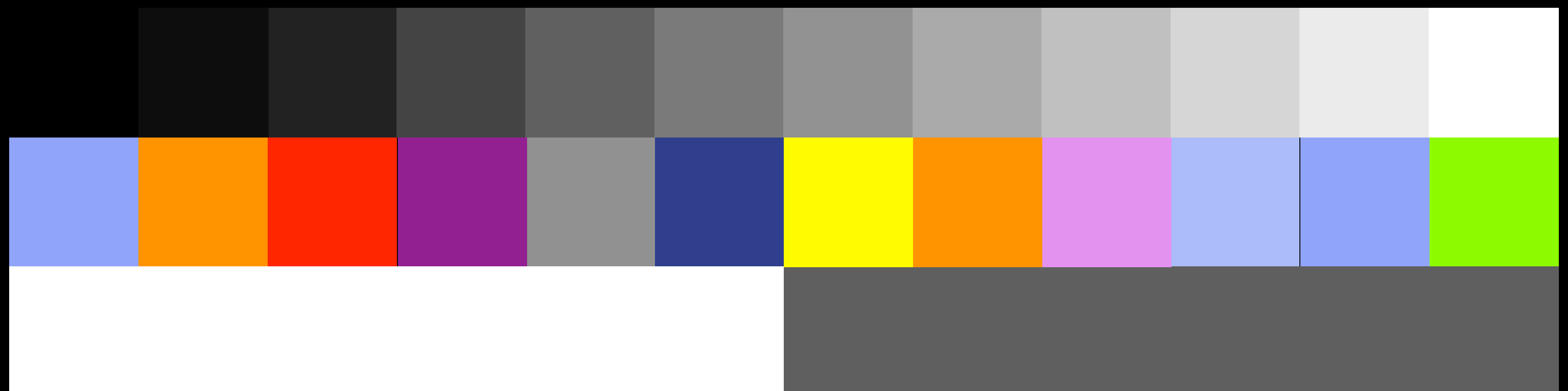
COMPUTER SECURITY LAB
RICE UNIVERSITY



Android Security: A Broad Overview & A Dive into the World of Advertising

Dan S. Wallach (Rice University)

Android development 101

Android apps are written in Java 6 (no lambdas, no streams)

Download and install Java SE, Android Studio (IntelliJ), and tons of Android SDKs

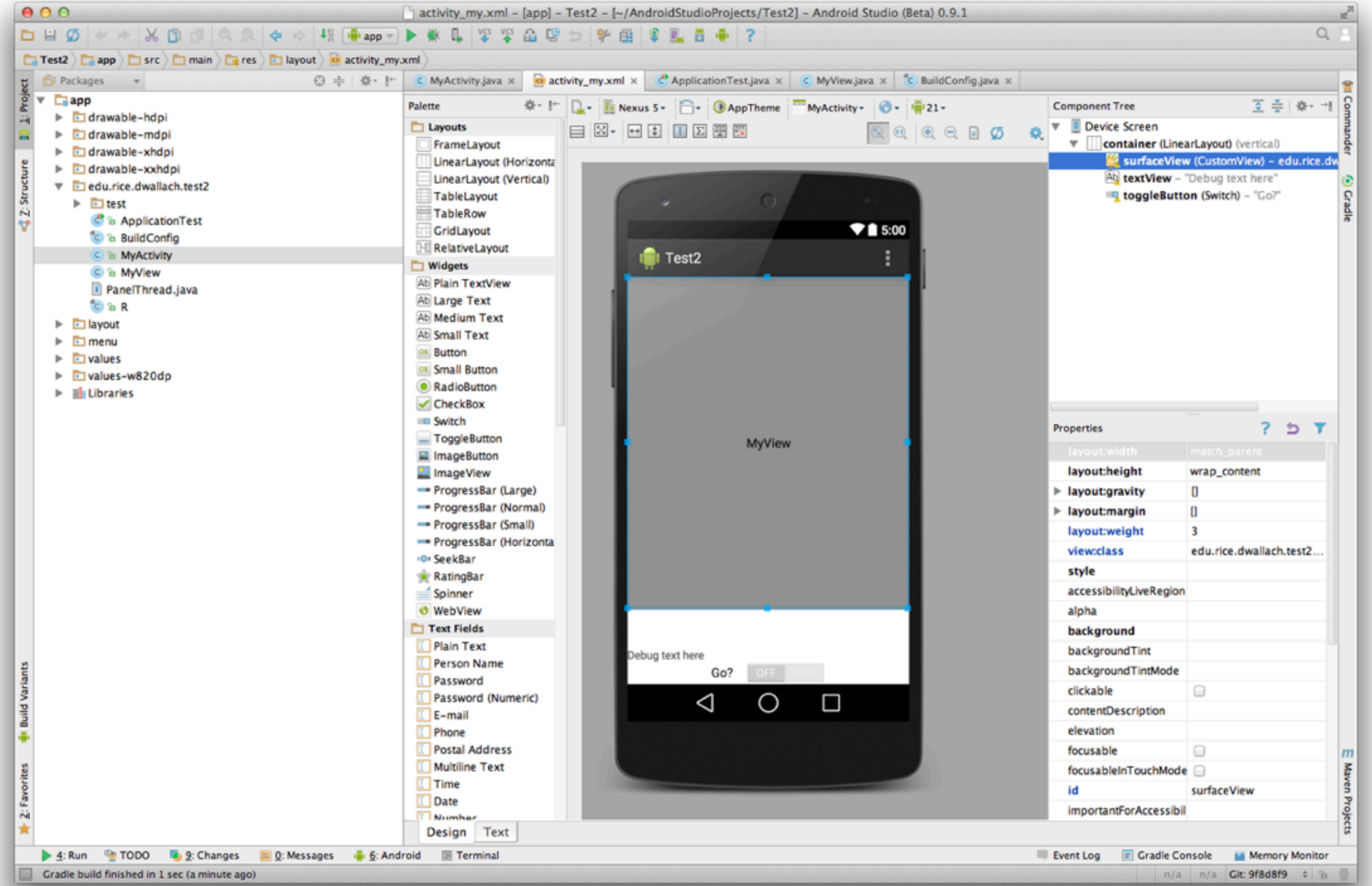
Start building a basic “Hello World” and work up from there

Android Studio generates lots of boilerplate

You lay out your UI with a graphical tool

Basic Java code to put it together is auto-generated

Android “emulator” is easy to use



activity_my.xml - [app] - Test2 - [~/AndroidStudioProjects/Test2] - Android Studio (Beta) 0.9.1

Component Tree

- Device Screen
  - container (LinearLayout) (vertical)
    - surfaceView (CustomView) - edu.rice.dw
    - textView - "Debug text here"
    - toggleButton (Switch) - "Go?"

Properties

layout:width	match_parent
layout:height	wrap_content
layout:gravity	[]
layout:margin	[]
layout:weight	3
view:class	edu.rice.dwallach.test2...
style	
accessibilityLiveRegion	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	☐
contentDescription	
elevation	
focusable	☐
focusableInTouchMode	☐
id	surfaceView
importantForAccessibil	

Gradle build finished in 1 sec (a minute ago)

Android Security Basics

Bugs = exploits:
How do you write bug-free code?

Android apps are written in Java

No buffer overflows, heap overflows, use-after-free, etc.

Most system services, APIs, etc. also written in Java

C is available (via JNI) but most apps don’t use it much

Maybe some use in games

Android process isolation

Each app runs with its own Unix user-id (based on digital signature)

Private local storage (akin to multi-user Unix)

We’re not depending on Java for isolation. This isn’t Java applets.

Apps must request “permissions” to do dangerous things.

Access the Internet, access local storage, access your photos, etc.

Some permissions are more interesting than others.

Access your microphone, be your SMS app, etc.

Evolving permissions UX

Prior to Android 6 (“Lollipop” and earlier):

All permission requests made in the app manifest (an XML file).

User sees permission dialog at install time. Take it or leave it.

Android 6 (“Marshmallow”):

Apps compiled against the new APIs must request permissions at runtime.

But every app now gets full Internet privileges without asking!

User can revoke permissions, even for older apps.

(CyanogenMod’s SecurityGuard will provide fake results, e.g., empty contacts list, while Android 6 will throw a security exception.)

SELinux / SEAndroid

Mandatory access controls built into the kernel

Not user-visible, mostly used by the system to lock itself down

So we’re good, right?

Apps are isolated from one another.

Permissions are approved/denied by users.

Relatively few vulnerabilities from common C bug patterns.

So we’re good, right? Nope.

Apps are isolated from one another.

Apps like to chat via IPC (“Binder” and “Intents”).

Opportunities for “confused deputy” attacks.

Permissions are approved/denied by users.

Dialog fatigue: users tend to say “yes” to anything.

Improvements in Android 6: asking at *time-of-use* instead of *time-of-install*.

Relatively few vulnerabilities from common C bug patterns.

Many libraries still implemented in C (media decoders, browser, etc.)

But can we auto-update around the problem?

The auto-update issue

Example: Android’s WebView widget (WebKit-based)

Like Chrome or any other browser, frequent updates are part of the security model.

Android 4.3 or earlier: WebView was baked into the system.

Android 4.4 and later: WebView is separately installed / updated from the Play Store.

“Google Play Services”: rolling big chunks of Android into an app

Security goodness: auto-updates from Google, new services on old platforms.

Big chunks of Android are no longer open source.

And Chinese Android phones aren’t connected to the Play Store at all.

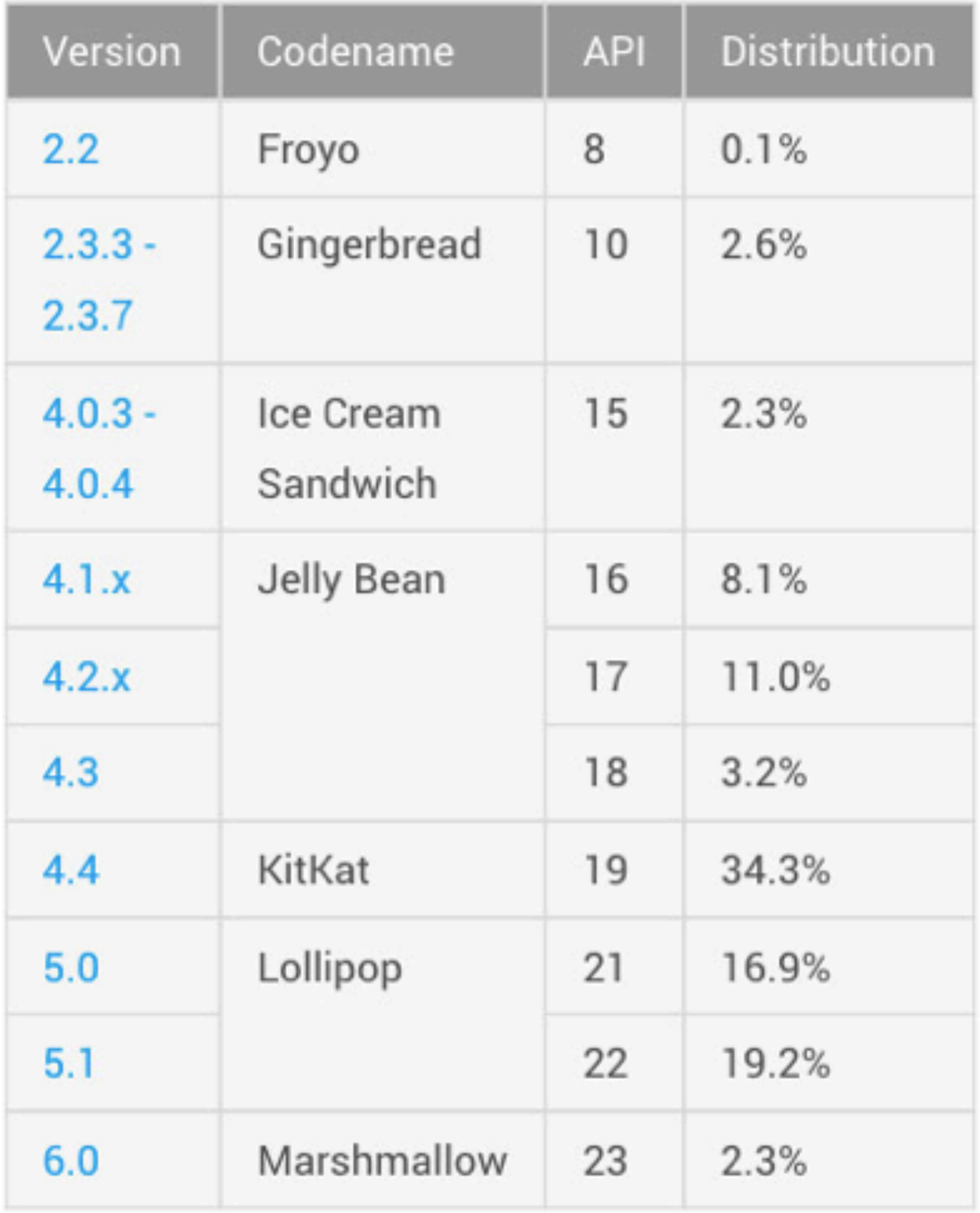
Latest news: Google and the OEMs are finally embarrassed about this.

Google is releasing monthly security updates for Nexus phones.

Other OEMs (hopefully) getting on board.

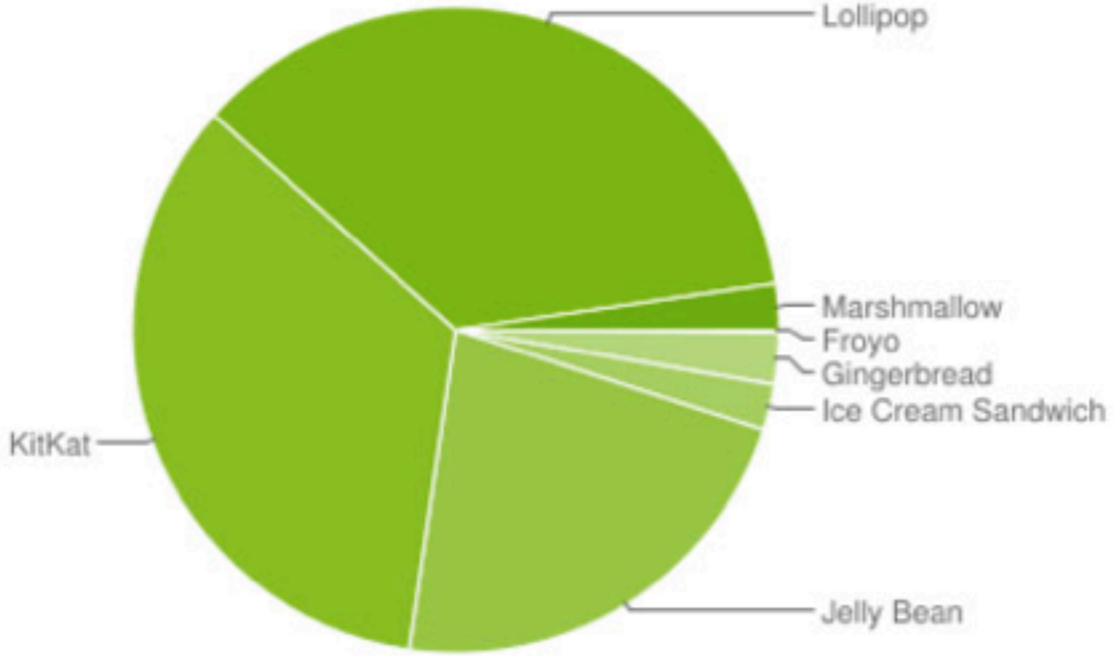
Android version distribution in the wild

Almost 73% are new enough to get WebView updates. Good, but not enough.



Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.3%
4.1.x	Jelly Bean	16	8.1%
4.2.x		17	11.0%
4.3		18	3.2%
4.4	KitKat	19	34.3%
5.0	Lollipop	21	16.9%
5.1		22	19.2%
6.0	Marshmallow	23	2.3%

Good {



Data as of March 2016

The app store “quality control” issue

Claimed benefit of Apple App Store vs. Google Play Store:

Apple tries to keep garbage apps out.

Google now has its “Bouncer” service:

Very little written in public.

Seems to be some combination of static and dynamic analysis.

Both Google and Apple can remotely uninstall malware apps.

Authentication

Apple and Google are furiously adding new features for this. E.g.,

“Smart Lock”: Your Android device pays attention to paired Bluetooth devices (car, watch, etc.) and decides whether to ask for your password.

Fingerprint reader: Much like Apple, much more user-friendly than passwords, and under various circumstances the phone will still ask for the password (e.g., when booting).

Federated identity: as in OpenID/Oauth, the user can approve and an app can authenticate as you without requiring your password.

Android: Security vs. Advertising

Smartphone security is tricky

Sensitive info available

Fine grained geolocation

User’s address book

Phone unique identifiers (IMEI, etc.)

Personal photos

Some apps abuse their access

Smartphone security is tricky

Sensitive info available

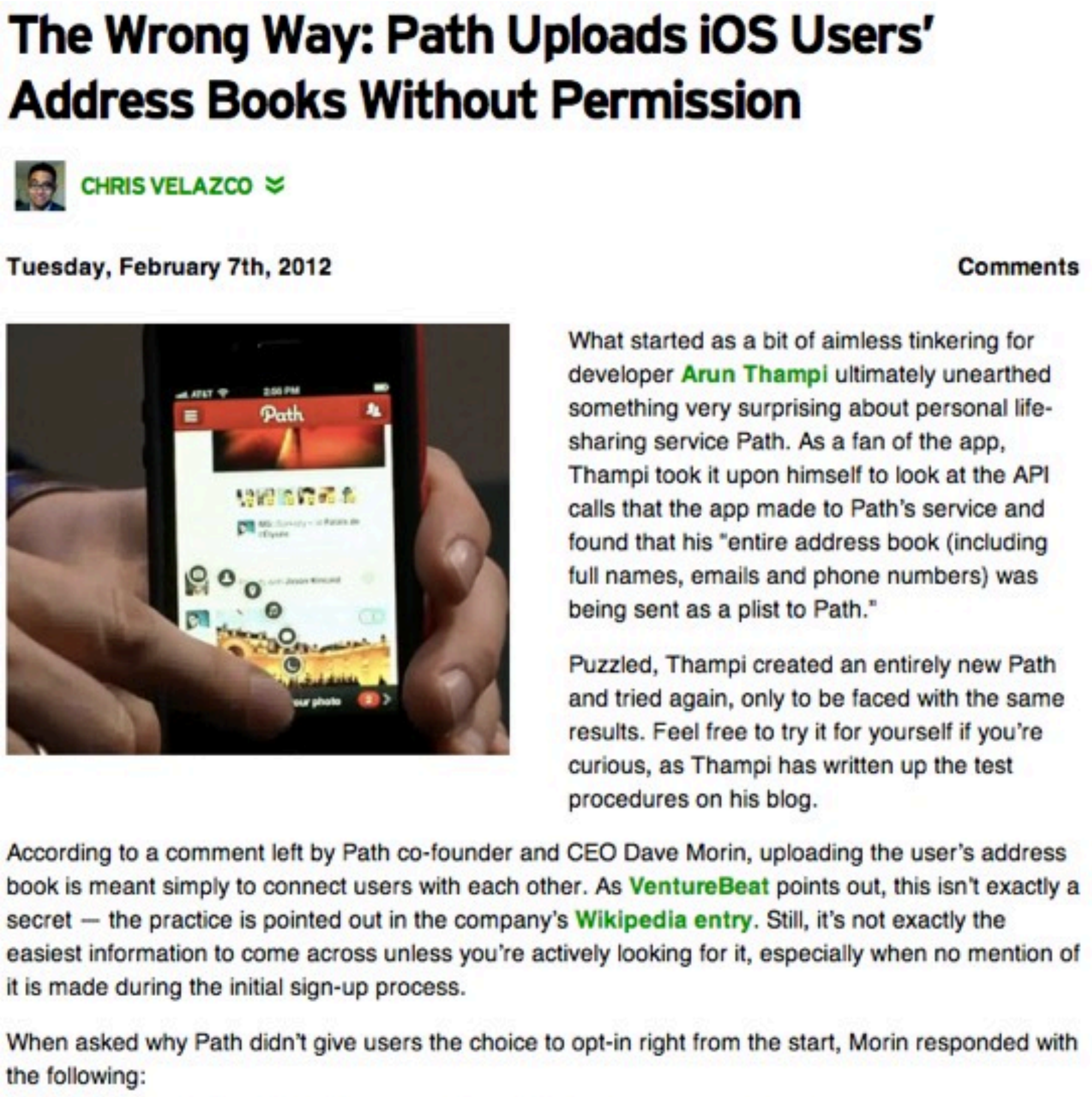
Fine grained geolocation

User’s address book

Phone unique identifiers (IMEI, etc.)

