

Improving Android Reliability and Security

Iulian Neamtiu, Assoc. Prof.

CS Chairs Meeting
June 14, 2018

Mobile OSes rapidly expanding their device range and user base....



ANDROID WEAR



PHONES



TABLETS



ANDROID TV

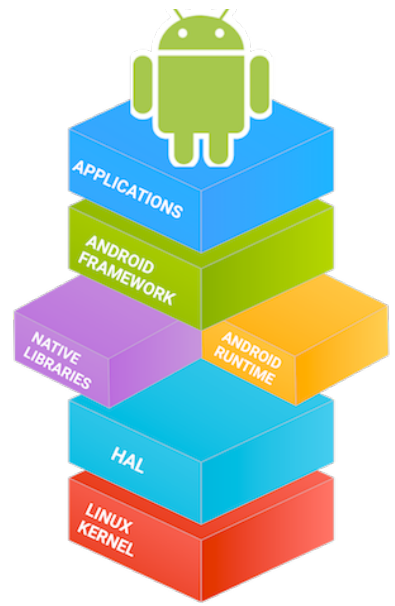


ANDROID AUTO

Android:
2 billion
monthly
active
users

... but users/developers/vendors have little insight/assurance of device behavior/reliability/security

Core SW stack (AOSP)



Preinstalled software



38 popular Android devices ship with malware already installed

CNN tech BUSINESS CULTURE GADGETS FUTURE STARTUPS

The FBI, CIA and NSA say American citizens shouldn't use Huawei phones

by Selena Larson @selenal Larson

ABC NEWS CONGRESS

Top intelligence official says Chinese ZTE cellphones pose security risk to U.S.

Third-party apps



- ASPLOS'18
- EASE'15
- ISSTA'16
- ISSRE'15

ACSAC'12

- OOPSLA'16
- ICSE'18
- ICSE'13
- MobiCom'12

The Nature of Mobile Bugs

Study: 22,000 confirmed&fixed mobile bugs (Android, iOS) over 7 years
Focus: High-severity bugs (crashes, data loss/corruption)

Android	iOS
Concurrency: 66%	Crash (non-concurrency): 52%
GUI: 23%	App. logic: 32%
Security: 5%	Build:12%

How to find/reproduce/fix these?

Concurrency Errors in Android

Harmful race in the Android OS!

Visible rows

ImageView#1
row 1

ImageView#2
row 2

ImageView#3
row 3

ImageView#4
row 4

row 5

Invisible rows

ImageView#1
row 6

ImageView#2
row 7

Main Thread

Background Thread

mr...@gmail.com <mr...@gmail.com> #23 Jan 11, 2015 08:59AM

I have around 700 crashes DAILY in production! This issue is not fixed and should be reopened. Come on guys! this is a really nasty bug that has been around for 3 months already!

va...@gmail.com <va...@gmail.com> #25 Jan 11, 2015 01:05PM

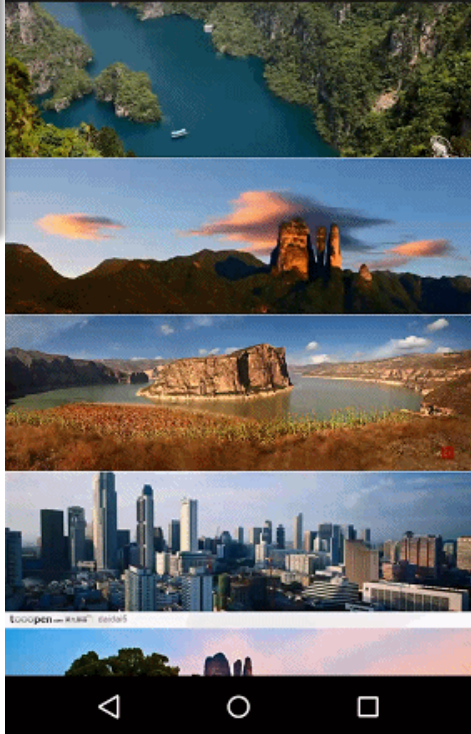
Unfortunately no, I cannot provide the code because it's really hard to reproduce this, but it is sill happens in com.android.support:recyclerview-v7:21.0.3 with stacktrace

