Blackshear+ SAS’11, Blackshear+ PLDI’13, Blackshear+ OOPSLA’15]

Manual

Triaging

Alarm

Report

proof of no bug

Program-
ming

✔
proof of no bug

✘
Research Agenda: The cooperative approach addresses the whole bug mitigation process.

Fissile Types: Checking Almost Everywhere Invariants
[<cite>Coughlin+ POPL’14</cite>]

Verifier

Program

Spec- ification

Thresher/Hopper: Goal-Directed Refutation Analysis
[<cite>Blackshear+ SAS’11, Blackshear+ PLDI’13, Blackshear+ OOPSLA’15</cite>]

Manual Triaging

Alarm Report

proof of no bug
Research Agenda: The cooperative approach addresses the whole bug mitigation process.

Fissile Types: Checking Almost Everywhere Invariants
[Coughlin+ POPL’14]

Programming

Test Input

Verifier

Test Output

proof of no bug

Alarm Report

Thresher/Hopper: Goal-Directed Refutation Analysis
[Blackshear+ SAS’11, Blackshear+ PLDI’13, Blackshear+ OOPSLA’15]
Research Agenda: The cooperative approach addresses the whole bug mitigation process.

**Fissile Types:** Checking Almost Everywhere Invariants

- [Coughlin+ POPL’14]

**Thresher/Hopper:** Goal-Directed Refutation Analysis

- [Blackshear+ SAS’11, Blackshear+ PLDI’13, Blackshear+ OOPSLA’15]

**Divva:** Synthesizing Short-Circuiting Data Structure Checks

- [in prep]

**Verifier**

**Program**

**Test Input**

**Speciﬁcation**

**Program**

**Test Output**

**Alarm Report**

**proof of no bug**

**Runner**

**Manual Triaging**

**Test Input**

**Speciﬁcation**

**Program**

**Test Output**

**Alarm Report**

**proof of no bug**
Research Agenda: The cooperative approach addresses the whole bug mitigation process.

**Fissile Types:** Checking Almost Everywhere Invariants
[Coughlin+ POPL’14]

**Divva:** Synthesizing Short-Circuiting Data Structure Checks
[in prep]

**Thresher/Hopper:** Goal-Directed Refutation Analysis
[Blackshear+ SAS’11, Blackshear+ PLDI’13, Blackshear+ OOPSLA’15]

**Manual Triaging**
Research Agenda: The cooperative approach addresses the whole bug mitigation process.

**Fissile Types:**
- Checking Almost Everywhere Invariants
  - [Coughlin+ POPL'14]

**Thresher/Hopper:**
- Goal-Directed Refutation Analysis
  - [Blackshear+ SAS’11, Blackshear+ PLDI’13, Blackshear+ OOPSLA’15]

**Divva:**
- Synthesizing Short-Circuiting Data Structure Checks
  - [in prep]

**Github**

**Program:**
- Manual Triaging

**Verifier**
-_runner

**Test Input**
- Program
  - Specified

**Test Output**
- Alarm Report

**Fixr:**
- Mining Bug Fixes from Commits
  - [in progress]
Research Agenda: The cooperative approach addresses the whole bug mitigation process.

- **Fissile Types:** Checking Almost Everywhere Invariants ([Coughlin+ POPL’14])

- **Thresher/Hopper:** Goal-Directed Refutation Analysis
  - [Blackshear+ SAS’11, Blackshear+ PLDI’13, Blackshear+ OOPSLA’15]

- **Divva:** Synthesizing Short-Circuiting Data Structure Checks
  - [in prep]

- **Jsana:** Abstract Domain Combinators for Dynamic Languages
  - [Cox+ ECOOP’13, Cox+ SAS’14, Cox+ ESOP’15]

- **Divva:** Synthesizing Short-Circuiting Data Structure Checks
  - [in prep]

- **Fissile Types:** Checking Almost Everywhere Invariants
  - [Coughlin+ POPL’14]

- **Program:**
  - **Program**
  - **Spec**
  - **ification**
  - **Test**
  - **Input**
  - **Runner**
  - **Verifier**

- **Github:**

- **Fixr:** Mining Bug Fixes from Commits
  - [in progress]

- **Manual Triaging**

- **Alarm Report**

- **Test Output**

- **proof of no bug**
Research Agenda: The cooperative approach addresses the whole bug mitigation process.

**Fissile Types:** Checking Almost Everywhere Invariants

**Divva:** Synthesizing Short-Circuiting Data Structure Checks

**Jsana:** Abstract Domain Combinators for Dynamic Languages

**Thresher/Hopper:** Goal-Directed Refutation Analysis

**Fixr:** Mining Bug Fixes from Commits

**Program-Ling**

- Github
- Test Input
- Specification
- Program
- Manual Triaging
- Verifier

**Verifier**

- Test Output
- Alarm Report
- Proof of no bug
Research Agenda: The cooperative approach addresses the whole bug mitigation process.

**Fissile Types:** Checking Almost Everywhere Invariants [Coughlin+ POPL’14]

**Divva:** Synthesizing Short-Circuiting Data Structure Checks [in prep]

**Program:**
- Test Input
- Specification
- Program

**Runner**
- Test Output

**Manual Triaging**

**Program:**
- Github
- BugFix
- Fixr: Mining Bug Fixes from Commits [in progress]

**Jsana:** Abstract Domain Combinators for Dynamic Languages
- Checking Almost Everywhere Invariants [Coughlin+ POPL’14]

**Auditr:** Securing Against “Inputs of Coma and Exposure” [in progress]

**Thresher/Hopper:** Goal-Directed Refutation Analysis
- [Blackshear+ SAS’11, Blackshear+ PLDI’13, Blackshear+ OOPSLA’15]
Research Agenda: The cooperative approach addresses the whole bug mitigation process.

**Fissile Types:** Checking Almost Everywhere Invariants  
[Coughlin+ POPL’14]

**This Talk**

**Thresher/Hopper:** Goal-Directed Refutation Analysis  
[Blackshear+ SAS’11, Blackshear+ PLDI’13, Blackshear+ OOPSLA’15]
This Talk: Highlights

**Thresher and Hopper: Goal-Directed Refutations for Heap Reachability**

Assist in triage of queries about heap relations

- **Idea:** Assume alarms false, prove them so automatically
- **Application:** Find memory leaks and eliminate crashes in Android
  - Filters out \( \sim 90\% \) of false alarms to expose true bugs
  - Going from \( \sim 450 \) hours of manual work to \( \sim 30 \) hours
- **Application:** Find crashes from unexpected event orderings
  - Prove \( \sim 92\% \) dereferences safe
This Talk: Highlights

Thresher and Hopper: Goal-Directed Refutations for Heap Reachability

Assist in triage of queries about heap relations

- Idea: Assume alarms false, prove them so automatically
- Application: Find memory leaks and eliminate crashes in Android
  - Filters out ~90% of false alarms to expose true bugs
  - Going from ~450 hours of manual work to ~30 hours
- Application: Find crashes from unexpected event orderings
  - Prove ~92% dereferences safe