Personal photos

Some apps abuse their access



The Wrong Way: Path Uploads iOS Users' Address Books Without Permission

CHRIS VELAZCO

Tuesday, February 7th, 2012

Comments

What started as a bit of aimless tinkering for developer Arun Thampi ultimately unearthed something very surprising about personal life-sharing service Path. As a fan of the app, Thampi took it upon himself to look at the API calls that the app made to Path's service and found that his "entire address book (including full names, emails and phone numbers) was being sent as a plist to Path."

Puzzled, Thampi created an entirely new Path and tried again, only to be faced with the same results. Feel free to try it for yourself if you're curious, as Thampi has written up the test procedures on his blog.

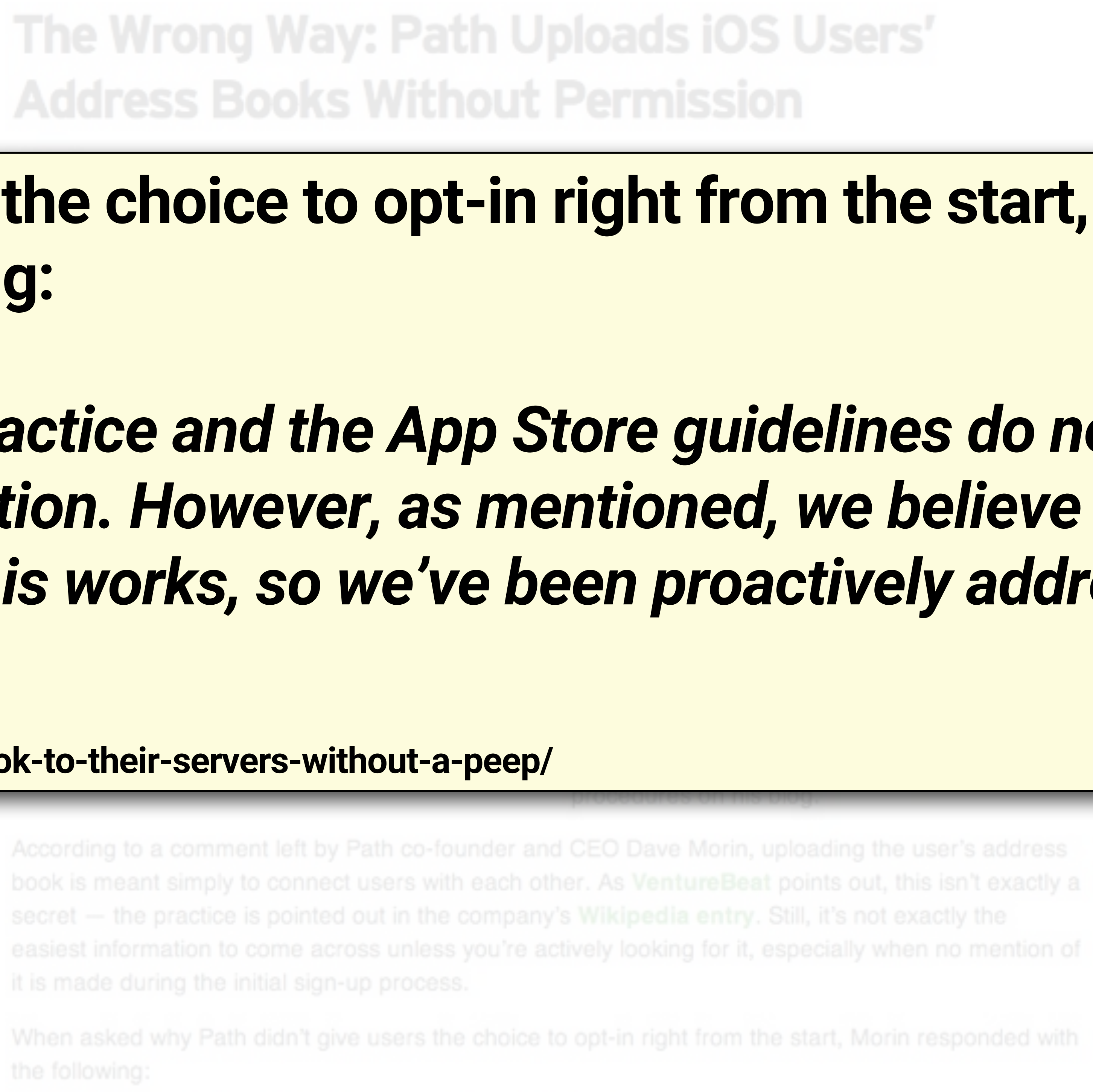
According to a comment left by Path co-founder and CEO Dave Morin, uploading the user's address book is meant simply to connect users with each other. As VentureBeat points out, this isn't exactly a secret — the practice is pointed out in the company's Wikipedia entry. Still, it's not exactly the easiest information to come across unless you're actively looking for it, especially when no mention of it is made during the initial sign-up process.

When asked why Path didn't give users the choice to opt-in right from the start, Morin responded with the following:

Smartphone security is tricky

Sensitive info available

Fine grained geolocation



The Wrong Way: Path Uploads iOS Users' Address Books Without Permission

procedures on his blog.

According to a comment left by Path co-founder and CEO Dave Morin, uploading the user's address book is meant simply to connect users with each other. As VentureBeat points out, this isn't exactly a secret — the practice is pointed out in the company's Wikipedia entry. Still, it's not exactly the easiest information to come across unless you're actively looking for it, especially when no mention of it is made during the initial sign-up process.

When asked why Path didn't give users the choice to opt-in right from the start, Morin responded with the following:

When asked why Path didn't give users the choice to opt-in right from the start, [Path CEO] Morin responded with the following:

> ***This is currently the industry best practice and the App Store guidelines do not specifically discuss contact information. However, as mentioned, we believe users need further transparency on how this works, so we've been proactively addressing this.***

techcrunch.com/2012/02/07/path-uploads-your-iphones-address-book-to-their-servers-without-a-peep/

Smartphone research

New OS services

Michael Dietz, Shashi Shekhar, Yuliy Pisetsky, Anhei Shu, and Dan S. Wallach, **Quire: Lightweight Provenance for Smartphone Operating Systems**, 21st USENIX Security Symposium (San Francisco, CA), August 2011.

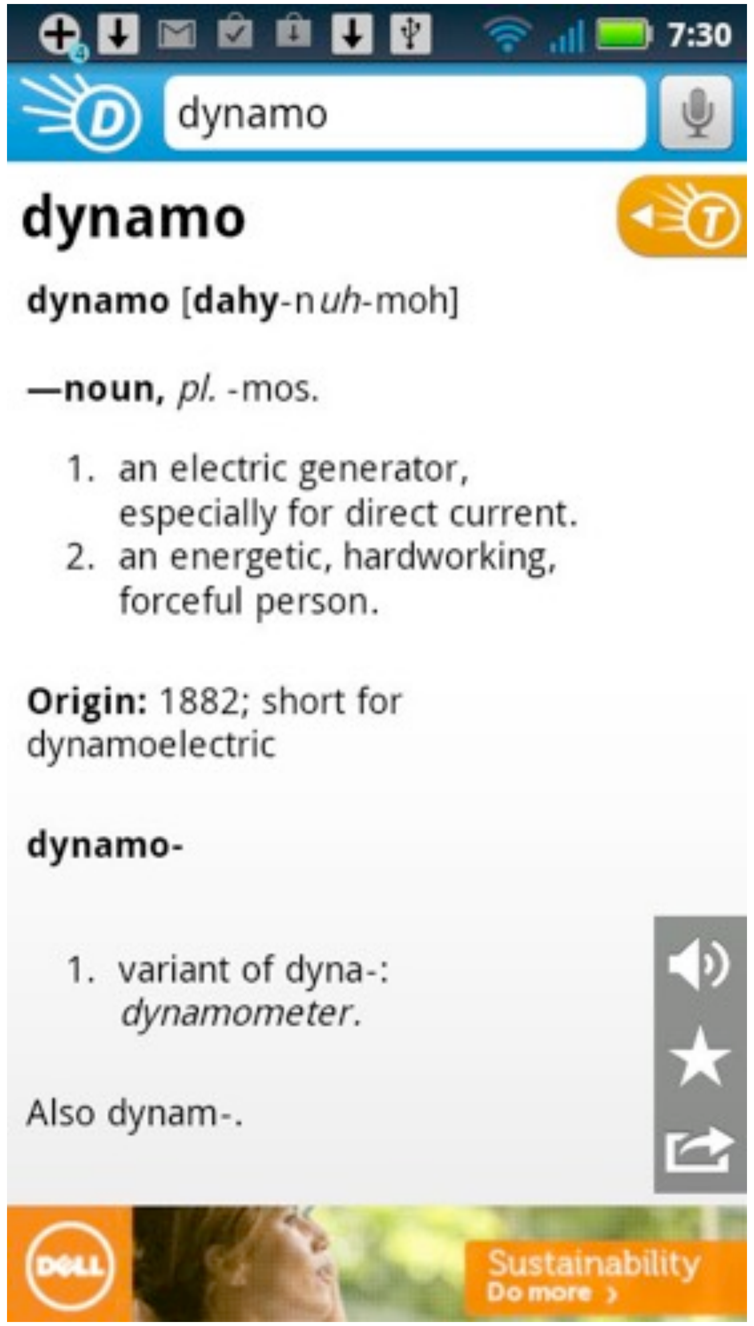
Shashi Shekhar, Michael Dietz, Dan S. Wallach, **AdSplit: Separating smartphone advertising from applications**, 22nd USENIX Security Symposium (Bellevue, WA), August 2012.

Measurements

Theodore Book, Adam Pridgen, and Dan S. Wallach, **Longitudinal analysis of Android ad library permissions**. Mobile Security Technologies (MOST) 2013.

Theodore Book and Dan S. Wallach, **A case of collution: A study of the interface between ad libraries and their apps**. 3rd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM), November 2013.

ADS!



dynamo

dynamo [**dahy**-n*uh*-moh]

—noun, *pl.* -mos.

1. an electric generator, especially for direct current.
2. an energetic, hardworking, forceful person.

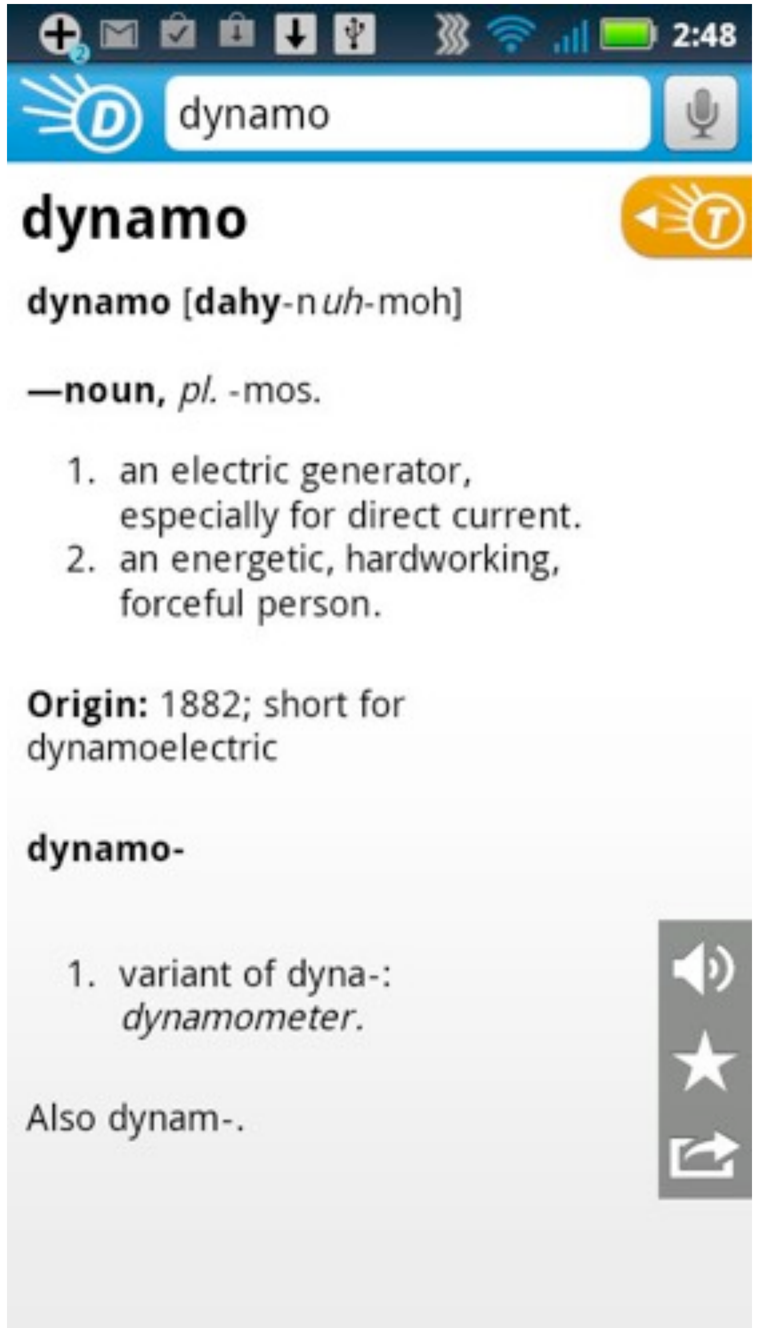
Origin: 1882; short for dynamoelectric

dynamo-

1. variant of dyna-: *dynamometer.*

Also dynam-.

Cost : Free



dynamo

dynamo [**dahy**-n*uh*-moh]

—noun, *pl.* -mos.

1. an electric generator, especially for direct current.
2. an energetic, hardworking, forceful person.

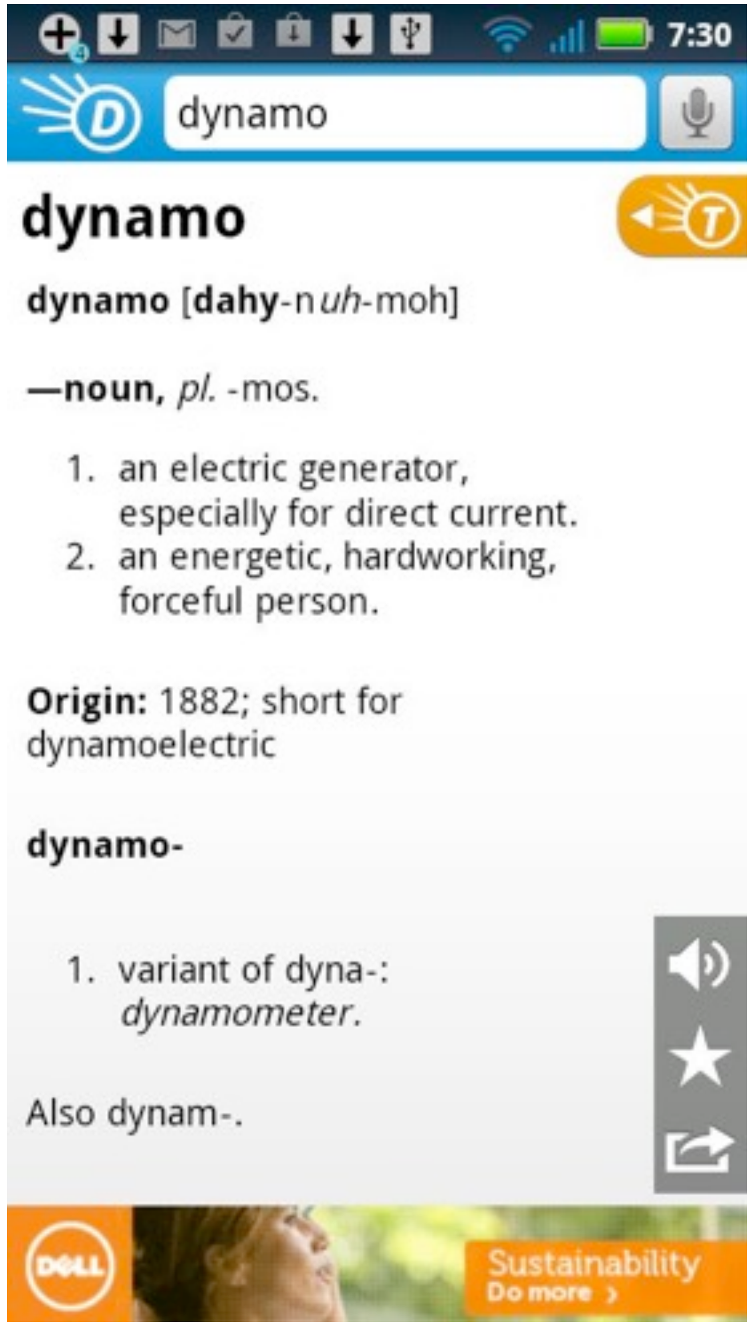
Origin: 1882; short for dynamoelectric

dynamo-

1. variant of dyna-: *dynamometer.*

Also dynam-.

Cost : $2.99



dynamo

dynamo [**dahy**-n*uh*-moh]

—noun, *pl.* -mos.

1. an electric generator, especially for direct current.
2. an energetic, hardworking, forceful person.

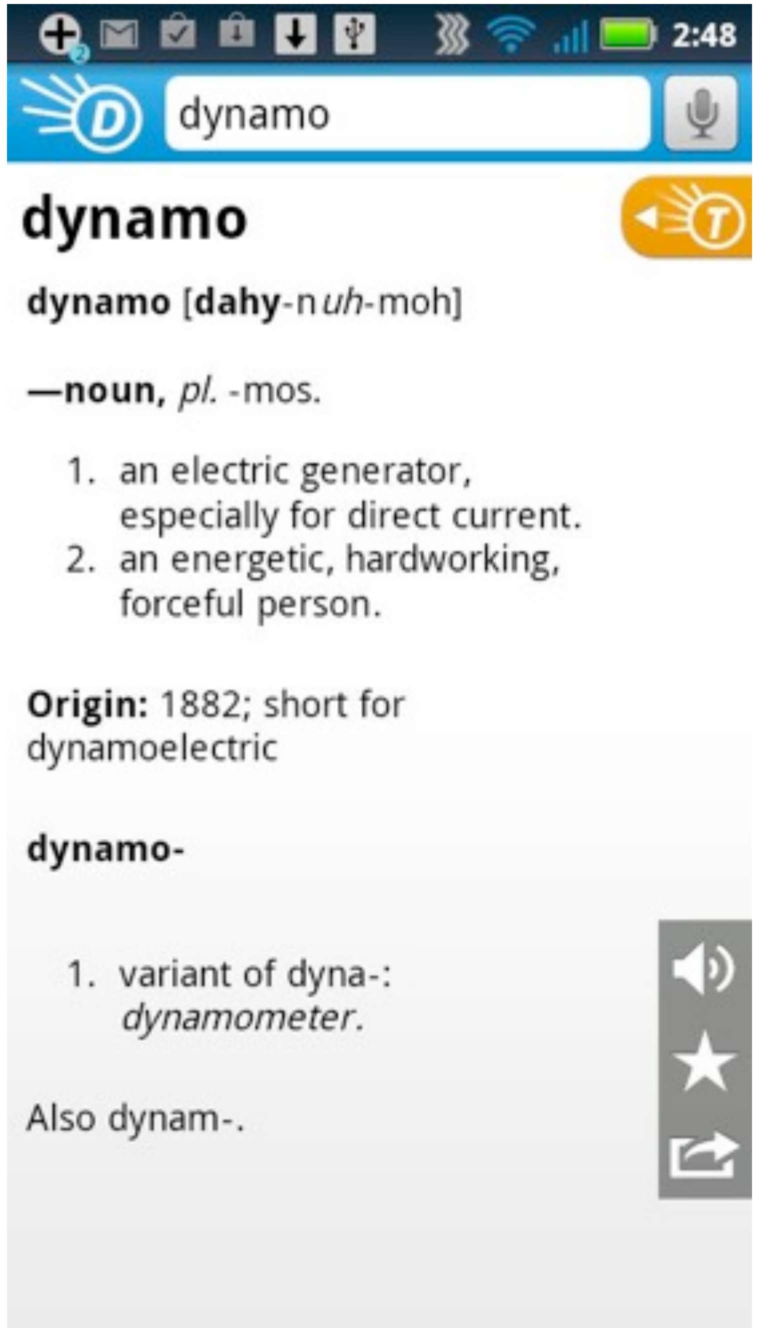
Origin: 1882; short for dynamoelectric

dynamo-

1. variant of dyna-: *dynamometer.*

Also dynam-.