ya...@gmail.com <ya...@gmail.com> #70 Aug 27, 2015 01:13AM

Hi,

This is very random, cannot reproduce every time.
Device: Nexus 6

ListViewTest



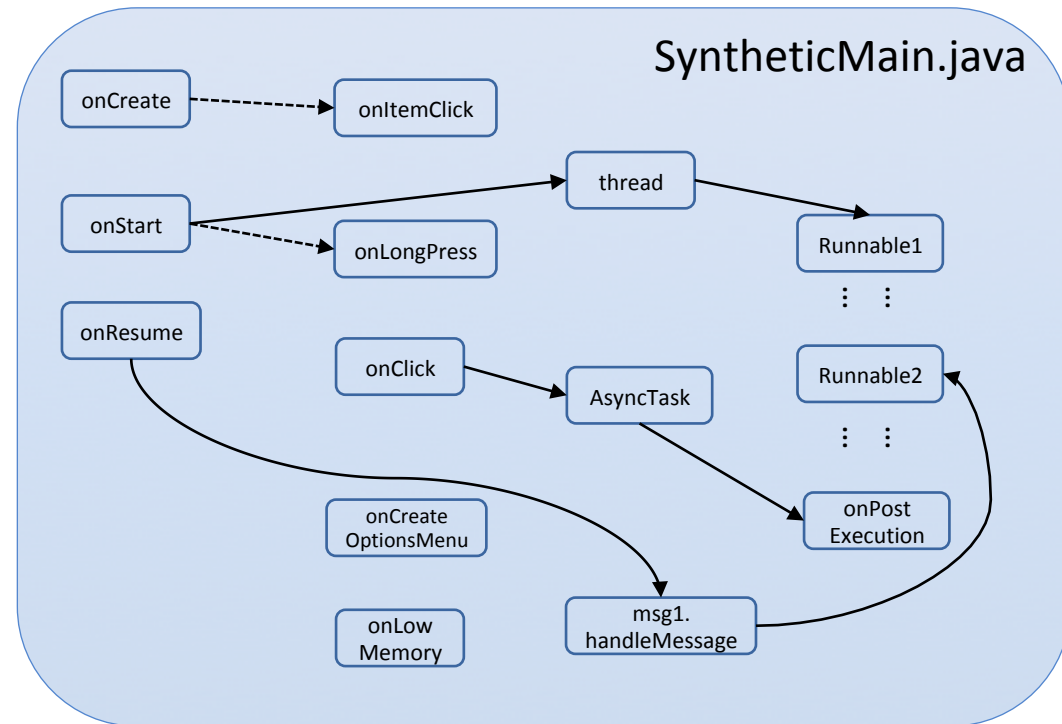
<https://code.google.com/p/android/issues/detail?id=77846>

Our Approach

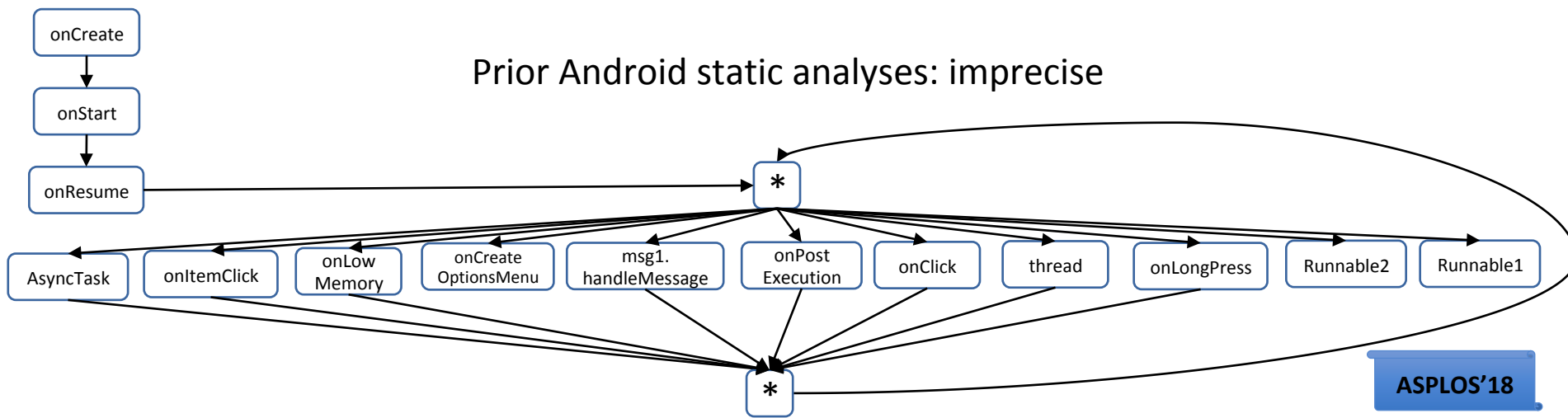
- Prior approaches: all dynamic
 - Low coverage, high rate of false negatives & false positives
 - Our approach
 - **Action** = context-sensitive event handling
 - Novel abstraction, reifies Callbacks, Threads, AsyncTasks, MsgHandler
 - **Happens-before** (HB): $A1 < A2$
 - A1 is completed before A2
 - **Bootstrapping** via program synthesis → precise static analysis
 - **Backward symbolic execution** to refute (most) false positives
- Effective and efficient