Fissile Types: Checking Reflection with Almost Everywhere Invariants

Strengthen type checking with symbolic analysis

- Interactive checking speeds: making IDE integration possible
- Application: Prevent “MethodNotFound” errors in Objective-C (MacOS/iOS)
This Talk: Highlights

**Thresher and Hopper**: Goal-Directed Refutations for Heap Reachability

Assist in triage of queries about heap relations

- Idea: Assume alarms false, prove them so automatically
- Application: Find memory leaks and **eliminate crashes in Android**
  - Filters out $\sim 90\%$ of false alarms to **expose true bugs**
  - Going from $\sim 450$ hours of manual work to $\sim 30$ hours
- Application: Find crashes from unexpected event orderings
  - Prove $\sim 92\%$ dereferences safe

**Fissile Types**

Strengthen type checking with symbolic analysis

- Interactive checking speeds: making
- Application: Prevent “MethodNotFound” errors in Objective-C (MacOS/iOS)
Goal-Directed Refutations for Heap Reachability
A bug that manifests spectacularly ...
A bug that manifests spectacularly ...
A bug that manifests spectacularly ...
A bug that manifests spectacularly ... Crash
A bug that manifests spectacularly ... Crash
Android memory leaks underly rotation-based crashes.
Android memory leaks underly rotation-based crashes.
Android memory leaks underly rotation-based crashes.

How can you have memory leaks with a garbage collected run-time?
How can you have memory leaks with a garbage collected run-time?

Activity objects encapsulate the UI
How can you have memory leaks with a garbage collected run-time?

Activity objects encapsulate the UI

Android OS

of type Activity
How can you have memory leaks with a garbage collected run-time?

Activity objects encapsulate the UI
How can you have memory leaks with a garbage collected run-time?

Activity objects encapsulate the UI
How can you have memory leaks with a garbage collected run-time?

```
Android OS
  ↓
  a_static_field
  ↓
  program heap
  ↓
  of type Activity
  ↓
  of type Activity
```

Activity objects encapsulate the UI
How can you have memory leaks with a garbage collected run-time?

Android OS

a_static_field

program heap

of type Activity

of type Activity

I can’t collect this dead Activity!

Activity objects encapsulate the UI
How can you have memory leaks with a garbage collected run-time?

Activity objects encapsulate the UI
How can you have memory leaks with a garbage collected run-time?

Bug: Holding reference to "old" Activity
How can you have memory leaks with a garbage collected run-time?

```
Android OS

(program heap)

I'm full of garbage!

a_static_field

Activity objects encapsulate the UI

I can't collect this dead Activity!

Bug: Holding reference to "old" Activity
```

"an Activity leak"
The expert recommendation ...
The expert recommendation ...

Avoiding memory leaks

Android applications are, at least on the T-Mobile G1, limited to 16 MB of heap. It's both a lot of memory for a phone and yet very little for what some developers want to achieve. Even if you do not plan on using all of this memory, you should use as little as possible to let other applications run without getting them killed. The more applications Android can keep in memory, the faster it will be for the user to switch between his apps. As part of my job, I run into memory leaks issues in Android applications and they are most of the time due to the same mistake: keeping a long-lived reference to a Context.

In Android, a Context is used for many operations but mostly to load and access resources. This is why all the widgets receive a Context as parameter in their constructor. In a regular Android application, you usually have two kinds of Contexts: Activity and Application. It's usually the first one that the developer passes to classes and methods that need a Context:

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    TextView label = new TextView(this);
    label.setText("Leaks are bad");
```
The expert recommendation ...

“Do not keep long-lived references to a context-activity”
The expert recommendation ...

“Do not keep long-lived references to a context-activity”

I don’t know how I created a long-lived reference to an Activity!
The expert recommendation ...

“Do not keep long-lived references to a context-activity”

I don’t know how I created a long-lived reference to an Activity!

Often: A misunderstanding of a library causes the library to keep the Activity reference.
The state of practice in debugging Activity leaks ...
The state of practice in debugging Activity leaks ...

1. Run the app
The state of practice in debugging Activity leaks ...

1. Run the app
2. Watch the heap usage
The state of practice in debugging Activity leaks ...

1. Run the app
2. Watch the heap usage
3. Dump the heap. Dig around and hope to find the culprit
The state of practice in debugging Activity leaks ...

Suppose we’re lucky and find a possible culprit. Now what?

- Where in the code is this object allocated?
- What about the object that references it?
- Where is the reference created?
- Is this reference needed?
- For what periods?

3. Dump the heap. Dig around and hope to find the culprit.
The state of practice in debugging Activity leaks ...

Suppose we’re lucky and find a possible culprit. Now what?

- Where in the code is this object allocated?
- What about the object that references it?
- Where is the reference created?
- Is this reference needed?
- For what periods?

“One of the most dreaded bugs in Android is a memory leak. They are nasty because one piece of code causes an issue and in some other piece of code, your application crashes.” – http://therockncoder.blogspot.com/2012/09/fixing-android-memory-leak.html
Answering “Is there an Activity leak?” with program analysis ...

Can an object ever be reached from another object via pointer dereferences?

Example
Answering “Is there an **Activity** leak?” with program analysis ... 

Can an object ever be reached from another object via pointer dereferences?

Is there a program execution where at some time

![Diagram with arrows pointing from `a_static_field` to `of type Activity`.

Can be answered with a points-to analysis.
Answering “Is there an Activity leak?” with program analysis ...

Can an object ever be reached from another object via pointer dereferences?

Example

Is there a program execution where at some time

\[ \text{a_static_field} \]

\[ \ldots \]

\[ \text{of type Activity} \]

Can be answered with a points-to analysis

Hidden Truth

with approximation
Answering “Is there an Activity leak?” with program analysis ...

Can an object ever be reached from another object via pointer dereferences?

Example

Is there a program execution where at some time

\[\text{a\_static\_field}\]

of type \textit{Activity}

Can be answered with a points-to analysis

with approximation

Some pointer relations may be false

Hidden Truth
But with the cooperative approach ...
But with the cooperative approach ...
Thresher addresses alarm triage in a particularly challenging domain.
Thresher addresses alarm triage in a particularly challenging domain.
Thresher addresses alarm triage in a particularly challenging domain.

Known: Precise points-to analysis challenging
Thresher addresses alarm triage in a particularly challenging domain.

Known: Precise points-to analysis challenging
Thresher addresses alarm triage in a particularly challenging domain.

Known: Precise points-to analysis challenging
Thresher addresses alarm triage in a particularly challenging domain.

Known: Precise points-to analysis challenging.

Manual

Verifier

Points-To Analyzer

Program

Points-To Facts

✔ proof of no bug

✘

Hind. “Pointer Analysis: Haven’t We Solved This Problem Yet?”
Thresher addresses alarm triage in a particularly challenging domain.

Hind. “Pointer Analysis: Haven’t We Solved This Problem Yet?”
- 75 papers, 9 PhD theses

Known: Precise points-to analysis challenging
Thresher addresses alarm triage in a particularly challenging domain.

Known: Precise points-to analysis challenging

Hind. “Pointer Analysis: Haven’t We Solved This Problem Yet?” (2001)

- 75 papers, 9 PhD theses
Thresher addresses alarm triage in a particularly challenging domain.

- **Verifier**
  - Program
  - Points-To Analyzer
  - Points-To Facts

- Manual

Hind. “Pointer Analysis: Haven’t We Solved This Problem Yet?” (2001)
  - 75 papers, 9 PhD theses

Known: Precise points-to analysis challenging
Thresher addresses alarm triage in a particularly challenging domain.

Hind. “Pointer Analysis: Haven’t We Solved This Problem Yet?” (2001)
- 75 papers, 9 PhD theses

Dagstuhl 13162: Pointer Analysis (2013)

Known: Precise points-to analysis challenging
Thresher addresses alarm triage in a particularly challenging domain.

Known: Precise points-to analysis challenging enough?
Manual triage is particularly hard for heap reachability reports.
Manual triage is particularly hard for heap reachability reports.
Manual triage is particularly hard for heap reachability reports.

allocated here

MyClass1.java

LibraryClass1.java
Manual triage is particularly hard for heap reachability reports.

allocated here

MyClass1.java

LibraryClass1.java

MyClass2.java
Manual triage is particularly hard for heap reachability reports.
Manual triage is particularly hard for heap reachability reports.
Manual triage is particularly hard for heap reachability reports.

allocated here

MyClass1.java

LibraryClass1.java

MyClass2.java

Library2Class1.class

java.util.HashMap.class

MyClass3.java
Manual triage is particularly hard for heap reachability reports.

Get abstract heap path + maybe allocation sites
Guesstimate: >1 to 2 hours per alarm to triage “well”
Examining manual triage ...
Examining manual triage ...
Examining manual triage ...

What does the user need to do with an alarm? He starts at, say, line 142 and traces back to see if a bug is possible given what’s happening.
What does the user need to do with an alarm? He starts at, say, line 142 and traces back to see if a bug is possible given what’s happening.

We can do this with analysis!
Examining manual triage ...

What does the user need to do with an alarm? He starts at, say, line 142 and traces back to see if a bug is possible given what’s happening.

We can do this with analysis!
Examining manual triage ...

What does the user need to do with an alarm? He starts at, say, line 142 and traces back to see if a bug is possible given what’s happening.

We can do this with analysis!

If we filter most false alarms, the user can triage more quickly and get to true bugs earlier (without frustration).
Thresher filters out false alarms by refuting them one-by-one.
Thresher filters out false alarms by refuting them one-by-one.
Thresher filters out false alarms by refuting them one-by-one.

Verifier

Points-To Analyzer (off-the-shelf)

Points-To Facts

Manual Triaging

Filter with Thresher

Program

Leak Alarms

✔ proof of no bug

✗
Thresher filters out false alarms by refuting them one-by-one.

Verifier

Points-To Analyzer (off-the-shelf)

Program -> Points-To analyzer

Points-To Facts

Filter with Thresher

Manual Triaging

Leak Alarms:

proof of no bug

✔

✗

Manual Triaging

Filter with Thresher

Points-To Analyzer (off-the-shelf)
Thresher filters out false alarms by refuting them one-by-one.
Thresher filters out false alarms by refuting them one-by-one.

Idea 1: Refute points-to on-demand with second "uber-precise" filter analysis
Thresher filters out false alarms by refuting them one-by-one.

Idea 1: Refute points-to on-demand with second “uber-precise” filter analysis

* -sensitive
Thresher filters out false alarms by refuting them one-by-one.

Idea 1: Refute points-to on-demand with second "uber-precise" filter analysis
Thresher filters out false alarms by refuting them one-by-one.

**Idea 1**: Refute points-to on-demand with second “uber-precise” filter analysis

**Idea 2**: Leverage the facts from the first analysis in the filter analysis to scale
Is Thresher effective at filtering?

Thresher analyzes Java VM bytecode

7 Android app benchmarks
2,000 to 40,000 source lines of code