Cost : Free



dynamo

dynamo [**dahy**-n*uh*-moh]

—noun, *pl.* -mos.

1. an electric generator, especially for direct current.
2. an energetic, hardworking, forceful person.

Origin: 1882; short for dynamoelectric

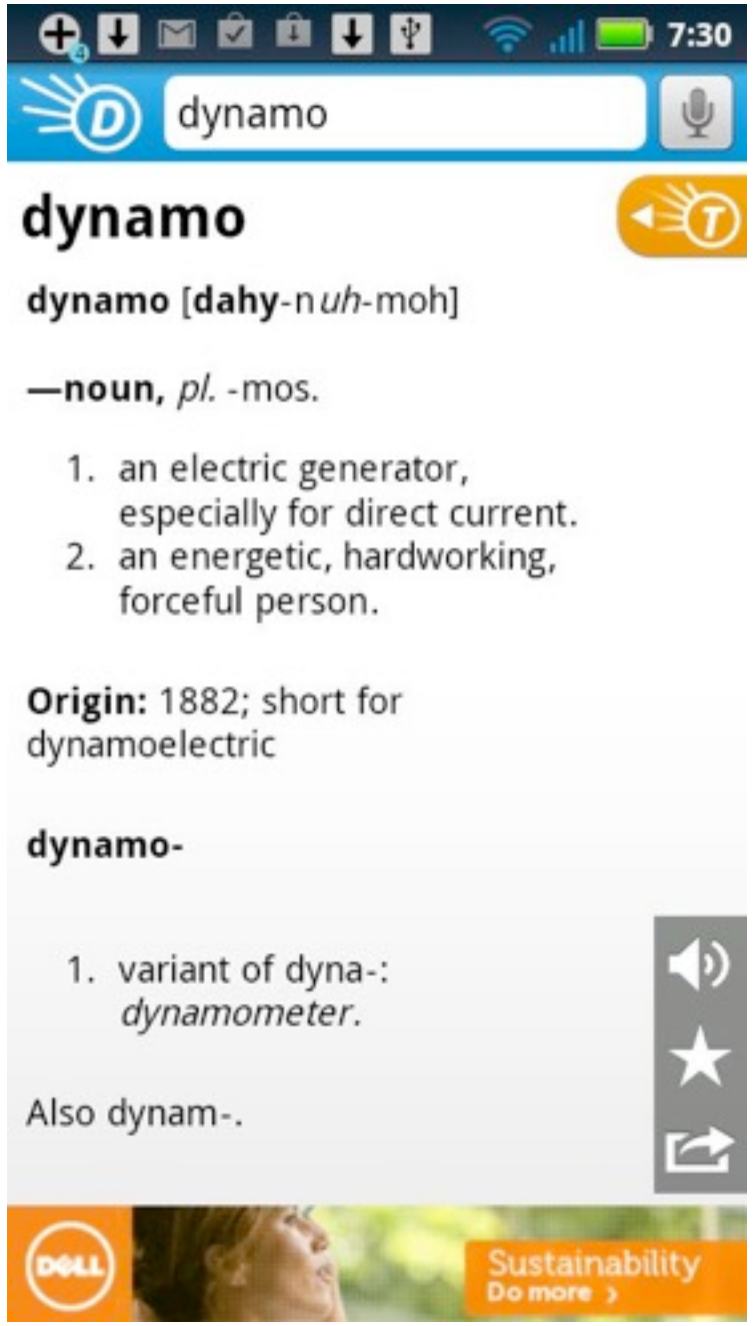
dynamo-

1. variant of dyna-: *dynamometer.*

Also dynam-.

Cost : $2.99

Downloads:
100,000 – 500,000



dynamo

dynamo [**dahy**-n*uh*-moh]

—noun, *pl.* -mos.

1. an electric generator, especially for direct current.
2. an energetic, hardworking, forceful person.

Origin: 1882; short for dynamoelectric

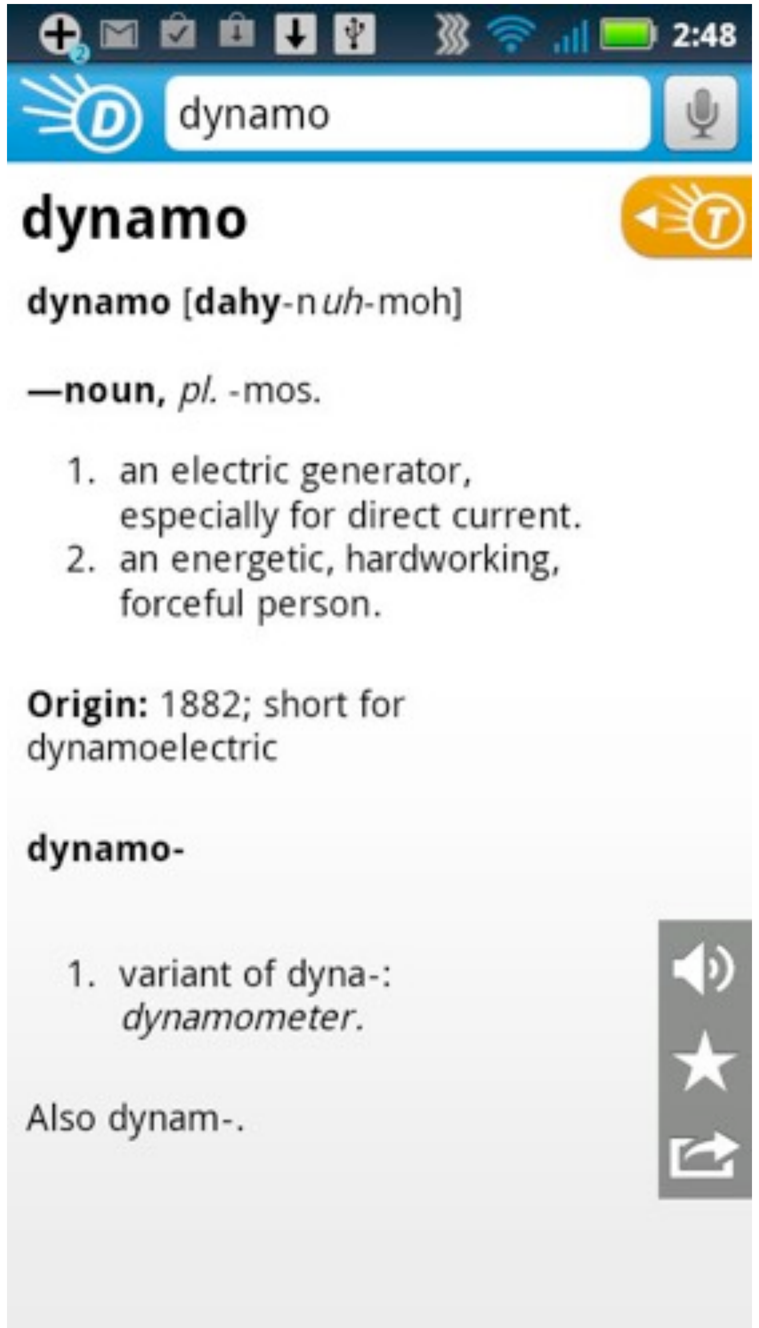
dynamo-

1. variant of dyna-: *dynamometer.*

Also dynam-.

Cost : Free

Downloads:
10,000,000 – 50,000,000



dynamo

dynamo [**dahy**-n*uh*-moh]

—noun, *pl.* -mos.

1. an electric generator, especially for direct current.
2. an energetic, hardworking, forceful person.

Origin: 1882; short for dynamoelectric

dynamo-

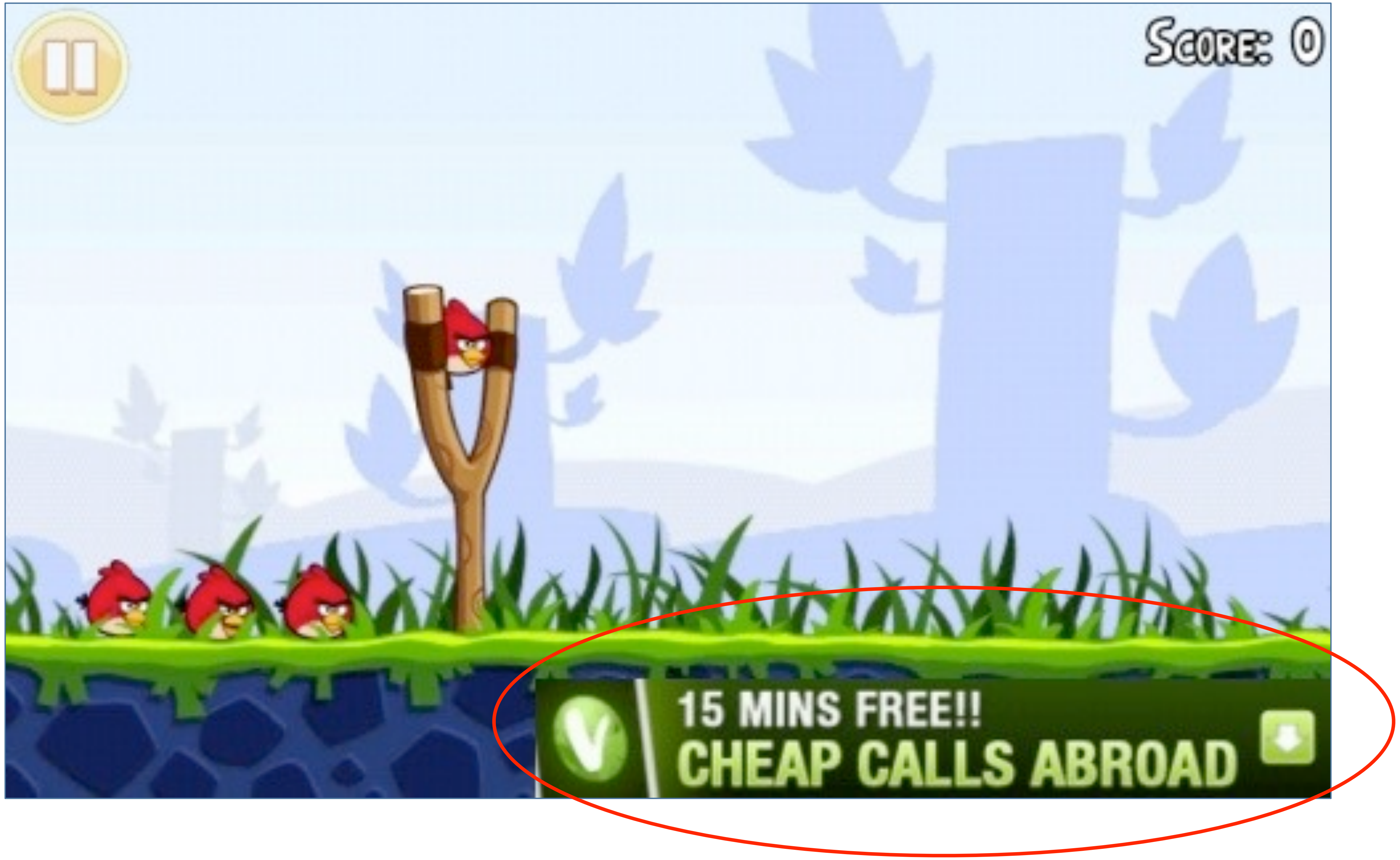
1. variant of dyna-: *dynamometer.*

Also dynam-.

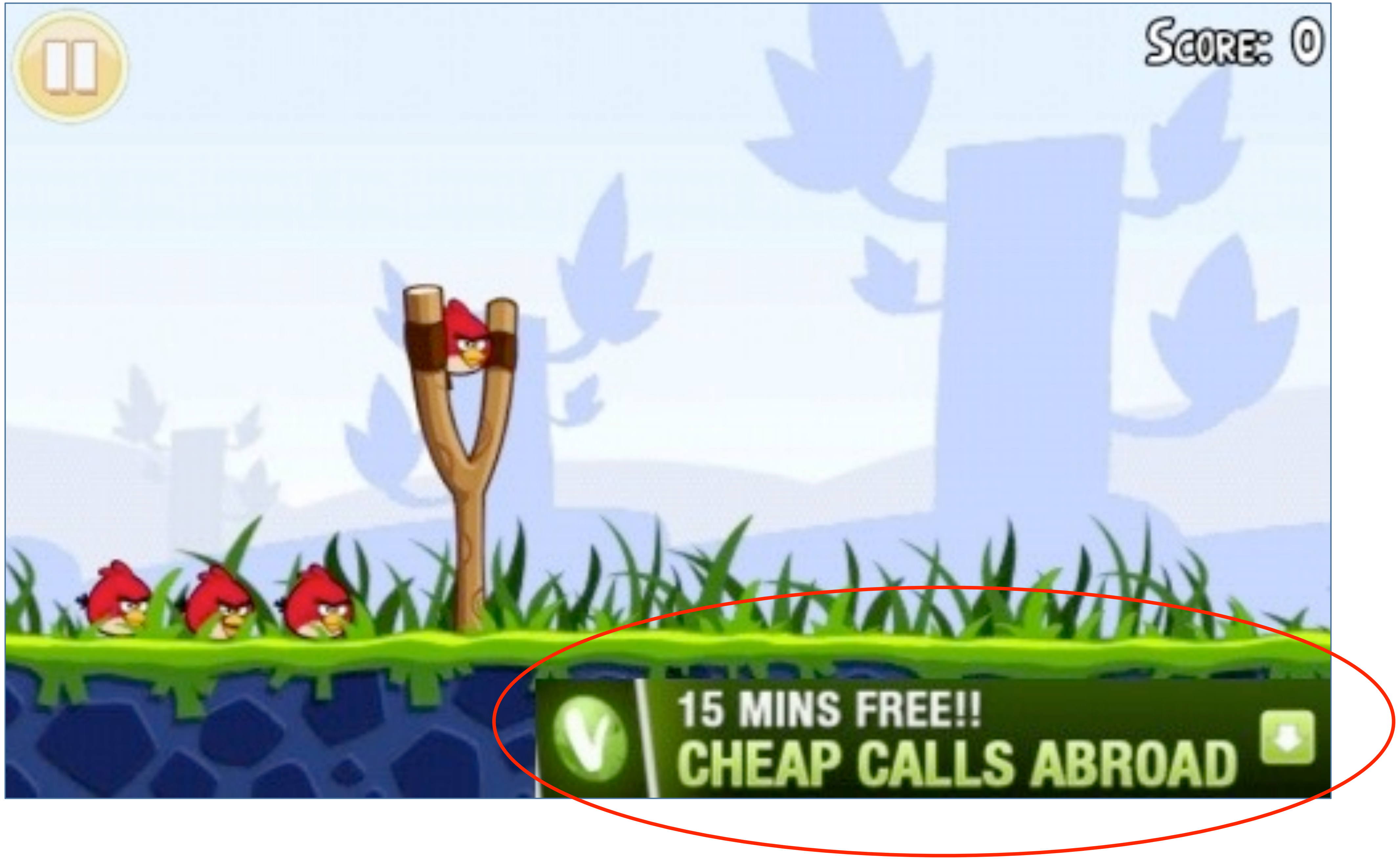
Cost : $2.99

Downloads:
100,000 – 500,000

Ads are widely used



Ads are widely used



(and advertising uses 75% of the power budget - Pathak et al., Eurosys 2012)

Mobile advertisements

Ecosystem similar to web

App developer gets money for hosting ads

Ads are third party libraries included with the app

Targeted advertising 101

More user data = Better targeting = $

Ad libraries incentivize developers to leak user data.

Even worse... permission usage

More user data = More permissions

Permission bloat: Apps requesting permissions exclusively for advertisements.

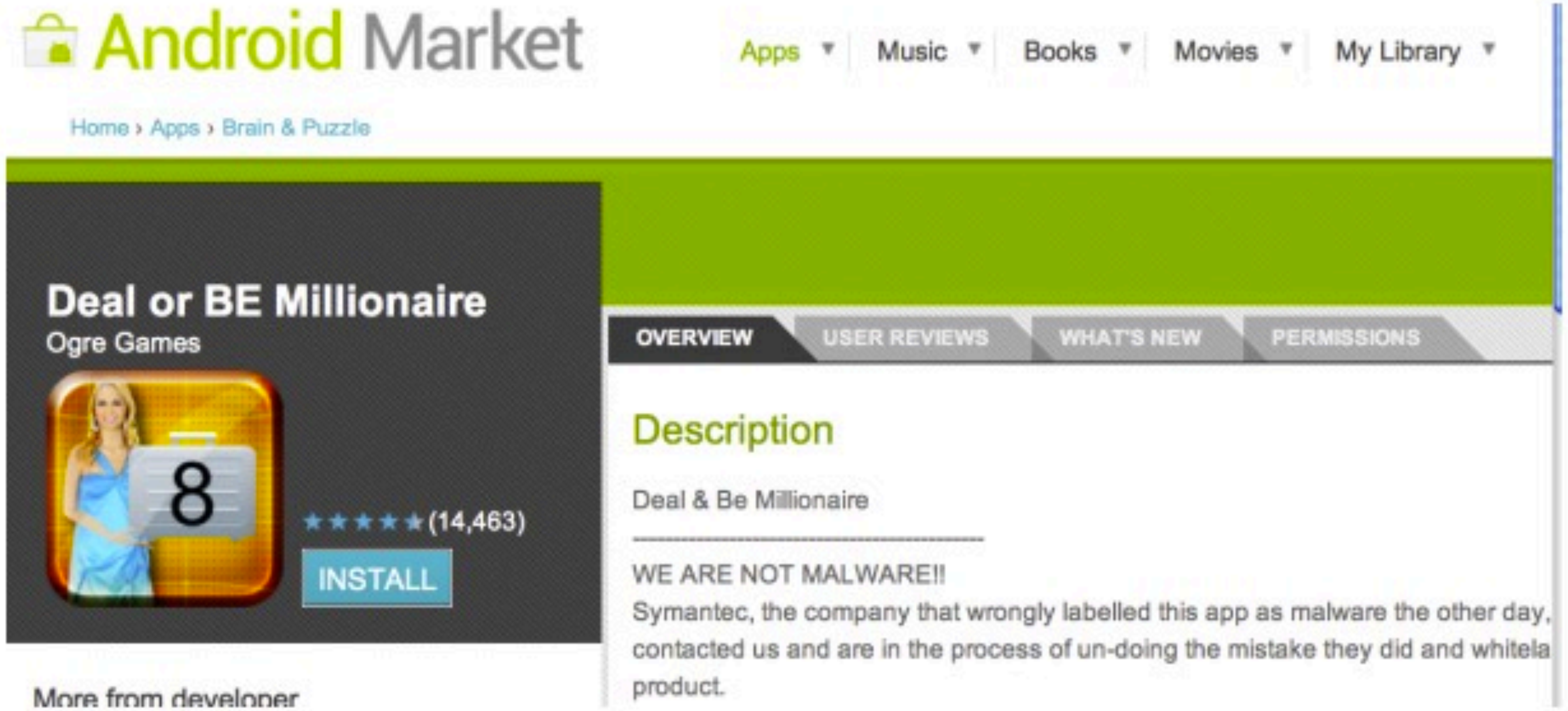
CNET › News › InSecurity Complex

Dubious Android apps may not be malware--just ads

Verizon-affiliated ICSA Labs steps into the controversy over Android apps that Symantec identified as malware.

by Elinor Mills | February 1, 2012 1:21 PM PST

Follow



Deal or BE Millionaire

Ogre Games

Description

Deal & Be Millionaire

WE ARE NOT MALWARE!!