Program Synthesis to “Boot-strap” Precise Static Analysis

- Create **synthetic “main” activity**
- Create **synthetic method call sites**
 1. Add lifecycle/GUI/system actions
 2. Build call graph for found actions
 3. Go back to step 1, iterate until fixpoint



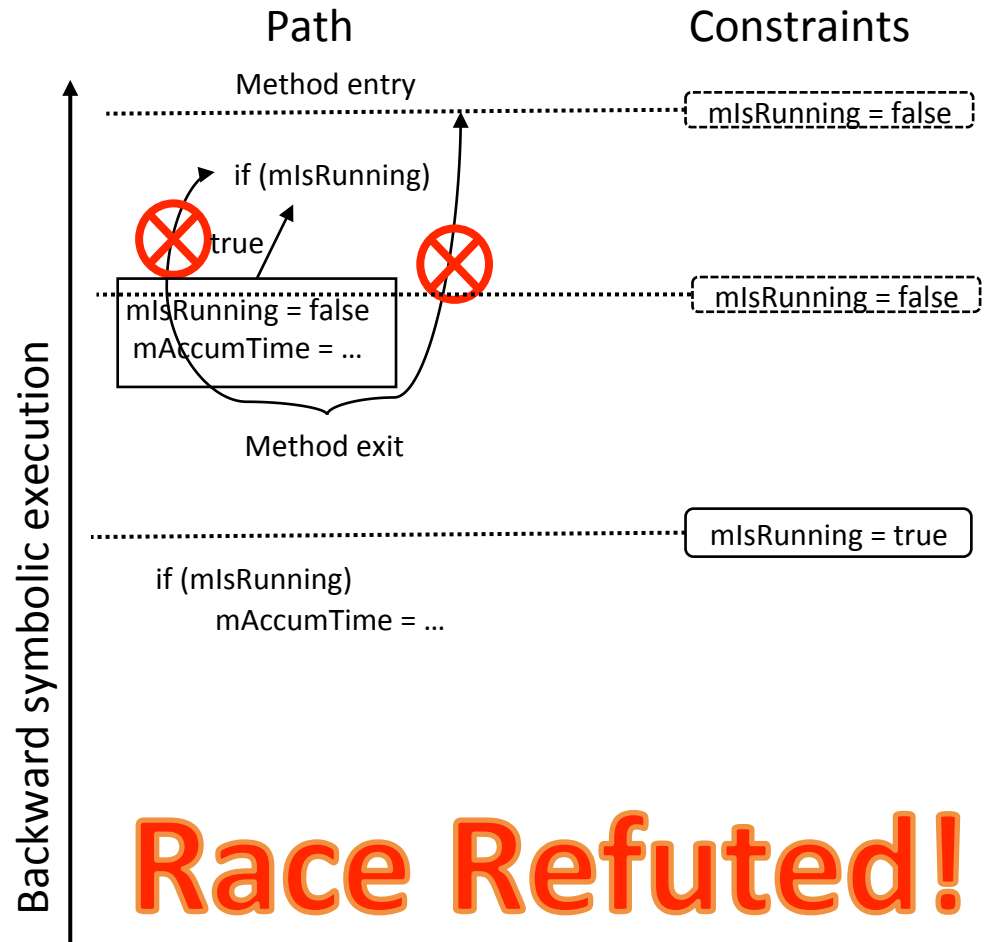
Prior Android static analyses: imprecise



Symbolic Execution-based Refutation

```
void stop(){// action B
  if (mIsRunning) {
    mIsRunning = false;
    mAccumTime=... //  $\alpha_B$ 
  }
}
```

```
Timer Runnable runner = {
void run() {//action A
  if (mIsRunning) {
    mAccumTime=... //  $\alpha_A$ 
    if (*) {
      ...
      postDelayed(runner,...);
    }
  }
  else
    mIsRunning=false;
}}
```








Race Refuted!