+ 880,000 sources lines of Android framework code

Off-the-shelf, state-of-the-art points-to analysis from WALA
<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Points-To Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td>PulsePoint</td>
<td>unknown</td>
<td>16</td>
</tr>
<tr>
<td>StandupTimer</td>
<td>2K</td>
<td>25</td>
</tr>
<tr>
<td>DroidLife</td>
<td>3K</td>
<td>3</td>
</tr>
<tr>
<td>SMSPopUp</td>
<td>7K</td>
<td>5</td>
</tr>
<tr>
<td>aMetro</td>
<td>20K</td>
<td>54</td>
</tr>
<tr>
<td>K9Mail</td>
<td>40K</td>
<td>208</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>72K</td>
<td>311</td>
</tr>
</tbody>
</table>
Is Thresher effective at filtering?

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Points-To Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td>PulsePoint</td>
<td>unknown</td>
<td>16</td>
</tr>
<tr>
<td>StandupTimer</td>
<td>2K</td>
<td>25</td>
</tr>
<tr>
<td>DroidLife</td>
<td>3K</td>
<td>3</td>
</tr>
<tr>
<td>SMSPopUp</td>
<td>7K</td>
<td>5</td>
</tr>
<tr>
<td>aMetro</td>
<td>20K</td>
<td>54</td>
</tr>
<tr>
<td>K9Mail</td>
<td>40K</td>
<td>208</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>72K</strong></td>
<td><strong>311</strong></td>
</tr>
</tbody>
</table>

staticfield-Activity *pairs*
Is Thresher effective at filtering?

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Points-To Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td>PulsePoint</td>
<td>unknown</td>
<td>16</td>
</tr>
<tr>
<td>StandupTimer</td>
<td>2K</td>
<td>25</td>
</tr>
<tr>
<td>DroidLife</td>
<td>3K</td>
<td>3</td>
</tr>
<tr>
<td>SMSPopUp</td>
<td>7K</td>
<td>5</td>
</tr>
<tr>
<td>aMetro</td>
<td></td>
<td>54</td>
</tr>
<tr>
<td>K9Mail</td>
<td></td>
<td>208</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>311</strong></td>
</tr>
</tbody>
</table>

triage “well” at ~1–2 hours per alarm

staticfield-Activity *pairs*
<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Points-To Alarms</th>
<th>Thresher Refuted</th>
</tr>
</thead>
<tbody>
<tr>
<td>PulsePoint</td>
<td>unknown</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>StandupTimer</td>
<td>2K</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>DroidLife</td>
<td>3K</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>SMSPopUp</td>
<td>7K</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>aMetro</td>
<td>20K</td>
<td>54</td>
<td>18</td>
</tr>
<tr>
<td>K9Mail</td>
<td>40K</td>
<td>208</td>
<td>130</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>72K</strong></td>
<td><strong>311</strong></td>
<td><strong>172</strong></td>
</tr>
</tbody>
</table>

**staticfield-Activity pairs**

**Filtered**
Is Thresher effective at filtering?

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Points-To Alarms</th>
<th>Thresher Refuted</th>
<th>True Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PulsePoint</td>
<td>unknown</td>
<td>16</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>StandupTimer</td>
<td>2K</td>
<td>25</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>DroidLife</td>
<td>3K</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>SMSPopUp</td>
<td>7K</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>aMetro</td>
<td>20K</td>
<td>54</td>
<td>18</td>
<td>36</td>
</tr>
<tr>
<td>K9Mail</td>
<td>40K</td>
<td>208</td>
<td>130</td>
<td>64</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>72K</strong></td>
<td><strong>311</strong></td>
<td><strong>172</strong></td>
<td><strong>115</strong></td>
</tr>
</tbody>
</table>

filtered staticfield-Activity pairs
Is Thresher effective at filtering?

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Points-To Alarms</th>
<th>Thresher Refuted</th>
<th>True Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PulsePoint</td>
<td>unknown</td>
<td>16</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>StandupTimer</td>
<td>2K</td>
<td>25</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>DroidLife</td>
<td>3K</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>SMSPopUp</td>
<td>7K</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>aMetro</td>
<td>20K</td>
<td>54</td>
<td>18</td>
<td>36</td>
</tr>
<tr>
<td>K9Mail</td>
<td>40K</td>
<td>208</td>
<td>130</td>
<td>64</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>72K</strong></td>
<td><strong>311</strong></td>
<td><strong>172</strong></td>
<td><strong>115</strong></td>
</tr>
</tbody>
</table>

**staticfield- Activity pairs**

**Filtered**

**Manual**
Is Thresher effective at filtering?

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Points-To Alarms</th>
<th>Thresher Refuted</th>
<th>True Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PulsePoint</td>
<td>unknown</td>
<td>16</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>StandupTimer</td>
<td>2K</td>
<td>25</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>DroidLife</td>
<td>3K</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>SMSPopUp</td>
<td>7K</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>aMetro</td>
<td>20K</td>
<td></td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>K9Mail</td>
<td>40K</td>
<td></td>
<td>64</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>72K</strong></td>
<td><strong>16</strong></td>
<td><strong>115</strong></td>
<td></td>
</tr>
</tbody>
</table>

Triage "well" at 10–15 minutes per 

staticfield-
Activity pairs  
Filtered  
Manual
Is Thresher effective at filtering?

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Points-To Alarms</th>
<th>Thresher Refuted</th>
<th>True Bugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PulsePoint</td>
<td>unknown</td>
<td>16</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>StandupTimer</td>
<td>2K</td>
<td>25</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>DroidLife</td>
<td>3K</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>SMSPopUp</td>
<td>7K</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>aMetro</td>
<td>20K</td>
<td>54</td>
<td>18</td>
<td>36</td>
</tr>
<tr>
<td>K9Mail</td>
<td>40K</td>
<td>208</td>
<td>130</td>
<td>64</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>72K</td>
<td><strong>311</strong></td>
<td><strong>172</strong></td>
<td><strong>115</strong></td>
</tr>
</tbody>
</table>
Is Thresher effective at filtering?

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Points-To Alarms</th>
<th>Thresher Refuted</th>
<th>True Bugs</th>
<th>Thresher Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PulsePoint</td>
<td>unknown</td>
<td>16</td>
<td>8</td>
<td>8</td>
<td>95</td>
</tr>
<tr>
<td>StandupTimer</td>
<td>2K</td>
<td>25</td>
<td>15</td>
<td>0</td>
<td>1068</td>
</tr>
<tr>
<td>DroidLife</td>
<td>3K</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SMSPopUp</td>
<td>7K</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>46</td>
</tr>
<tr>
<td>aMetro</td>
<td>20K</td>
<td>54</td>
<td>18</td>
<td>36</td>
<td>18</td>
</tr>
<tr>
<td>K9Mail</td>
<td>40K</td>
<td>208</td>
<td>130</td>
<td>64</td>
<td>374</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>72K</td>
<td><strong>311</strong></td>
<td><strong>172</strong></td>
<td><strong>115</strong></td>
<td><strong>1602</strong></td>
</tr>
</tbody>
</table>
Is Thresher effective at filtering?

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Points-To Alarms</th>
<th>Thresher Refuted</th>
<th>True Bugs</th>
<th>Thresher Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PulsePoint</td>
<td>unknown</td>
<td>16</td>
<td>8</td>
<td>8</td>
<td>95</td>
</tr>
<tr>
<td>StandupTimer</td>
<td>2K</td>
<td>25</td>
<td>15</td>
<td>0</td>
<td>1068</td>
</tr>
<tr>
<td>DroidLife</td>
<td>3K</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SMSPopUp</td>
<td>7K</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>46</td>
</tr>
<tr>
<td>aMetro</td>
<td>20K</td>
<td>54</td>
<td>18</td>
<td>36</td>
<td>18</td>
</tr>
<tr>
<td>K9Mail</td>
<td>40K</td>
<td>208</td>
<td>130</td>
<td>64</td>
<td>374</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>72K</td>
<td>311</td>
<td>172</td>
<td>115</td>
<td><strong>1602</strong></td>
</tr>
</tbody>
</table>

< ~coffee to lunch break>
Is Thresher effective at filtering?

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Points-To Alarms</th>
<th>Thresher Refuted</th>
<th>True Bugs</th>
<th>Thresher Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PulsePoint</td>