Symantec, the company that wrongly labelled this app as malware the other day, contacted us and are in the process of un-doing the mistake they did and whitela product.

Permission use over time: is it getting worse?

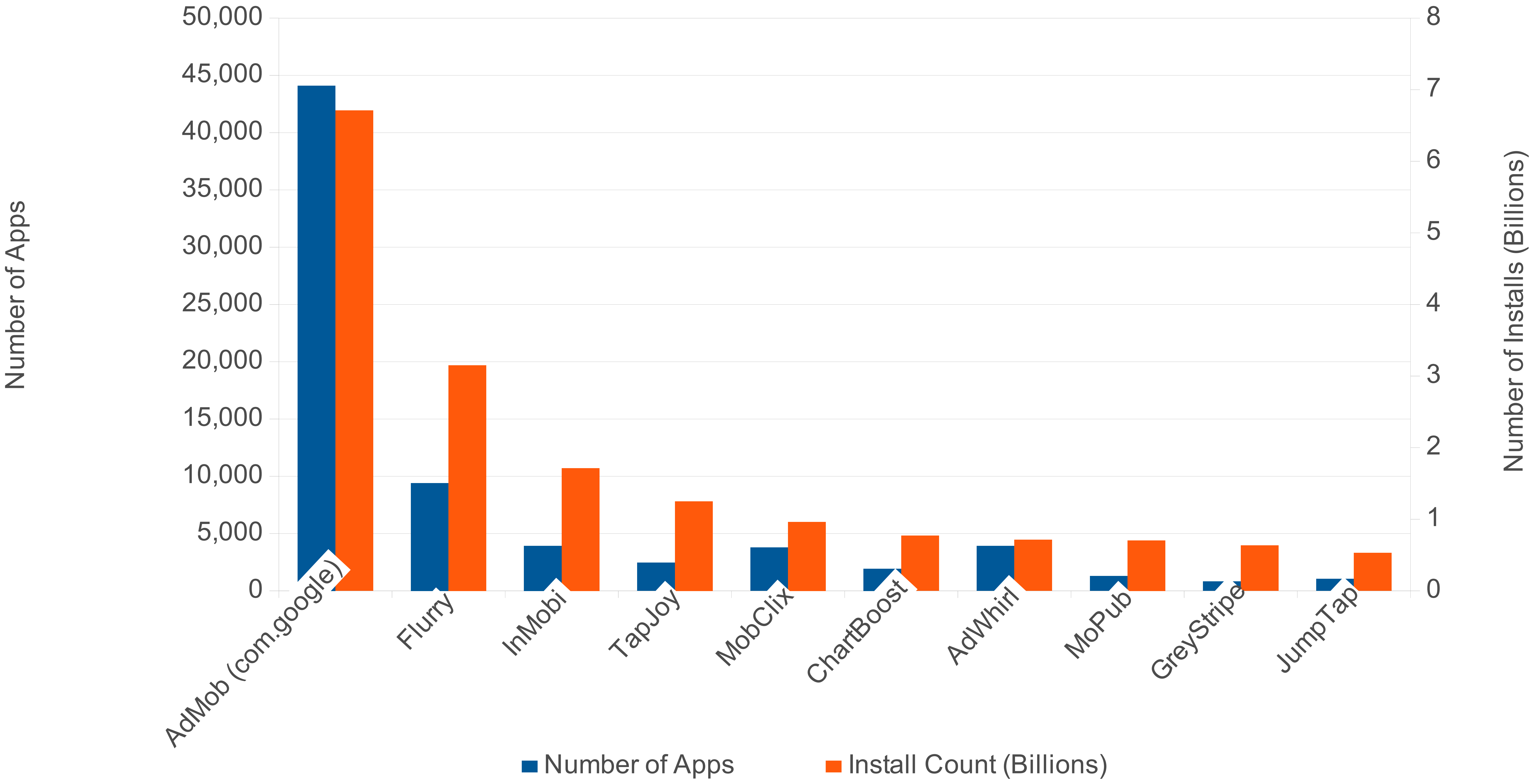
Ad libraries

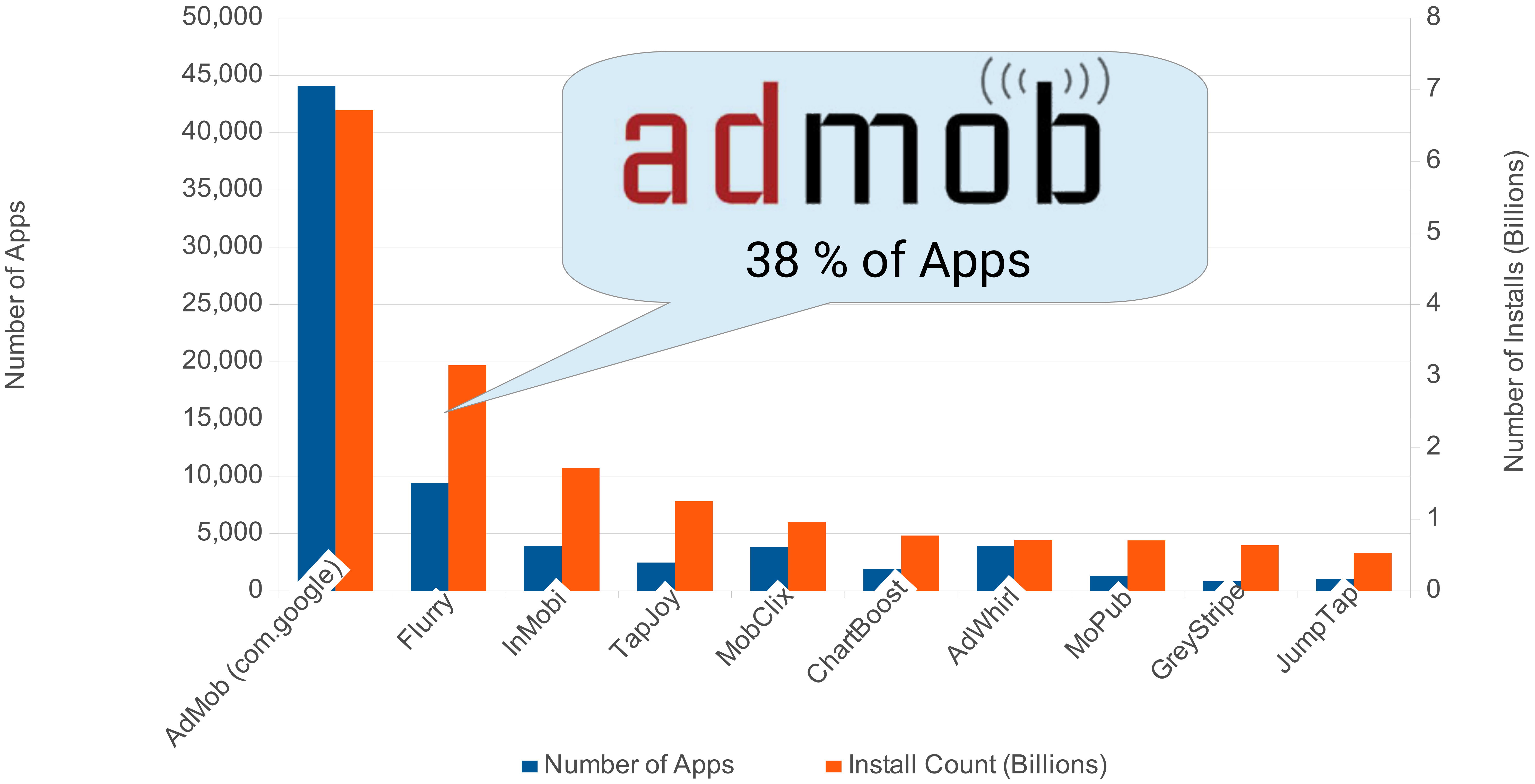
114,000 apps analyzed

56% contained at least one ad library

108,000 ad library copies identified

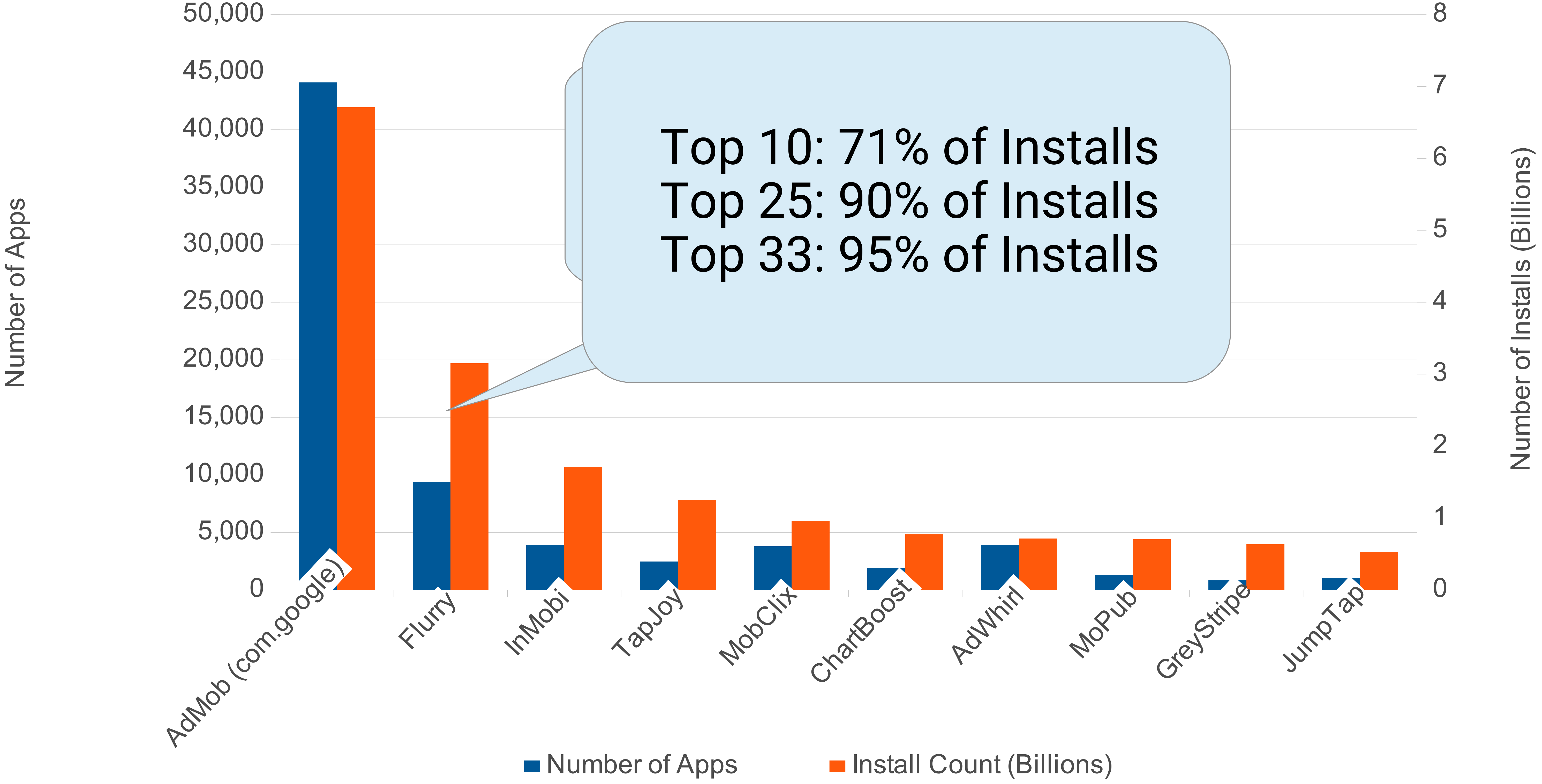
68 different ad library families identified


Number of Apps
50,000
45,000
40,000
35,000
30,000
25,000
20,000
15,000
10,000
5,000
0
Number of Installs (Billions)
8
7
6
5
4
3
2
1
0
AdMob (com.google)
Flurry
InMobi
TapJoy
MobClix
ChartBoost
AdWhirl
MoPub
GreyStripe
JumpTap
Number of Apps
Install Count (Billions)



admob

38 % of Apps



Top 10: 71% of Installs
Top 25: 90% of Installs
Top 33: 95% of Installs

Dating ad libraries

Metadata for apps retrieved from Google Play:

Install count (As a range: e.g. 5,000 – 10,000)

Release date for latest version

The release date of the earliest app using a library approximates the library release date.

Measuring permission usage

Separate library code from application code

Simple static analysis of library code to extract Android API calls

Stowaway (Felt, et al., 2011)

Map API calls to Android permissions

PScout (Au, et al., 2012)

Caveats

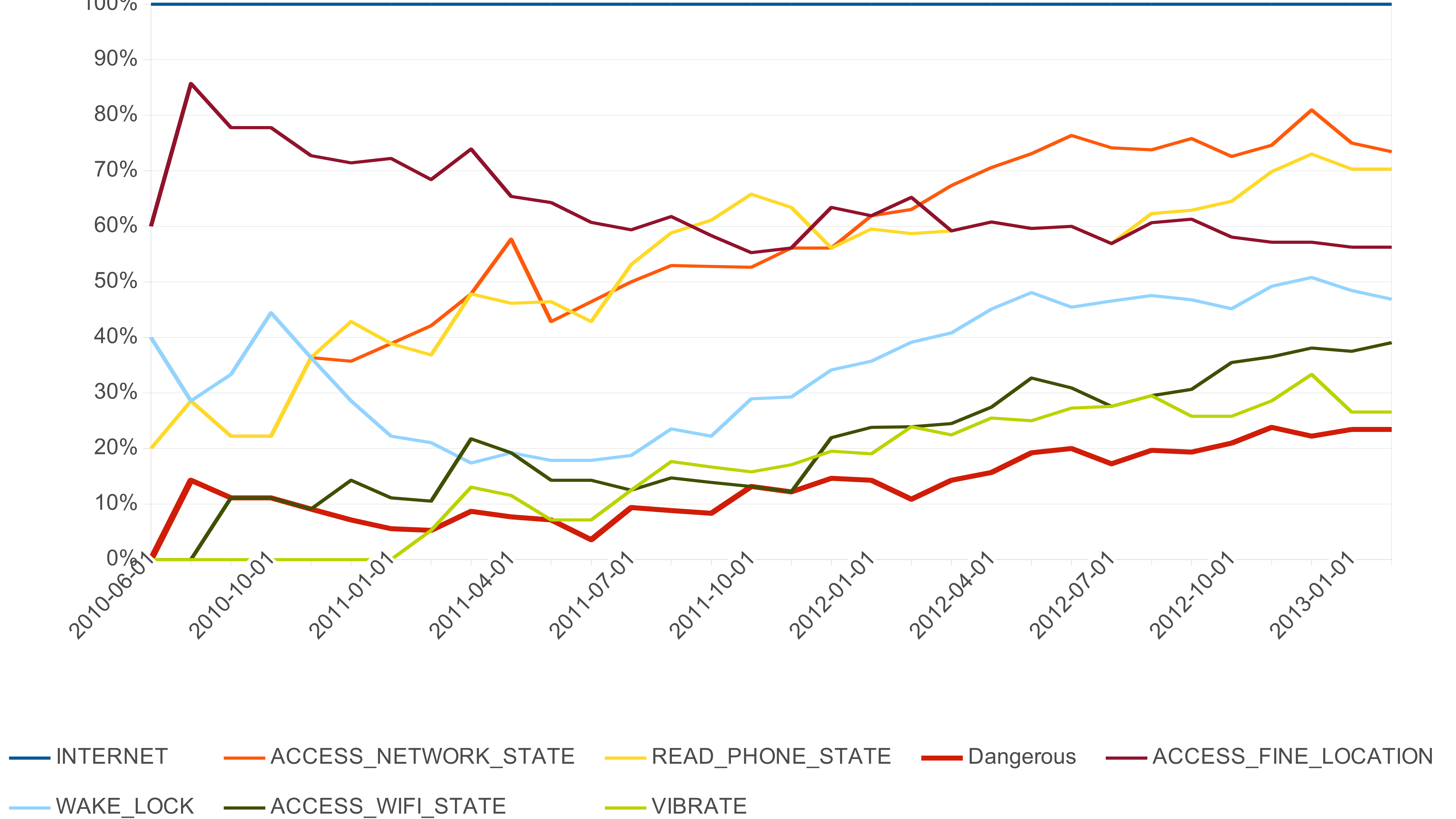
We are examining libraries, not applications

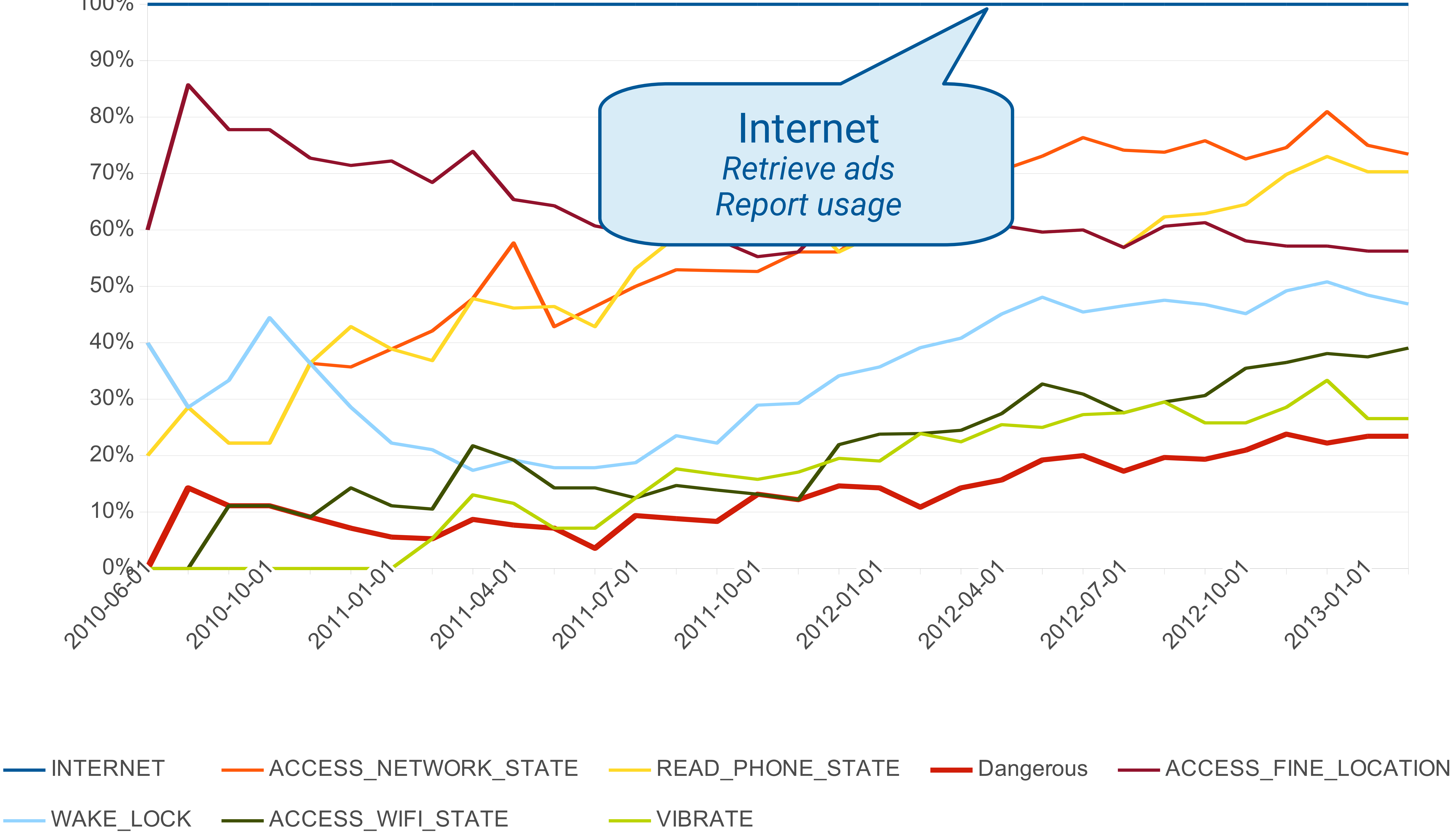
Don't verify if a library routine is used by any particular application