Can α_B occur before α_A ? **No!**

Evaluation

Effectiveness: races found

App		Installs (millions)	Candidate racy pairs	After our analysis	True races	False posi- tives	Event Racer
Barcode Scanner		> 100	64	15	11	4	7
VLC		> 100	202	35	32	3	0
FB Reader		> 10	836	106	93	13	5
K-9		> 5	1,347	89	72	17	1
NPR		> 1	607	21	21	0	3
Across 20 apps			431	33	29	4	4

13x

Best prior
work (dynamic)

Dataset: 194 open-source apps; 20 analyzed manually

Analysis time: about 30 minutes/apps

ASPLOS'18

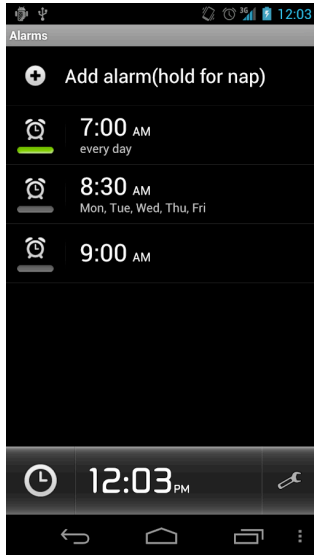
Skype		> 1,000	178
Snapchat		> 500	464
Netflix		> 100	793
Acrobat Reader		> 100	3,134
PayPal		> 50	51
Walmart		> 10	48

(paper in
preparation)

GUI bugs

GUI Bugs: Restart Errors

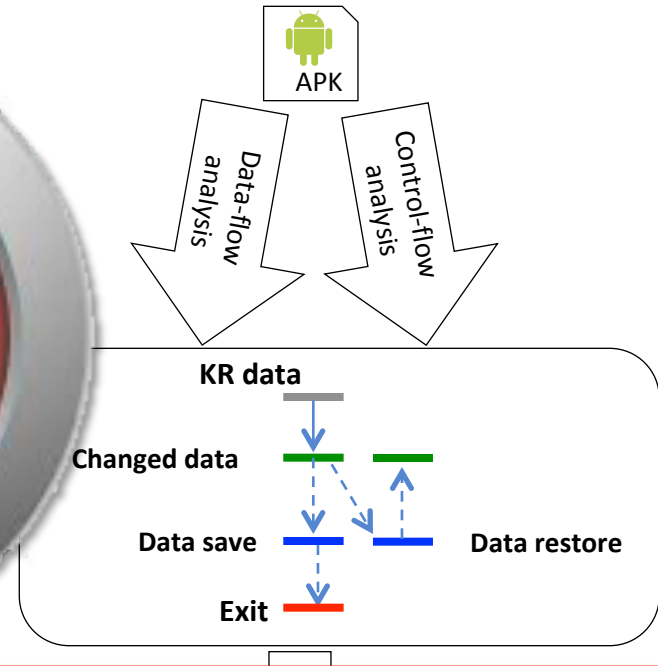
Earlier version of **Alarm Clock Plus** app,
more than 5 million installs



change n
alarm



or sen
then bri
alarm is rese



Restart is good: cleans the state

Restart is bad: cleans the state

Our approach exposes data kept/lost upon restart

Kill-a
Desk
Mob
An

Developers *must* do the rest; confusing and error-prone
Our approach: define KR hierarchy, static analysis to find KR errors

Research
Award 2015

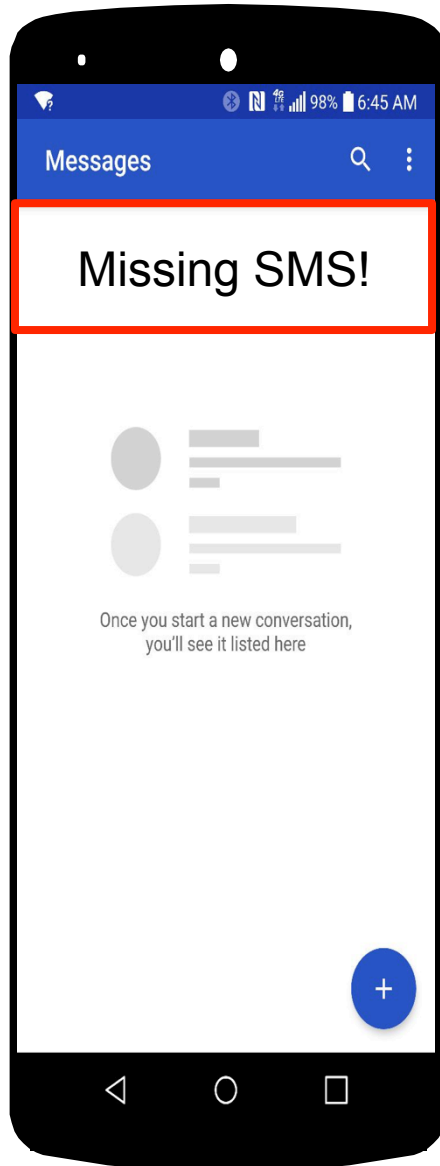
Results: **49 confirmed bugs in 37 apps**, including in well-known apps:
Dr.WebAnti-virusLight, Symantec Norton Snap, Motorola Camera,
Alarm Clock Plus, OI File Manager