<td>unknown</td>
<td>16</td>
<td>8</td>
<td>8</td>
<td>95</td>
</tr>
<tr>
<td>StandupTimer</td>
<td>2K</td>
<td>25</td>
<td>15</td>
<td>0</td>
<td>1068</td>
</tr>
<tr>
<td>DroidLife</td>
<td>3K</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>SMSPopUp</td>
<td>7K</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>46</td>
</tr>
<tr>
<td>aMetro</td>
<td>20K</td>
<td>54</td>
<td>18</td>
<td>36</td>
<td>18</td>
</tr>
<tr>
<td>K9Mail</td>
<td>40K</td>
<td>208</td>
<td>130</td>
<td>64</td>
<td>374</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>72K</strong></td>
<td><strong>311</strong></td>
<td><strong>172</strong></td>
<td><strong>115</strong></td>
<td><strong>1602</strong></td>
</tr>
</tbody>
</table>
## Is Thresher effective at filtering?

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Points-To Alarms</th>
<th>Thresher Refuted</th>
<th>True Bugs</th>
<th>Thresher Time (s)</th>
<th>False Alarm %</th>
</tr>
</thead>
<tbody>
<tr>
<td>PulsePoint</td>
<td>unknown</td>
<td>16</td>
<td>8</td>
<td>8</td>
<td>95</td>
<td>0</td>
</tr>
<tr>
<td>StandupTimer</td>
<td>2K</td>
<td>25</td>
<td>15</td>
<td>0</td>
<td>1068</td>
<td>100</td>
</tr>
<tr>
<td>DroidLife</td>
<td>3K</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SMSPopUp</td>
<td>7K</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>46</td>
<td>0</td>
</tr>
<tr>
<td>aMetro</td>
<td>20K</td>
<td>54</td>
<td>18</td>
<td>36</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>K9Mail</td>
<td>40K</td>
<td>208</td>
<td>130</td>
<td>64</td>
<td>374</td>
<td>18</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>72K</strong></td>
<td><strong>311</strong></td>
<td><strong>172</strong></td>
<td><strong>115</strong></td>
<td><strong>1602</strong></td>
<td><strong>17</strong></td>
</tr>
</tbody>
</table>
Is Thresher effective at filtering?

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Points-To Alarms</th>
<th>Thresher Refuted</th>
<th>True Bugs</th>
<th>Thresher Time (s)</th>
<th>False Alarm %</th>
<th>Filtered %</th>
</tr>
</thead>
<tbody>
<tr>
<td>PulsePoint</td>
<td>unknown</td>
<td>16</td>
<td>8</td>
<td>8</td>
<td>95</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>StandupTimer</td>
<td>2K</td>
<td>25</td>
<td>15</td>
<td>0</td>
<td>1068</td>
<td>100</td>
<td>60</td>
</tr>
<tr>
<td>DroidLife</td>
<td>3K</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>SMSPopUp</td>
<td>7K</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>46</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>aMetro</td>
<td>20K</td>
<td>54</td>
<td>18</td>
<td>36</td>
<td>18</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>K9Mail</td>
<td>40K</td>
<td>208</td>
<td>130</td>
<td>64</td>
<td>374</td>
<td>18</td>
<td>90</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>72K</strong></td>
<td><strong>311</strong></td>
<td><strong>172</strong></td>
<td><strong>115</strong></td>
<td><strong>1602</strong></td>
<td><strong>17</strong></td>
<td><strong>88</strong></td>
</tr>
</tbody>
</table>
Is Thresher effective at filtering?

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Points-To Alarms</th>
<th>Thresher Refuted</th>
<th>True Bugs</th>
<th>Thresher Time (s)</th>
<th>False Alarm %</th>
<th>Filtered %</th>
</tr>
</thead>
<tbody>
<tr>
<td>PulsePoint</td>
<td>unknown</td>
<td>16</td>
<td>8</td>
<td>8</td>
<td>95</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>StandupTimer</td>
<td>2K</td>
<td>25</td>
<td>15</td>
<td>0</td>
<td>1068</td>
<td>100</td>
<td>60</td>
</tr>
<tr>
<td>DroidLife</td>
<td>3K</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>SMSPopUp</td>
<td>7K</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>46</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>aMetro</td>
<td>20K</td>
<td>54</td>
<td>18</td>
<td>36</td>
<td>18</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>K9Mail</td>
<td>40K</td>
<td>208</td>
<td>130</td>
<td>64</td>
<td>374</td>
<td>18</td>
<td>90</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>72K</strong></td>
<td><strong>311</strong></td>
<td><strong>172</strong></td>
<td><strong>115</strong></td>
<td><strong>1602</strong></td>
<td><strong>17</strong></td>
<td><strong>88</strong></td>
</tr>
</tbody>
</table>

False alarms down to 17% from 63% (points-to analysis only)
Thresher filters 88% of false alarms from points-to analysis
False alarms down to 17% from 63% (points-to analysis only)

Thresher filters 88% of false alarms from points-to analysis

Guesstimate
Triage “well” without versus with: ~450 hours versus ~30 hours
Triage “ok” without: ~30 hours

<table>
<thead>
<tr>
<th></th>
<th>Alarms</th>
<th>Related</th>
<th>Bugs</th>
<th>Time (s)</th>
<th>Alarm %</th>
</tr>
</thead>
<tbody>
<tr>
<td>PulsePoint</td>
<td>unknown</td>
<td>16</td>
<td>8</td>
<td>8</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>StandupTimer</td>
<td>2K</td>
<td>25</td>
<td>15</td>
<td>0</td>
<td>1068</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>DroidLife</td>
<td>3K</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>SMSPopUp</td>
<td>7K</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>aMetro</td>
<td>20K</td>
<td>54</td>
<td>18</td>
<td>36</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>K9Mail</td>
<td>40K</td>
<td>208</td>
<td>130</td>
<td>64</td>
<td>374</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>90</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>72K</strong></td>
<td><strong>311</strong></td>
<td><strong>172</strong></td>
<td><strong>115</strong></td>
<td><strong>1602</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>17</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>88</strong></td>
</tr>
</tbody>
</table>
... in the process of finding leaks in apps
class HashMap {
    static Object[] EMPTY = new Object[2]; ...
    HashMap() { this.tbl = EMPTY; capacity initially empty }

    void put(Object key, Object val) {
        if (need capacity) {
            this.tbl = new Object[more capacity];
            copy from old table
        }
        this.tbl[bucket using hash of key] = val;
    }

    HashMap(Map m) {
        if (m.size() < 1) { this.tbl = EMPTY; }
        else { this.tbl = new Object[at least m.size()]; }  
        copy from m
    }
}
class HashMap {
    static Object[] EMPTY = new Object[2]; ... 
    HashMap() { this.tbl = EMPTY; capacity initially empty }

    void put(Object key, Object val) {
        if (need capacity) {
            this.tbl = new Object[more capacity];
            copy from old table
        }
        this.tbl[bucket using hash of key] = val;
    }

    HashMap(Map m) { 
        if (m.size() < 1) { this.tbl = EMPTY; }
        else { this.tbl = new Object[at least m.size()]; } 
        copy from m
    }
}
class HashMap {
    static Object[] EMPTY = new Object[2]; ...
    HashMap() { this.tbl = EMPTY; capacity initially empty }
    void put(Object key, Object val) {
        if (need capacity) {
            this.tbl = new Object[more capacity];
            copy from old table
        }
        this.tbl[bucket using hash of key] = val;
    }
    HashMap(Map m) {
        if (m.size() < 1) { this.tbl = EMPTY; }
        else { this.tbl = new Object[at least m.size()]; } 
        copy from m
    }
}
class HashMap {
    static Object[] EMPTY = new Object[2]; ...
    HashMap() { this.tbl = EMPTY; }  
    capacity initially empty

    void put(Object key, Object val) {
        if (need capacity) {
            this.tbl = new Object[more capacity];
            copy from old table
        }
        this.tbl[bucket using hash of key] = val;
    }

    HashMap(Map m) {
        if (m.size() < 1) { this.tbl = EMPTY; }
        else { this.tbl = new Object[at least m.size()]; }
        copy from m
    }
}
class HashMap {
    static Object[] EMPTY = new Object[2]; ...
    HashMap() {
        this.tbl = EMPTY;  \textit{capacity initially empty}
    }
    void put(Object key, Object val) {
        if (need capacity) {
            this.tbl = new Object[more capacity];
            copy from old table
        }
        this.tbl[bucket using hash of key] = val;
    }
    HashMap(Map m) {
        if (m.size() < 1) {
            this.tbl = EMPTY;
        } else {
            this.tbl = new Object[at least m.size()];
            copy from m
        }
    }
}
Find the Android’s HashMap bug ... 