Don't verify if necessary permissions found in manifest

Don't detect dynamically loaded / generated code

(Grace, et al., “Unsafe Exposure,” 2012)

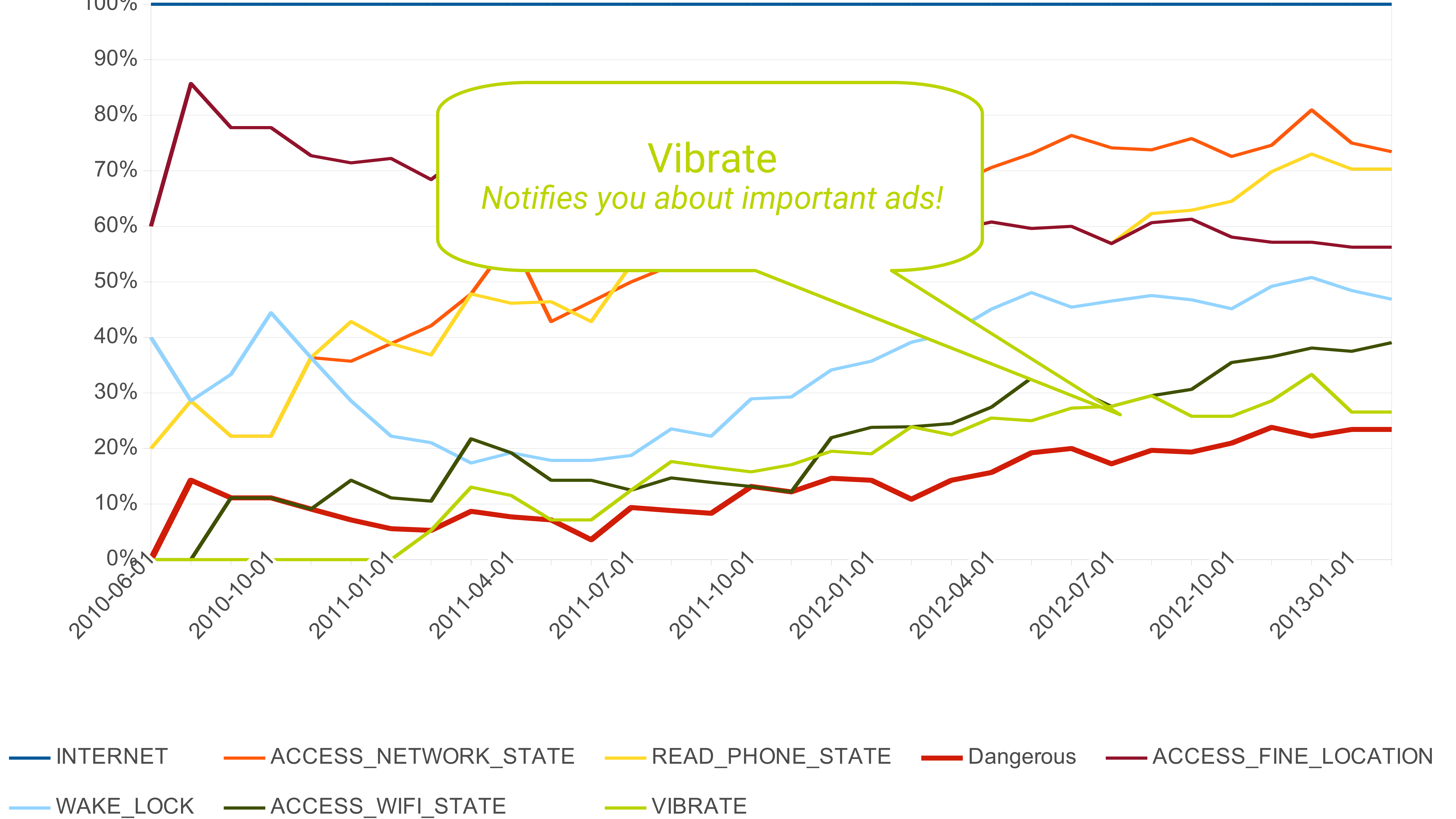

100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
2010-06-01
2010-10-01
2011-01-01
2011-04-01
2011-07-01
2011-10-01
2012-01-01
2012-04-01
2012-07-01
2012-10-01
2013-01-01
INTERNET
ACCESS_NETWORK_STATE
READ_PHONE_STATE
Dangerous
ACCESS_FINE_LOCATION
WAKE_LOCK
ACCESS_WIFI_STATE
VIBRATE



Internet

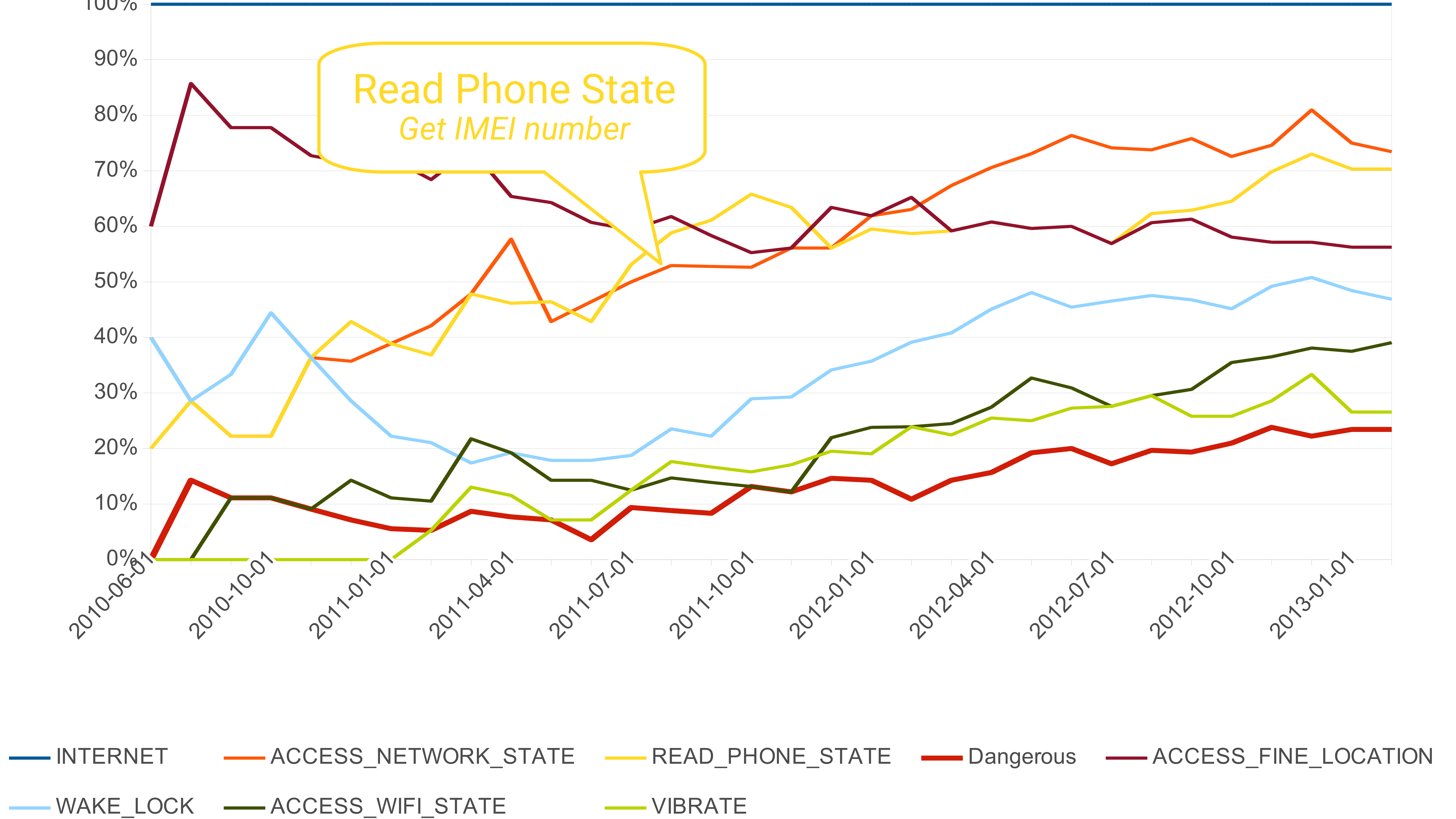
Retrieve ads

Report usage



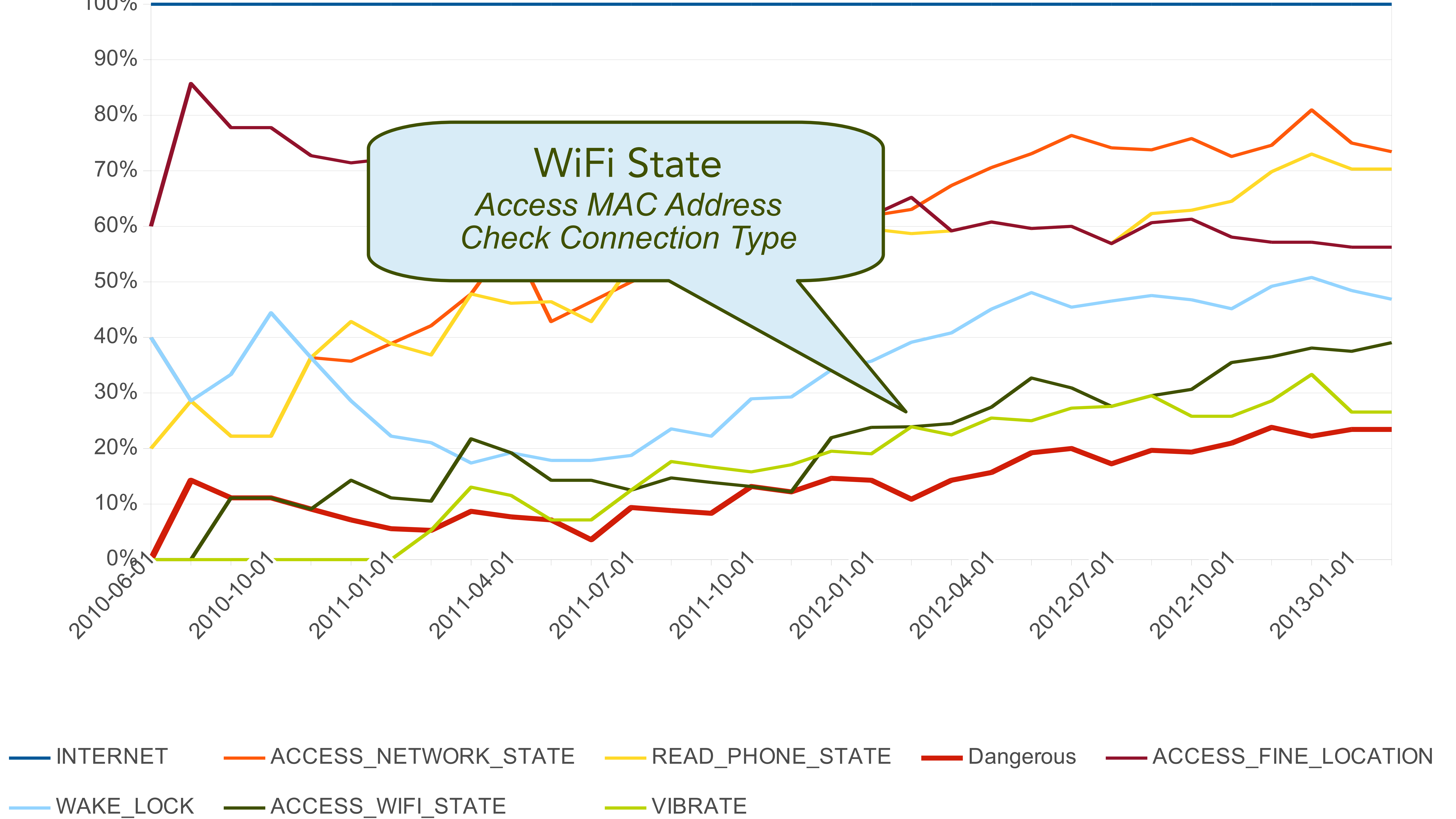
Vibrate

Notifies you about important ads!