OOPSLA'16

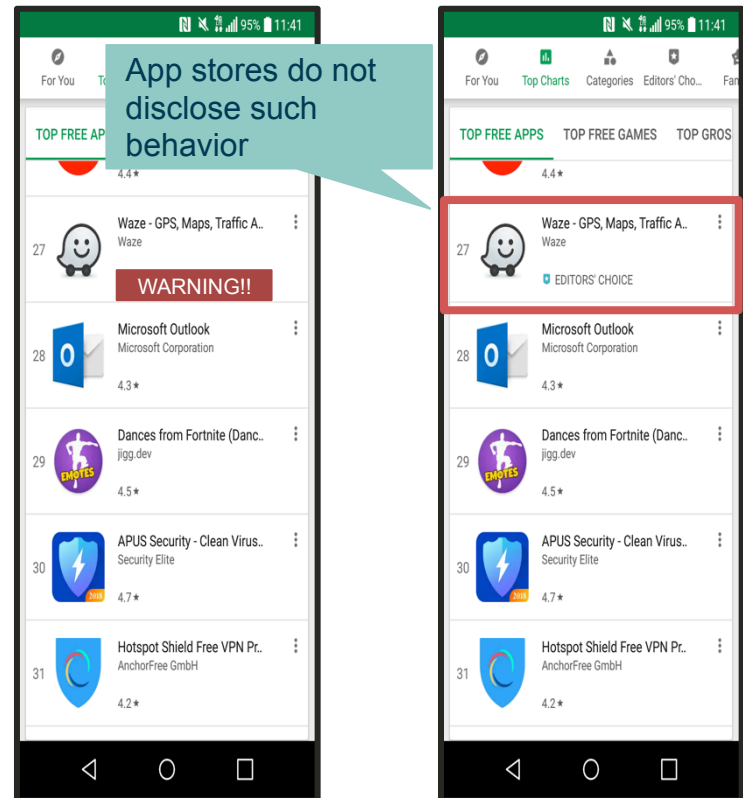
Security

'Self Hiding' Behavior (SHB)

A malicious app has deleted a user's messages without their knowledge or consent

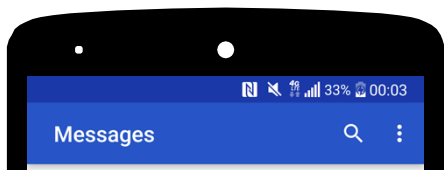


SHB = Behavior meant to hide an app's presence or actions ⇒ **Fundamentally Deceptive**
Our approach detects and characterizes this behavior



SHB#6: Delete Message

Malware **DroidKungFu1**
deletes SMS messages



```
Specialinvoke $r3.<java.lang.StringBuilder: void <init>(java.lang.String>("content://sms/")
$r4 = virtualinvoke $r3.<java.lang.StringBuilder: java.lang.String toString() >()
$r5 = staticinvoke <android.net.Uri: android.net.Uri parse(java.lang.String)>($r4)
virtualinvoke $r2.<android.net.Uri: void delete(android.net.Uri)>($r5)
```

Control &
data flow

Control &
data flow

Control &
data flow

Essentially, the SMS
messages self
destruct!

NOT user-initiated!

Results: “Good” Apps Behave Badly

Dataset: 3,219 malicious apps; 6,233 benign apps

Our static analysis can separate benign from malicious with 87.19% F-measure

Malware employs self-hiding (1.5 SH/app; unsurprising)

Some good apps employ self-hiding (0.2 SH/app)

HIDE RUNNING APP



Accesses and initializes location without user's knowledge

BLOCK CALL



Not a False Positive!

MUTE PHONE



Manipulates the ringer mode

HIDE NOTIFICATION / BLOCK MESSAGE



Blocks notifications without user consent

LURK/HOVER for a File Explorer?



Interposes between user and app