```java
class HashMap {
    static Object[] EMPTY = new Object[2]; ...

    HashMap() {
        this.tbl = EMPTY; // capacity initially empty
    }

    void put(Object key, Object val) {
        if (need capacity) {
            this.tbl = new Object[more capacity];
            copy from old table
        }
        this.tbl[bucket using hash of key] = val;
    }

    HashMap(Map m) {
        if (m.size() < 1) { this.tbl = EMPTY; }
        else { this.tbl = new Object[at least m.size()]; }
        copy from m
    }
}
```

null object pattern: should not be written to

allocate new backing array on first write

An “evil” implementation of the Map interface can corrupt EMPTY. Then, all HashMaps created in the future will be corrupted.
Find the Android’s HashMap bug ...

```java
class HashMap {
    static Object[] EMPTY = new Object[2]; ...
    HashMap() { this.tbl = EMPTY; } // capacity initially empty

    void put(Object key, Object val) {
        if (need capacity) {
            this.tbl = new Object[more capacity];
            copy from old table
        }
        this.tbl[bucket using hash of key] = val;
    }

    HashMap(Map m) {
        if (m.size() < 1) { this.tbl = EMPTY; }
        else { this.tbl = new Object[at least m.size()]; } // copy from m
    }
}
```

null object pattern: should not be written to
allocate new backing array on first write
return 0
return “evil” content

An “evil” implementation of the Map interface can corrupt EMPTY. Then, all HashMaps created in the future will be corrupted.
class HashMap {
    static Object[] EMPTY = new Object[2];

    HashMap() {
        this.tbl = EMPTY;
    }
    void put(Object key, Object val) {
        if (need capacity) {
            this.tbl = new Object[more capacity];
            copy from old table
        }
        this.tbl[bucket using hash of key] = val;
    }
    HashMap(Map m) {
        if (m.size() < 1) {
            this.tbl = EMPTY;
        } else {
            this.tbl = new Object[at least m.size()];
        }
        copy from m
    }
}

HashMap(Map m) {
    if (m.size() < 1) {
        this.tbl = EMPTY;
    } else {
        this.tbl = new Object[at least m.size()];
    }
    copy from m
}

return 0

An “evil” implementation of the Map interface can corrupt EMPTY. Then, all HashMaps created in the future will be corrupted.

What if you store passwords in a HashMap?
class HashMap {
    static Object[] EMPTY = new Object[2];
    Object[] tbl = EMPTY;
    int capacity = 1;
    ...

    HashMap() {
        this.tbl = EMPTY;
    }

    void put(Object key, Object val) {
        if (need capacity) {
            this.tbl = new Object[more capacity];
        }
        this.tbl[bucket using hash of key] = val;
    }

    HashMap(Map m) {
        if (m.size() < 1) { this.tbl = EMPTY; }
        else { this.tbl = new Object[at least m.size()]; }
        copy from m
    }
}

return 0

return "evil" content

What if you store passwords in a HashMap?

We reported this, Google fixed it
https://android-review.googlesource.com/#/c/52183/

An "evil" implementation of the Map interface can corrupt EMPTY. Then, all HashMaps created in the future will be corrupted.
Contribution: Addressed the false alarm problem with a "smart and precise filter" and a goal-directed refutation analysis.
Another more challenging reason for ...
Another more challenging reason for ...

Roughly, 3% of all commits fix NullPointerExceptions
Callback-oriented programming ...
Callback-oriented programming ...
Callback-oriented programming ...

callbacks (e.g., Activity.onCreate)
Callback-oriented programming ...

callbacks (e.g., Activity.onCreate)
Callback-oriented programming ...

Android components have an ordered, event-driven lifecycle

callbacks (e.g., Activity.onCreate)
Callback-oriented programming ...

Android components have an ordered, event-driven lifecycle

Android Framework

callbacks (e.g., Activity.onCreate)

App

Activity

onCreate

onResume

onClick

onPause

onDestroy

mHostDb = null;
mService = null;
Callback-oriented programming ...

Android components have an ordered, event-driven lifecycle.

Android Framework

callbacks (e.g., Activity.onCreate)

But, lifecycles of different components and other callbacks can interleave ...

mHostDb = null;
mService = null;
Challenge: Verifying safety (of dereferences) depends on callback interleaving.

Android components have an ordered, event-driven lifecycle.

But, lifecycles of different components and other callbacks can interleave ...

safe?

mHostDb.s();

onResume

onClick

onPause

onDestroy

mHostDb = null;

mService = null;
... over a shared, global heap
... over a shared, global heap

```
onCreate
  └── onResume
      ├── onClick
      │    └── onPause
      │         └── onDestroy
  └── onDestroy

onCreate
  └── onResume
      ├── onClick
      │    └── onPause
      │         └── onDestroy
  └── onDestroy
...```
... over a shared, global heap
... over a shared, global heap

Challenge: Verifying safety (of dereferences) depends ordering of heap writes
But verifying safety shouldn’t be so hard ...
But verifying safety shouldn’t be so hard ...

```
    mHostDb.s();

    onCreate
      ↓
    onResume
      ↓
    onClick
      ↓
    onPause
      ↓
    onDestroy
```
But verifying safety shouldn’t be so hard ...

```java
mHostDb.s();
```

```
 safe?
```

```
onCreate
```

```
onResume
```

```
onClick
```

```
onPause
```

```
onDestroy
```
But verifying safety shouldn’t be so hard ...

Idea: Safety of a particular dereference should not require reasoning about all callback interleavings

```java
mHostDb.s();
```
But verifying safety shouldn’t be so hard ...

Idea: Safety of a particular dereference should not require reasoning about all callback interleavings.

A “smart” goal-directed analysis could consider relevant callback orderings without considering all of them—jumping analysis.
Evaluation: Proving dereferences safe
Evaluation: Proving dereferences safe

10 open source Android apps
3,000 to 55,000 lines of code
10 to 100 components
120 to 1,320 callbacks
Evaluation: Proving dereferences safe

10 open source Android apps

3,000 to 55,000 lines of code

10 to 100 components

120 to 1,320 callbacks

Event product graph would have $10^{10}$ to $10^{111}$ nodes (with one instance per class)
Evaluation: Proving dereferences safe

10 open source Android apps

3,000 to 55,000 lines of code

10 to 100 components

120 to 1,320 callbacks
Evaluation: Proving dereferences safe

10 open source Android apps
3,000 to 55,000 lines of code
10 to 100 components
120 to 1,320 callbacks

Compared 3 analyses

Nit: type inference
Thresher: no jumping
Hopper: jumping
Is Hopper effective at proving safety?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290 (52)</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625 (37)</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74 (77)</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330 (53)</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988 (36)</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
<td><strong>5412 (54)</strong></td>
</tr>
</tbody>
</table>
Is Hopper effective at proving safety?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290 (52)</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625 (37)</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74 (77)</td>
</tr>
<tr>
<td>Tor</td>
<td>53</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330 (53)</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>266</td>
<td>62562</td>
<td>41071</td>
<td>9968</td>
<td>5412 (54)</td>
</tr>
</tbody>
</table>

Many derefs
Is Hopper effective at proving safety?