Read Phone State

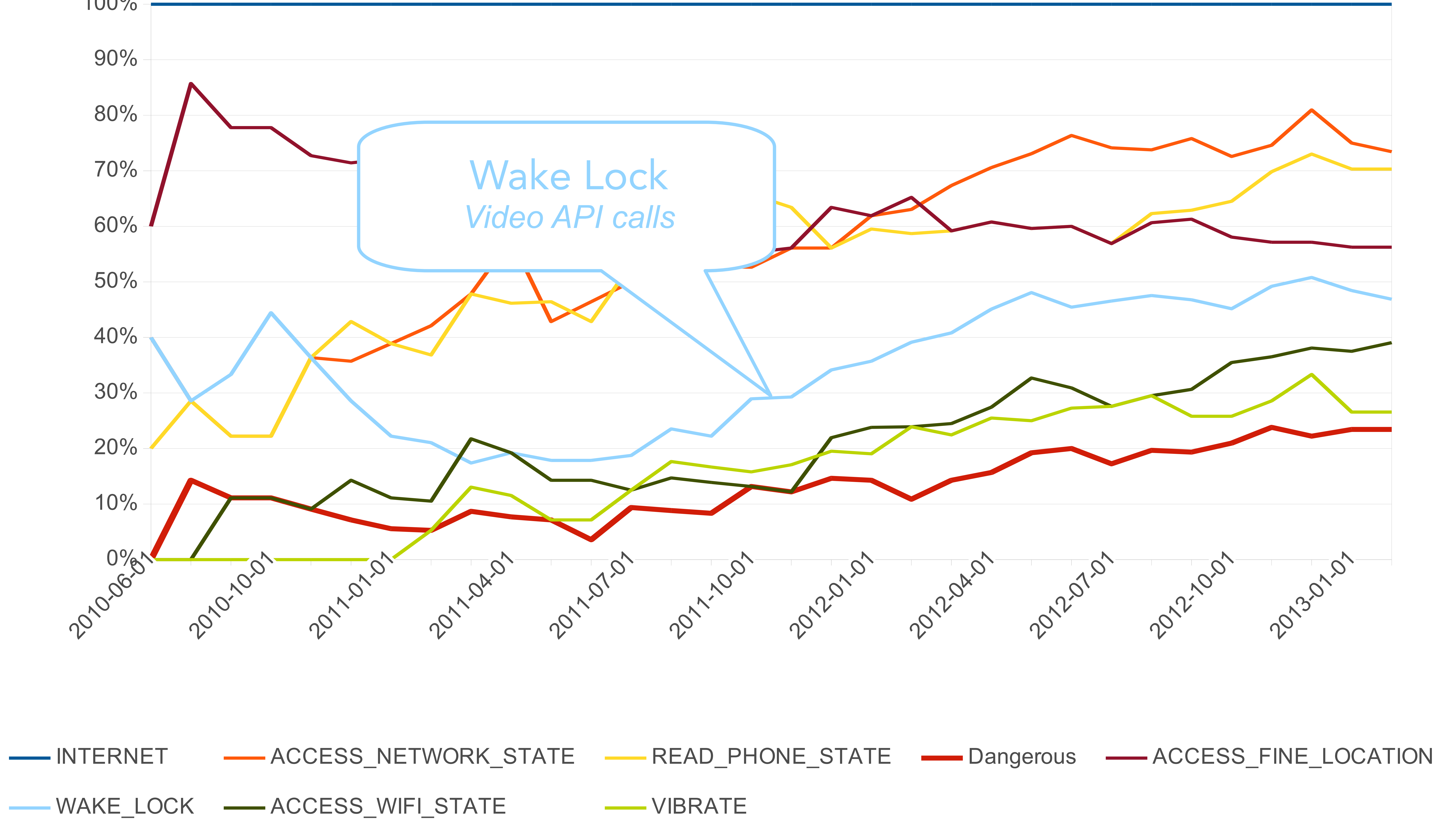
Get IMEI number



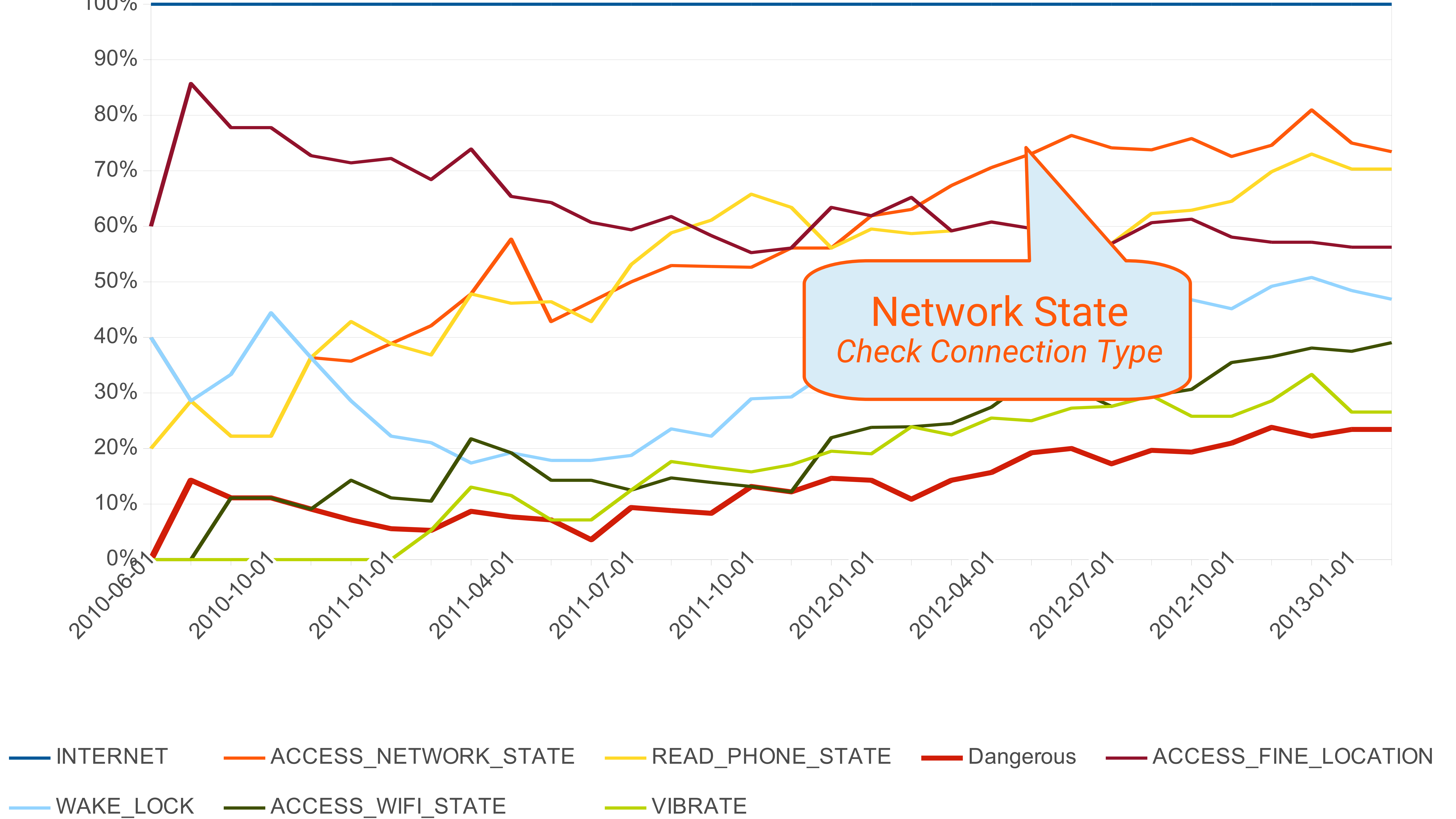
WiFi State

Access MAC Address

Check Connection Type

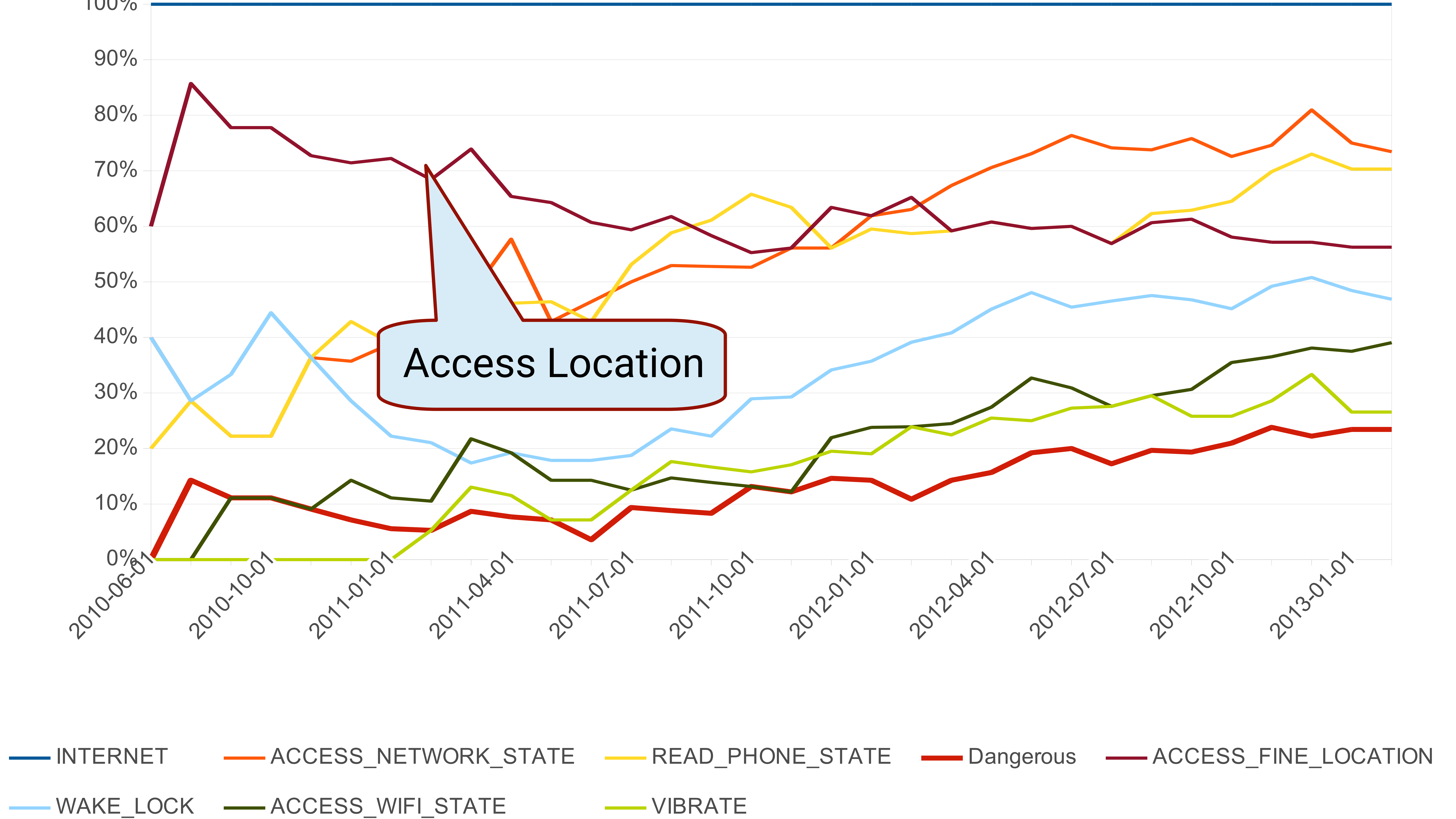


Wake Lock
Video API calls

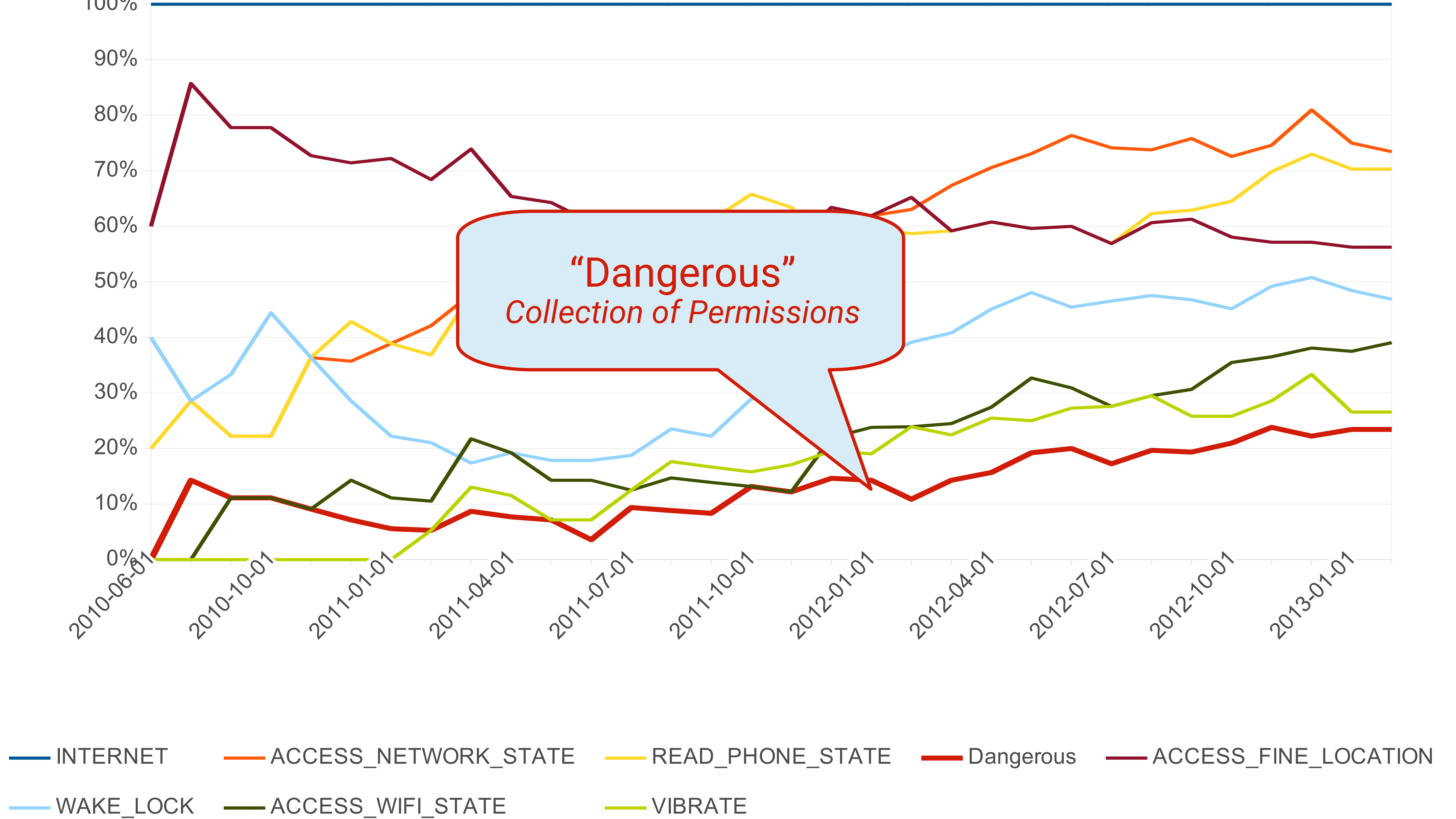


Network State

Check Connection Type

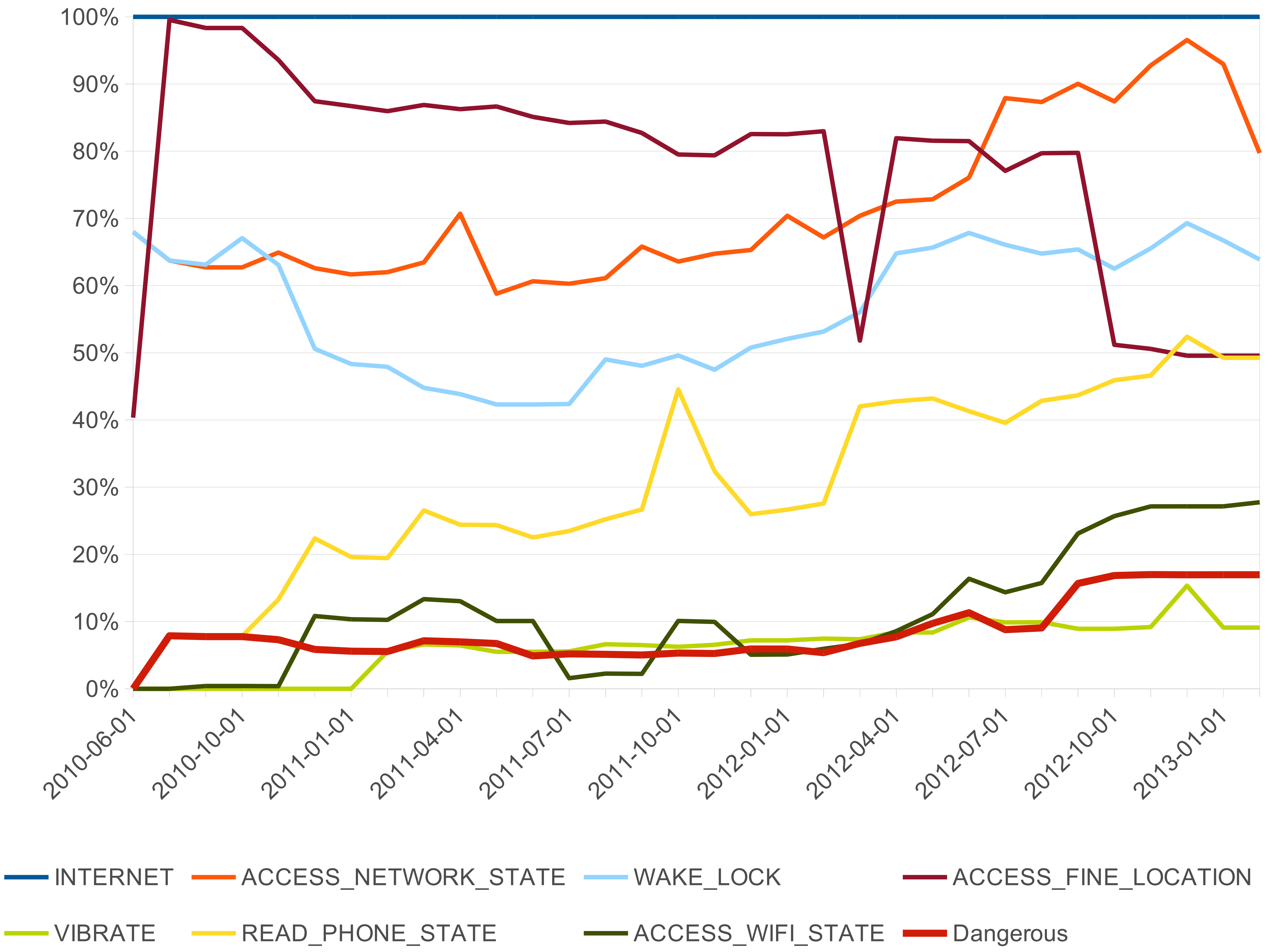


Access Location

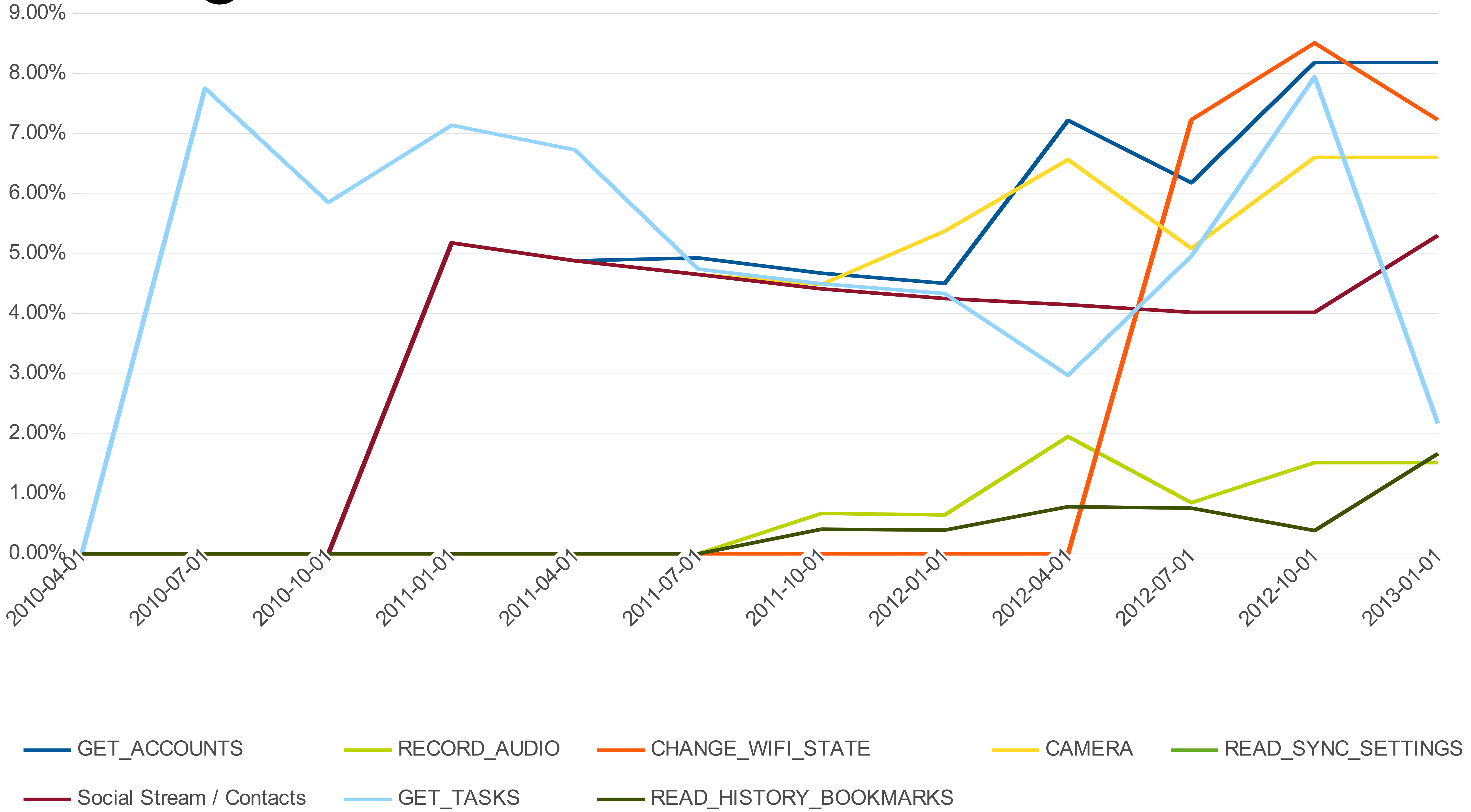


"Dangerous"
Collection of Permissions

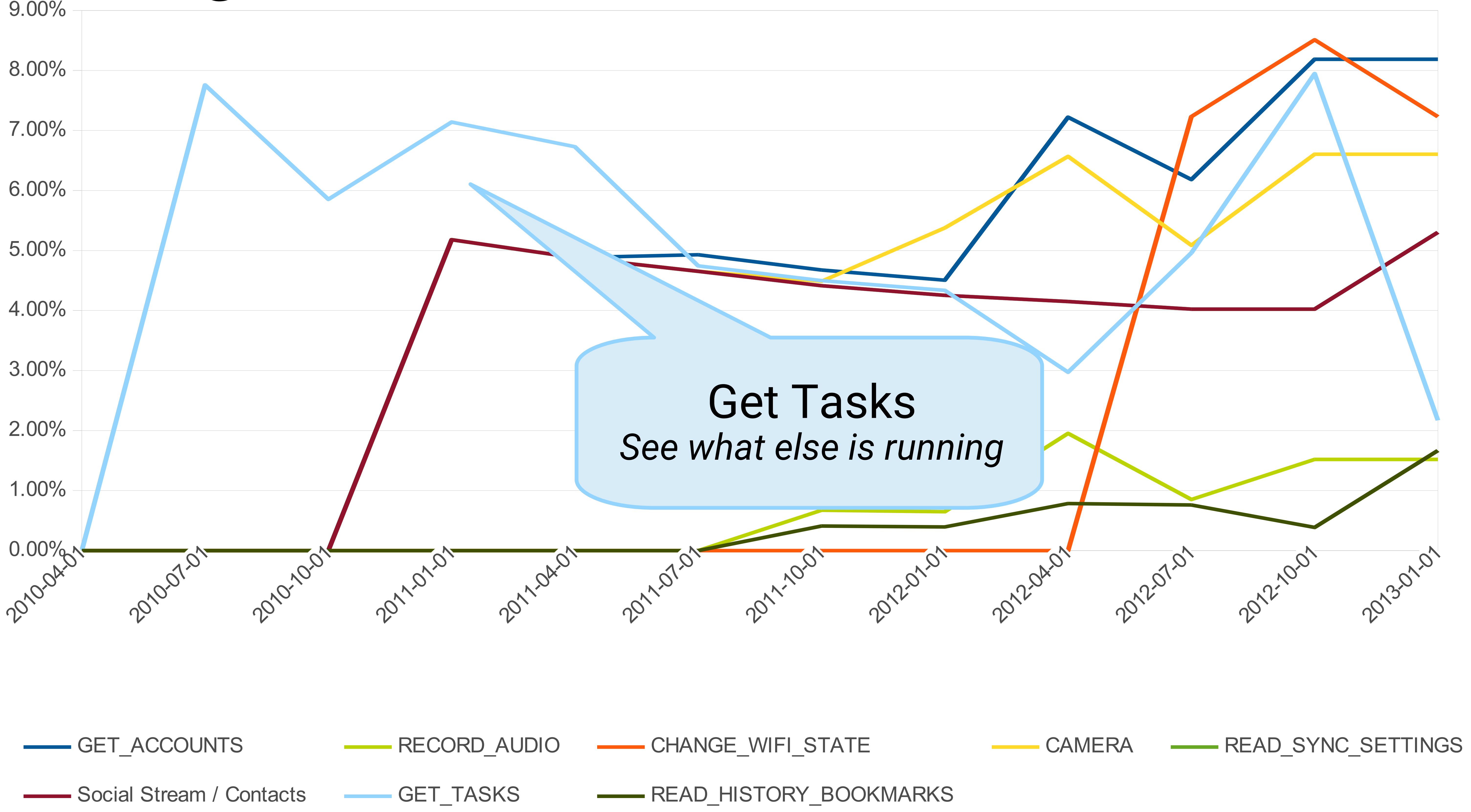
Install-weighted permissions



“Dangerous” Permissions



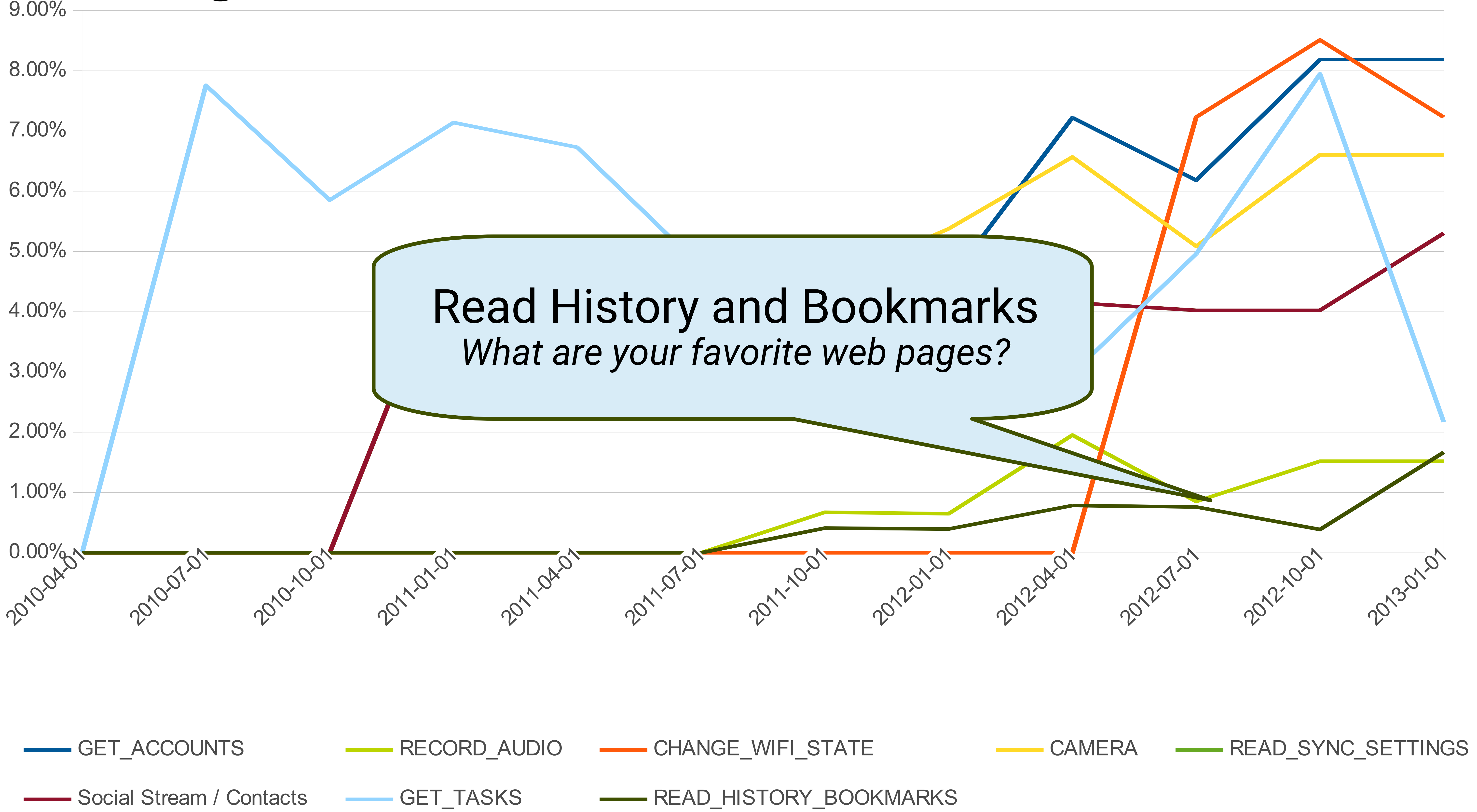
“Dangerous” Permissions



Get Tasks

See what else is running

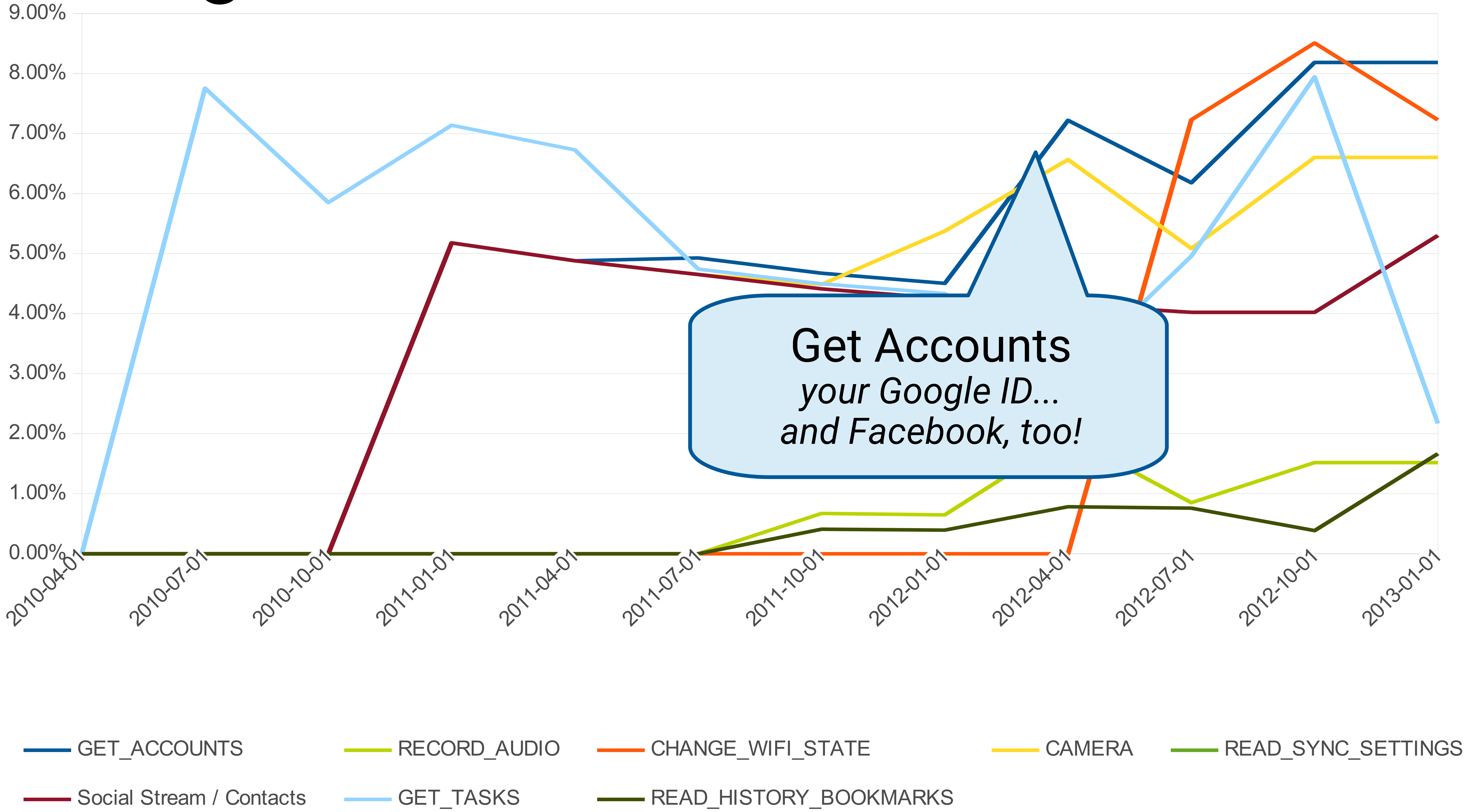
“Dangerous” Permissions



Read History and Bookmarks

What are your favorite web pages?

“Dangerous” Permissions

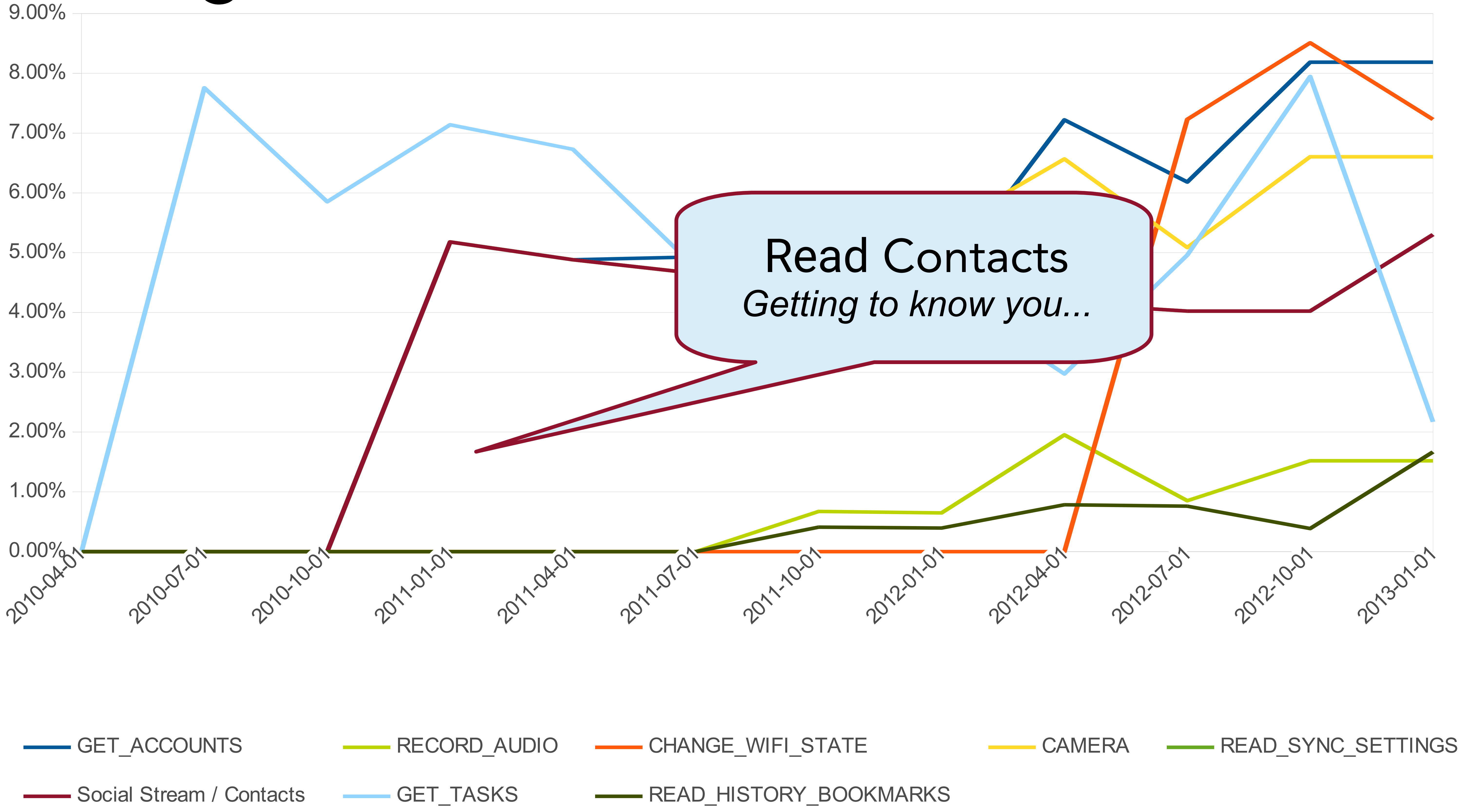


Get Accounts

your Google ID...

and Facebook, too!

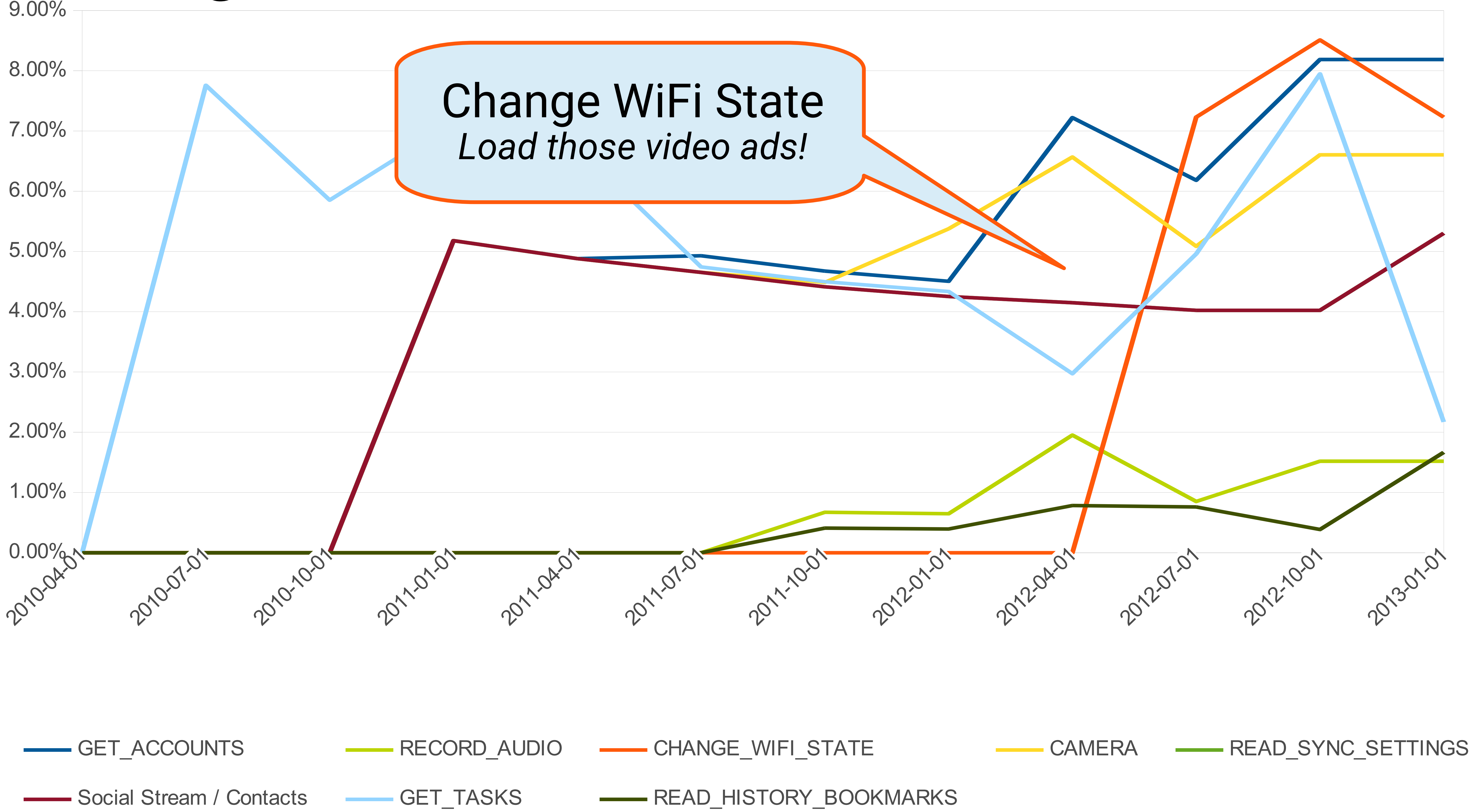
“Dangerous” Permissions



Read Contacts

Getting to know you...

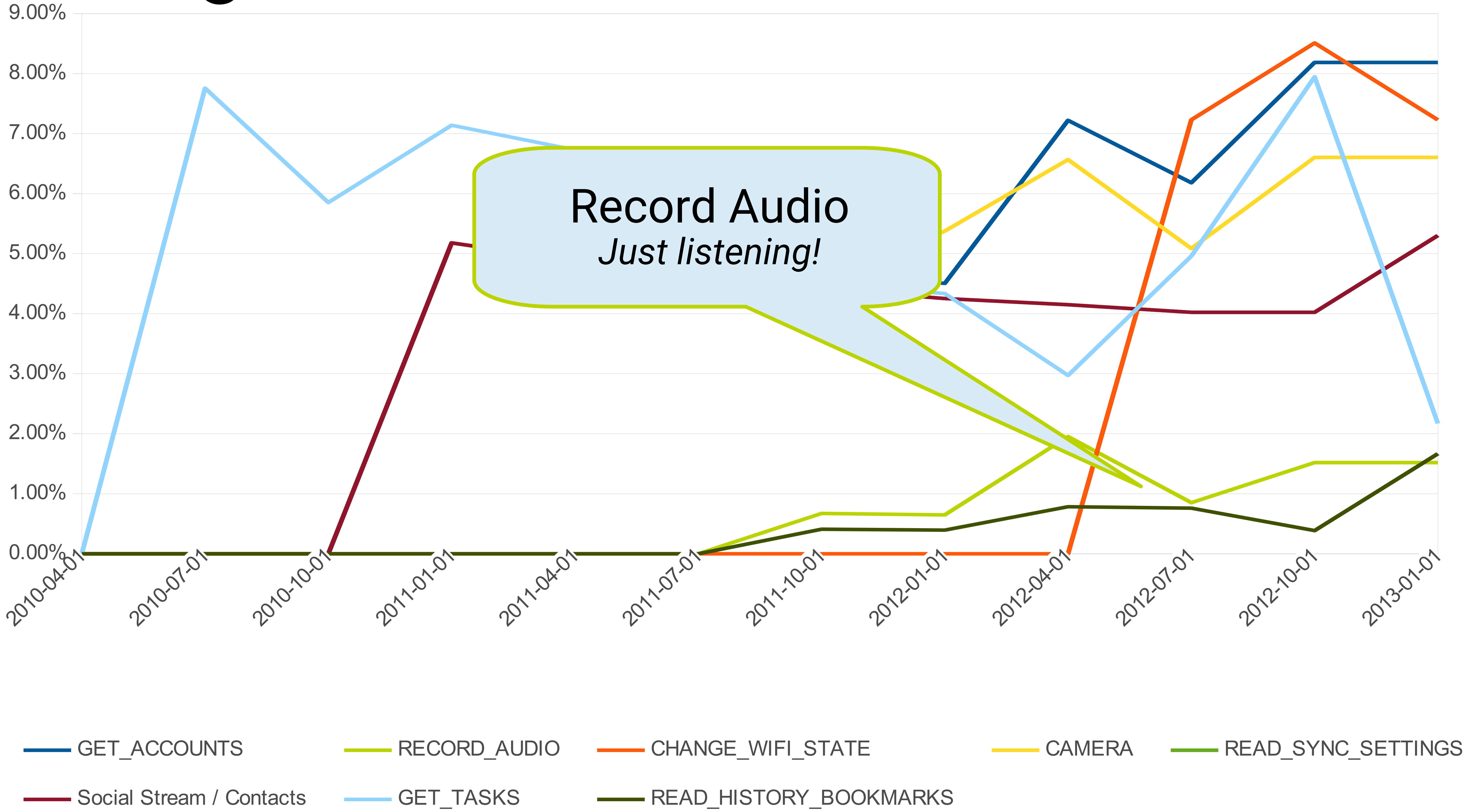
"Dangerous" Permissions



Change WiFi State

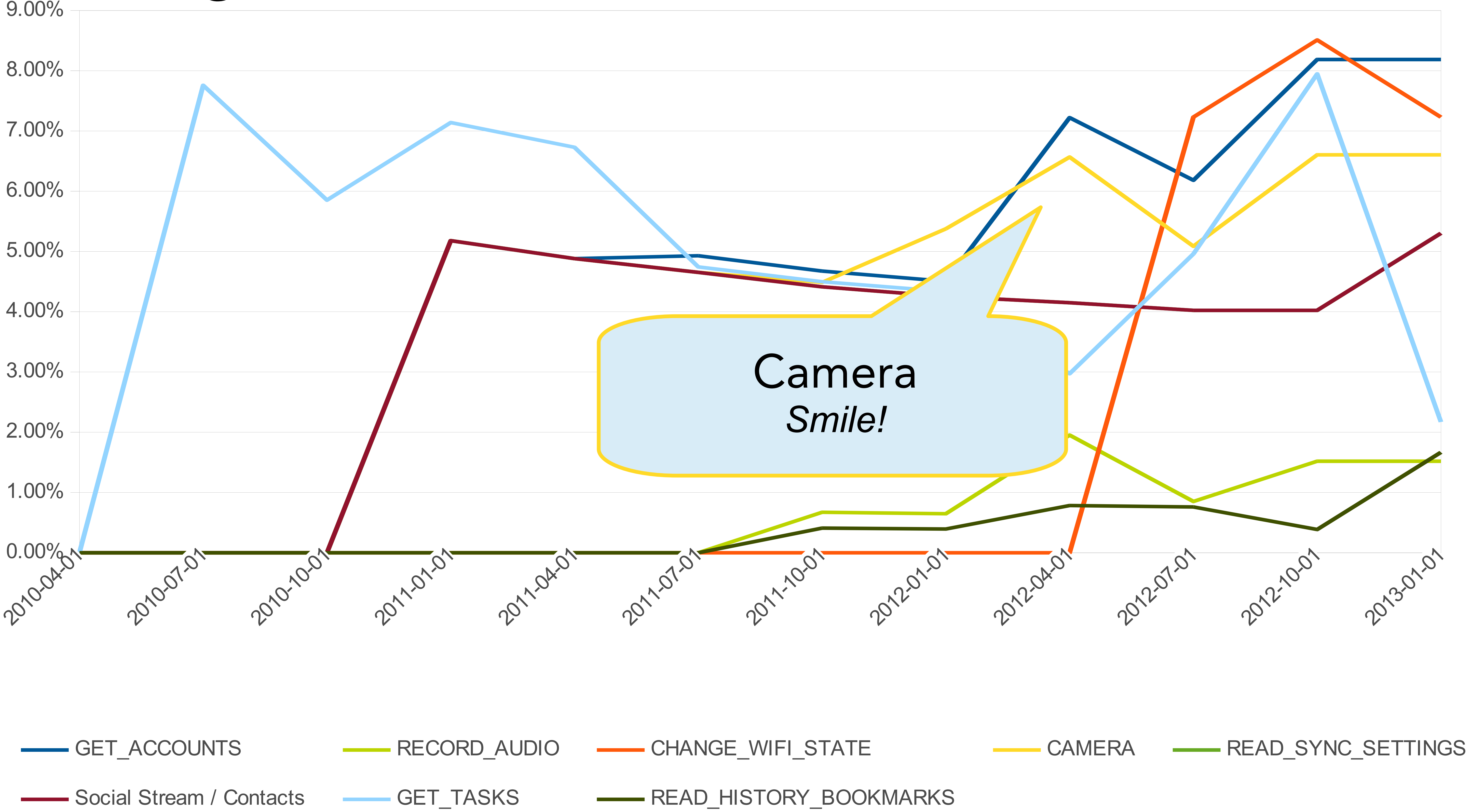
Load those video ads!