<table>
<thead>
<tr>
<th>Project</th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290 (52)</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625 (37)</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74 (77)</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330 (53)</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988 (36)</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
<td><strong>5412 (54)</strong></td>
</tr>
</tbody>
</table>
### Is Hopper effective at proving safety?

<table>
<thead>
<tr>
<th>Type inference</th>
<th>No jumping</th>
<th>Jumping</th>
</tr>
</thead>
<tbody>
<tr>
<td>unproven derefs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software</th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290 (52)</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625 (37)</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74 (77)</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330 (53)</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988 (36)</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
<td><strong>5412 (54)</strong></td>
</tr>
</tbody>
</table>
Is Hopper effective at proving safety?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290 (52)</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625 (37)</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74 (77)</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330 (53)</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988 (36)</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
<td><strong>5412 (54)</strong></td>
</tr>
</tbody>
</table>
Is Hopper effective at proving safety?

### unproven derefs

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290 (52)</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625 (37)</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74 (77)</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330 (53)</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988 (36)</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
<td><strong>5412 (54)</strong></td>
</tr>
</tbody>
</table>

**Thr and Hop: 10 second budget per deref**
Is Hopper effective at proving safety?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290 (52)</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625 (37)</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74 (77)</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330 (53)</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988 (36)</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
<td><strong>5412 (54)</strong></td>
</tr>
</tbody>
</table>
### unproven derefs

<table>
<thead>
<tr>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
</tr>
</tbody>
</table>

**Thresher has 74% fewer than Nit**
Is Hopper effective at proving safety?

<table>
<thead>
<tr>
<th>Software</th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290 (52)</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625 (37)</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74 (77)</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330 (53)</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988 (36)</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
<td><strong>5412 (54)</strong></td>
</tr>
</tbody>
</table>

Hopper has 54% fewer unproven derefs than Thresher.
Is Hopper effective at proving safety?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
<th>% Proven</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
<td>92</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
<td>94</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
<td>93</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
<td>90</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290 (52)</td>
<td>92</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625 (37)</td>
<td>92</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74 (77)</td>
<td>97</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330 (53)</td>
<td>94</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988 (36)</td>
<td>90</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
<td>91</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
<td><strong>5412 (54)</strong></td>
<td><strong>92</strong></td>
</tr>
</tbody>
</table>
Is Hopper effective at proving safety?

<table>
<thead>
<tr>
<th>KLOC</th>
<th>Derf</th>
<th>Nit</th>
<th>Thr Hop (Impr %)</th>
<th>% Proven</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
</tr>
</tbody>
</table>

Compare with state-of-the-art NPE checking work that reports 84-91% proven on normal Java programs!
Remaining manual triage ...
Triaged 200 alarms (from Hopper), 189 false
Remaining manual triage ...

Triaged 200 alarms (from Hopper), 189 false

Reasons: insufficient Android modeling, imprecise container and string domains
Remaining manual triage ...

Triaged 200 alarms (from Hopper), 189 false

Reasons: insufficient Android modeling, imprecise container and string domains

Only 17 false alarms due to timeouts
Remaining manual triage ...

Triaged 200 alarms (from Hopper), 189 false

Reasons: insufficient Android modeling, imprecise container and string domains

Only 17 false alarms due to timeouts

Found 11 bugs in 4 apps (lastfm, seriesguide, connectbot, wordpress)

5 bugs due to bad ordering assumptions

10/11 patches accepted
Remaining manual triage ...

Triaged 200 alarms (from Hopper), 189 false
Reasons: insufficient Android modeling, imprecise container and string domains
Only 17 false alarms due to timeouts

Found 11 bugs in 4 apps (lastfm, seriesguide, connectbot, wordpress)
5 bugs due to bad ordering assumptions
10/11 patches accepted
Research Agenda: The cooperative approach addresses the whole bug mitigation process.

**Fissile Types:** Checking Almost Everywhere Invariants

[Coughlin+ POPL’14]

**Divva:** Synthesizing Short-Circuiting Data Structure Checks

[in prep]

**Jsana:** Abstract Domain Combinators for Dynamic Languages

[Cox+ ECOOP’13, Cox+ SAS’14, Cox+ ESOP’15]

**Thresher/Hopper:** Goal-Directed Refutation Analysis

[Blackshear+ SAS’11, Blackshear+ PLDI’13, Blackshear+ OOPSLA’15]

**Divva:** Synthesizing Short-Circuiting Data Structure Checks

[in prep]

**Auditr:** Securing Against “Inputs of Coma and Exposure”

[in progress]

**Fixr:** Mining Bug Fixes from Commits

[in progress]
Research Agenda: The cooperative approach addresses the whole bug mitigation process.

- **Verifier**
  - ✔ proof of no bug
  - ✔

- **Alarm**
  - ✔
  - ✘

- **Program**
  - Program-
  - ming

- **Test**
  - Test 
  - Input

- **Manual**
  - Manual-
  - Triaging

- **Spec-ification**
  - Spec-
  - ification

- **Run-
er**

- **Divva**
  - Divva: Synthesizing Short-
  - Circuiting Data Structure 
  - Checks 
  - [in prep]

- **Jsana**
  - Jsana: Abstract Domain Combinators 
  - for Dynamic Languages 
  - [Coughlin+ POPL’14]

- **Github**

- **Thresher/Hopper**
  - Thresher/Hopper: Goal-Directed 
  - Refutation Analysis 
  - [Blackshear+ SAS’11, Blackshear+ PLDI’13, 
    Blackshear+ OOPSLA’15]

- **Fissile Types**
  - Checking Almost 
  - Everywhere 
  - Invariants 
  - [Coughlin+ POPL’14]

- **Manual**
  - Manual-
  - Triaging

- **Fixr**
  - Fixr: Mining Bug 
  - Fixes from Commits 
  - [in progress]

- **Auditr**
  - Auditr: Securing Against “Inputs of 
  - Coma and Exposure” 
  - [in progress]
Research Agenda: The cooperative approach addresses the whole bug mitigation process.

Fissile Types: Checking Almost Everywhere Invariants [Coughlin+ POPL’14]

Divva: Synthesizing Short-Circuiting Data Structure Checks [in prep]

Jsana: Abstract Domain Combinators for Dynamic Languages [Cox+ ECOOP’13, Cox+ SAS’14, Cox+ ESOP’15]

Thresher/Hopper: Goal-Directed Refutation Analysis [Blackshear+ SAS’11, Blackshear+ PLDI’13, Blackshear+ OOPSLA’15]

Divva: Synthesizing Short-Circuiting Data Structure Checks [in prep]

Manual Triaging

Thresher/Hopper: Goal-Directed Refutation Analysis [Blackshear+ SAS’11, Blackshear+ PLDI’13, Blackshear+ OOPSLA’15]

Jsana: Abstract Domain Combinators for Dynamic Languages [Cox+ ECOOP’13, Cox+ SAS’14, Cox+ ESOP’15]

Fissile Types: Checking Almost Everywhere Invariants [Coughlin+ POPL’14]

Github

Fixr: Mining Bug Fixes from Commits [in progress]

Auditr: Securing Against “Inputs of Coma and Exposure” [in progress]
Checking Reflection with Almost Everywhere Invariants
Method Reflection and the Great Divide
Method Reflection and the Great Divide

object[string]()
Method Reflection and the Great Divide

reflective method call: dispatch based on **run-time value** (in `string`)

```
object[string]()
```
Method Reflection and the Great Divide

reflective method call: dispatch based on run-time value (in string)

object[string]()

type system designers

"web 2.0" developers
Method Reflection and the Great Divide

reflective method call: dispatch based on run-time value (in string)

object[string]()

type system designers

“web 2.0” developers

Type system designers worry.

What gets called? What if object has no method named by string?
Method Reflection and the Great Divide

Reflective method call: dispatch based on run-time value (in string)

object[string]()

type system designers

"web 2.0" developers

Type system designers worry.

What gets called? What if object has no method named by string?

"Web 2.0" developers think it's cool.

I can flexible and compact code, so I will take it over static safety.
Method Reflection and the Great Divide

reflective method call: dispatch based on run-time value (in string)

object[string]()

type system designers

"web 2.0" developers

Type system designers worry.

When do we call object[string]()

by string?

"Web 2.0" developers think it’s cool.

"MethodNotFound" checked at run time

static safety.
Programs are often (1) safe, (2) not type safe, (3) but almost so
Programs are often (1) safe, (2) not type safe, (3) but almost so.
Programs are often (1) safe, (2) not type safe, (3) but almost so

callback.o[callback.m]()
Programs are often (1) safe, (2) not type safe, (3) but almost so

```c
callback.o[callback.m]()
```

safe assuming a relationship invariant between .o and .m
Programs are often (1) safe, (2) not type safe, (3) but almost so

Program

invariant holds

safe assuming a relationship invariant between .o and .m

callback.o[callback.m]()
Programs are often (1) safe, (2) not type safe, (3) but almost so

callback.o[callback.m]()

safe assuming a relationship invariant between .o and .m

Program

Invariant holds

Invariant broken
Programs are often (1) safe, (2) not type safe, (3) but almost so

\[ \text{callback.o[callback.m]}() \]

safe assuming a relationship invariant between .o and .m

invariant holds

invariant broken

but only temporarily

Program
Programs are often (1) safe, (2) not type safe, (3) but almost so

Tolerate “temporary” violation with

\[ \text{callback.o}[\text{callback.m}]() \]

Program

invariant holds

safe assuming a relationship invariant between .o and .m

invariant broken

but only temporarily
Is Fissile effective at proving reflective call safety?

Fissile analyzes **Objective-C** source

9 benchmarks (6 libraries + 3 apps)

1,000 to 176,000 lines of code

461,000 lines in total

Type annotations

seeded with 76 reflection checks in system libraries

needed only 136 annotations in benchmarks (total)
Is Fissile effective at proving reflective call safety?

Fissile analyzes **Objective-C**

9 benchmarks (6 libraries + 3 apps)

1,000 to 176,000 lines of code

461,000 lines in total

Type annotations

seeded with 76 reflection checks in system libraries

needed only 136 annotations in benchmarks (total)
Is Fissile effective at proving reflective call safety?

Fissile analyzes Objective-C

9 benchmarks (6 libraries + 3 apps)
1,000 to 176,000 lines of code
461,000 lines in total

Type annotations

Proved 86% of check sites (up from 76%) at interactive speeds (~4 to 90 kloc/s)

Including Skim, Adium, and OmniGraffle
Is Fissile effective at proving reflective call safety?

Fissile analyzes **Objective-C**

9 benchmarks (6 libraries + 3 apps)

1,000 to 176,000 lines of code

461,000 lines in total

**Proved 86% of check sites (up from 76%) at interactive speeds (~4 to 90 kloc/s)**

**Including** Skim, Adium, and OmniGraffle
Is Fissile effective at proving reflective call safety?

Fissile analyzes **Objective-C**

- 9 benchmarks (6 libraries + 3 apps)
- 1,000 to 176,000 lines of code
- 461,000 lines in total

Type annotations seeded with 76 reflection checks in system libraries needed only 136 annotations in benchmarks (total)

Including Skim, Adium, and OmniGraffle

Proved **86%** of check sites (up from 76%) at interactive speeds (~4 to 90 kloc/s)

Big Deal: makes IDE integration possible
Summary: The cooperative approach addresses the whole bug mitigation process.

- **Fissile Types**: Checking Almost Everywhere Invariants [Coughlin+ POPL’14]
- **Divva**: Synthesizing Short-Circuiting Data Structure Checks [in prep]
- **Jsana**: Abstract Domain Combinators for Dynamic Languages [Cox+ ECOOP’13, Cox+ SAS’14, Cox+ ESOP’15]
- **Thresher/Hopper**: Goal-Directed Refutation Analysis [Blackshear+ SAS’11, Blackshear+ PLDI’13, Blackshear+ OOPSLA’15]
- **Divva**: Synthesizing Short-Circuiting Data Structure Checks [in prep]
- **Auditr**: Securing Against “Inputs of Coma and Exposure” [in progress]
- **Fixr**: Mining Bug Fixes from Commits [in progress]
Future Directions
Future Directions

The Cooperative Approach

Unit Test Synthesis: test input + test code to unequivocally show bug
Evidence for Alarms: alarm explanations, “probability” of bug
Hardening: synthesize efficient dynamic checks
Patch Synthesis: synthesize bug fixes
Input Debugging: bugs in input
Performance/Scalability/Security Bugs: beyond correctness bugs
Analysis Engines: in new software domains
Future Directions

The Cooperative Approach

Unit Test Synthesis: test input + test code to unequivocally show bug
Evidence for Alarms: alarm explanations, “probability” of bug
Hardening: synthesize efficient dynamic checks
Patch Synthesis: synthesize bug fixes
Input Debugging: bugs in input
Performance/Scalability/Security Bugs: beyond correctness bugs
Analysis Engines: in new software domains

Software Domains

Dynamically-Typed: web, science+engineering
Concurrent: event-driven systems, user-interactive systems, servers
Distributed: “big data” software
Verifier

Runnner

Manual Triaging

Alarm Report

Program Triaging

Test Input

Specification

Program

Github

Test Output

proof of no bug

www.cs.colorado.edu/~bec
plv.colorado.edu

www.plv.colorado.edu