“Dangerous” Permissions



Record Audio
Just listening!

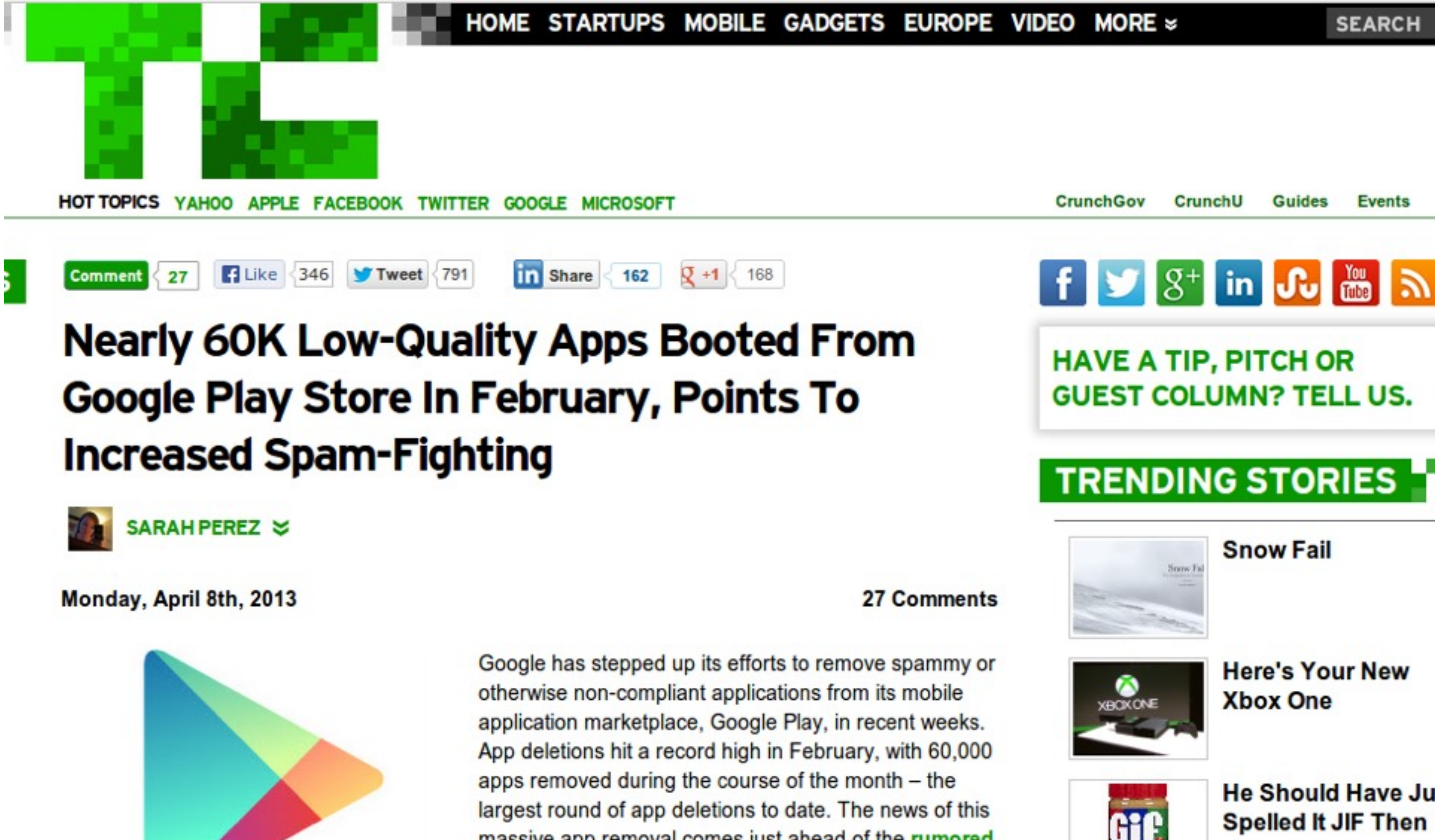
“Dangerous” Permissions



Camera

Smile!

The App Purge of 2013


HOME STARTUPS MOBILE GADGETS EUROPE VIDEO MORE
SEARCH
HOT TOPICS YAHOO APPLE FACEBOOK TWITTER GOOGLE MICROSOFT
CrunchGov CrunchU Guides Events
Comment 27
Like 346
Tweet 791
Share 162
+1 168
HAVE A TIP, PITCH OR GUEST COLUMN? TELL US.
TRENDING STORIES
Snow Fail
Here's Your New Xbox One
He Should Have Ju
Spelled It JIF Then

Nearly 60K Low-Quality Apps Booted From Google Play Store In February, Points To Increased Spam-Fighting

SARAH PEREZ

Monday, April 8th, 2013

27 Comments

Google has stepped up its efforts to remove spammy or otherwise non-compliant applications from its mobile application marketplace, Google Play, in recent weeks. App deletions hit a record high in February, with 60,000 apps removed during the course of the month – the largest round of app deletions to date. The news of this massive app removal comes just ahead of the rumored

What did Google do?

Resampled our apps from Google Play

Analyzed “missing” apps

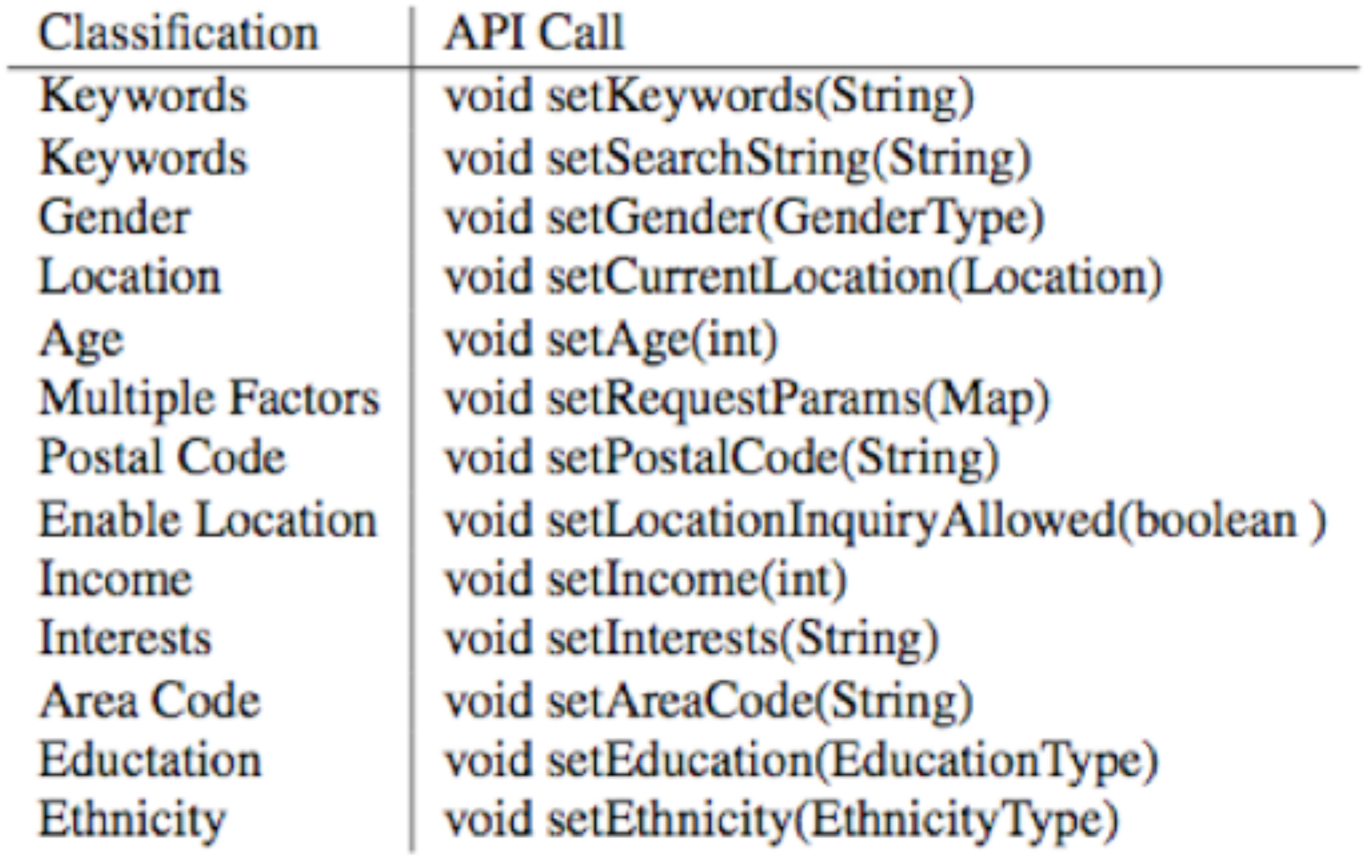
Apps with certain libraries tended to disappear

Google's action vs. ad library

Ad Library	Percent of Apps Removed
EverBadge	60.5%
Hunt Mobile	45.5%
AirPush	40.7%
SendDroid	31.2%
Waps	29.7%
TapIt	28.4%
Average	*11.6%*

Are apps leaking sensitive data?

Ad libraries have sensitive APIs

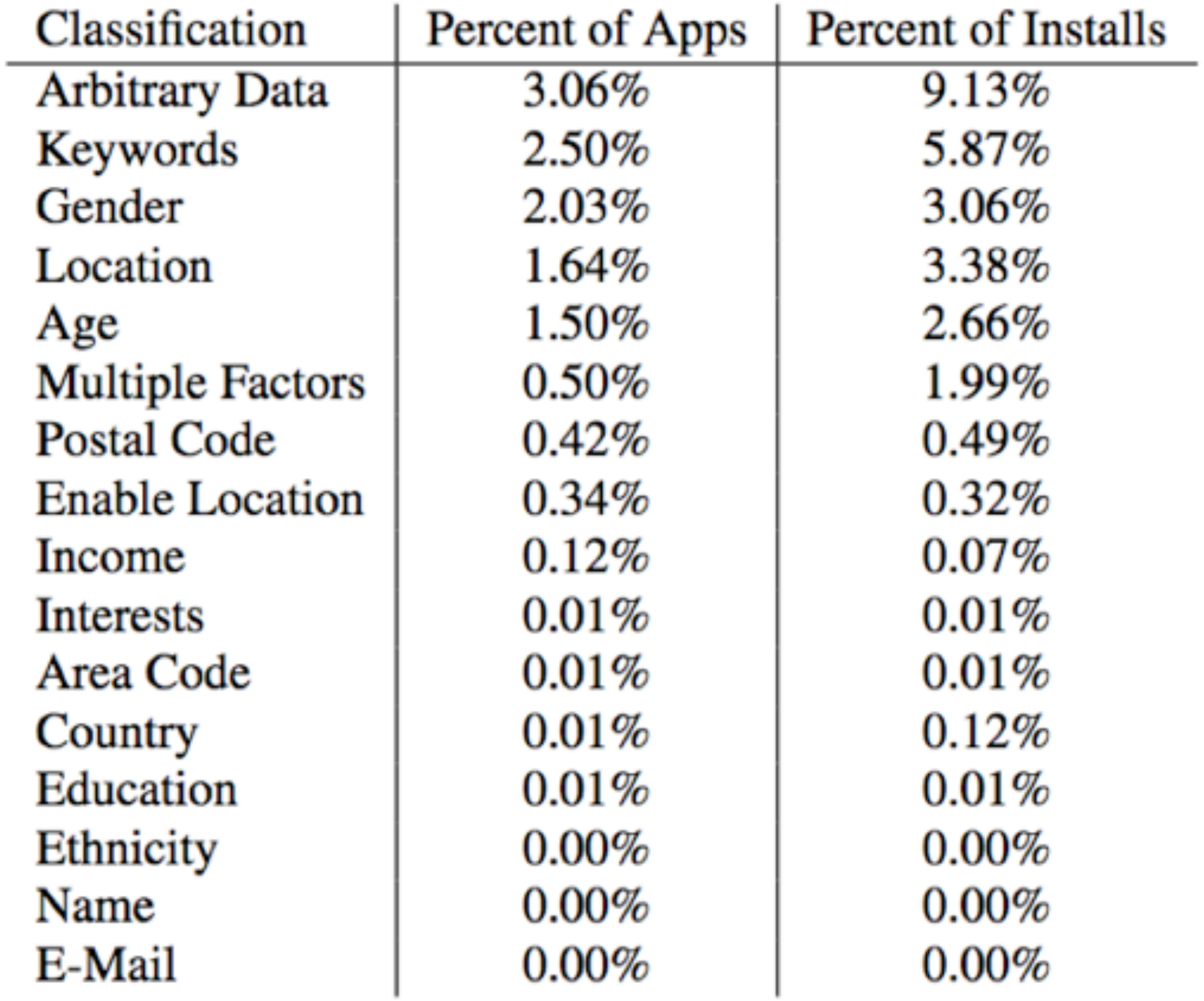


Classification	API Call
Keywords	void setKeywords(String)
Keywords	void setSearchString(String)
Gender	void setGender(GenderType)
Location	void setCurrentLocation(Location)
Age	void setAge(int)
Multiple Factors	void setRequestParams(Map)
Postal Code	void setPostalCode(String)
Enable Location	void setLocationInquiryAllowed(boolean)
Income	void setIncome(int)
Interests	void setInterests(String)
Area Code	void setAreaCode(String)
Eductation	void setEducation(EducationType)
Ethnicity	void setEthnicity(EthnicityType)

Example: API for InMobi

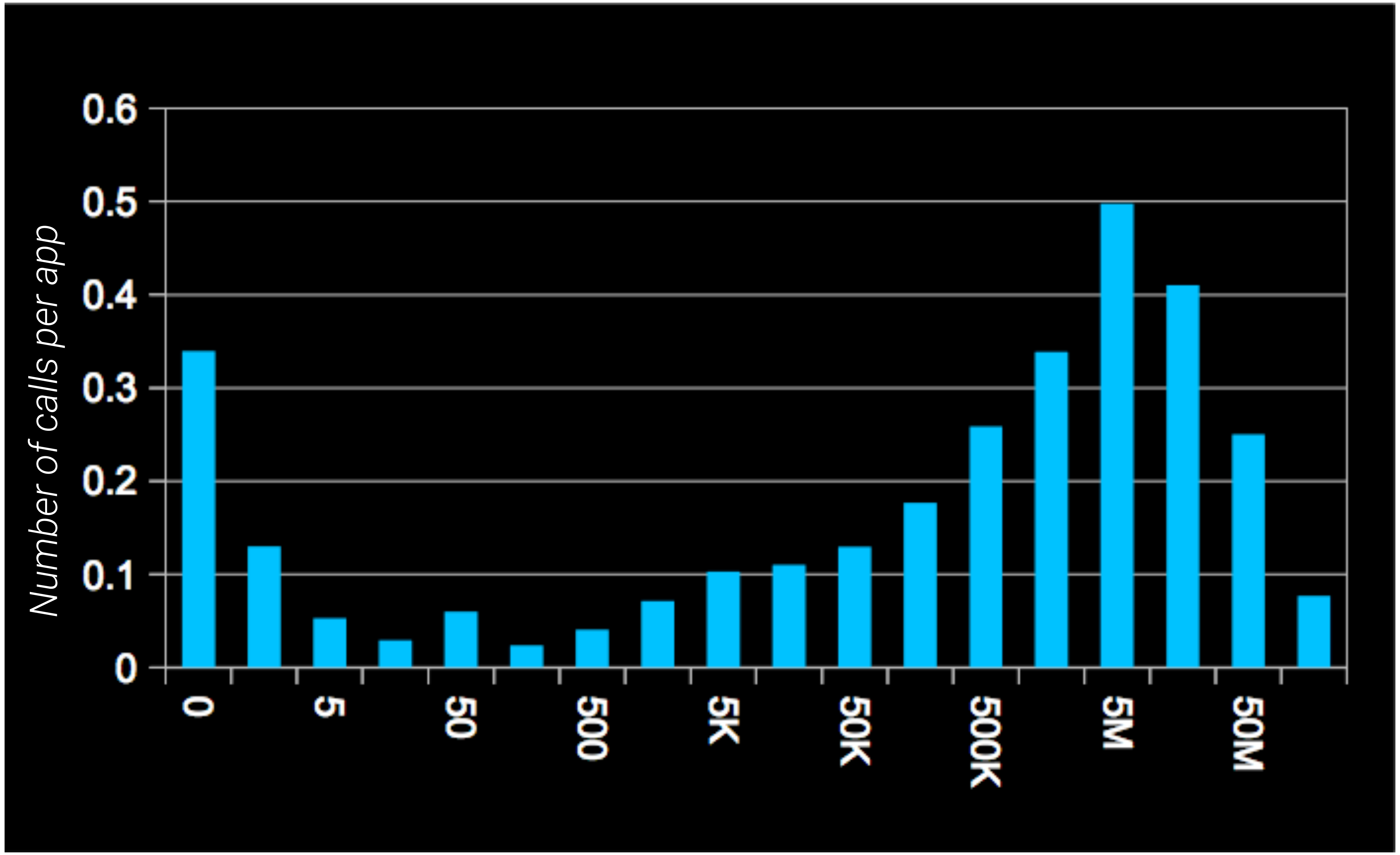
Goal: enumerate use of APIs in top-20 ad libraries from our corpus of Android apps

How often are these APIs used?

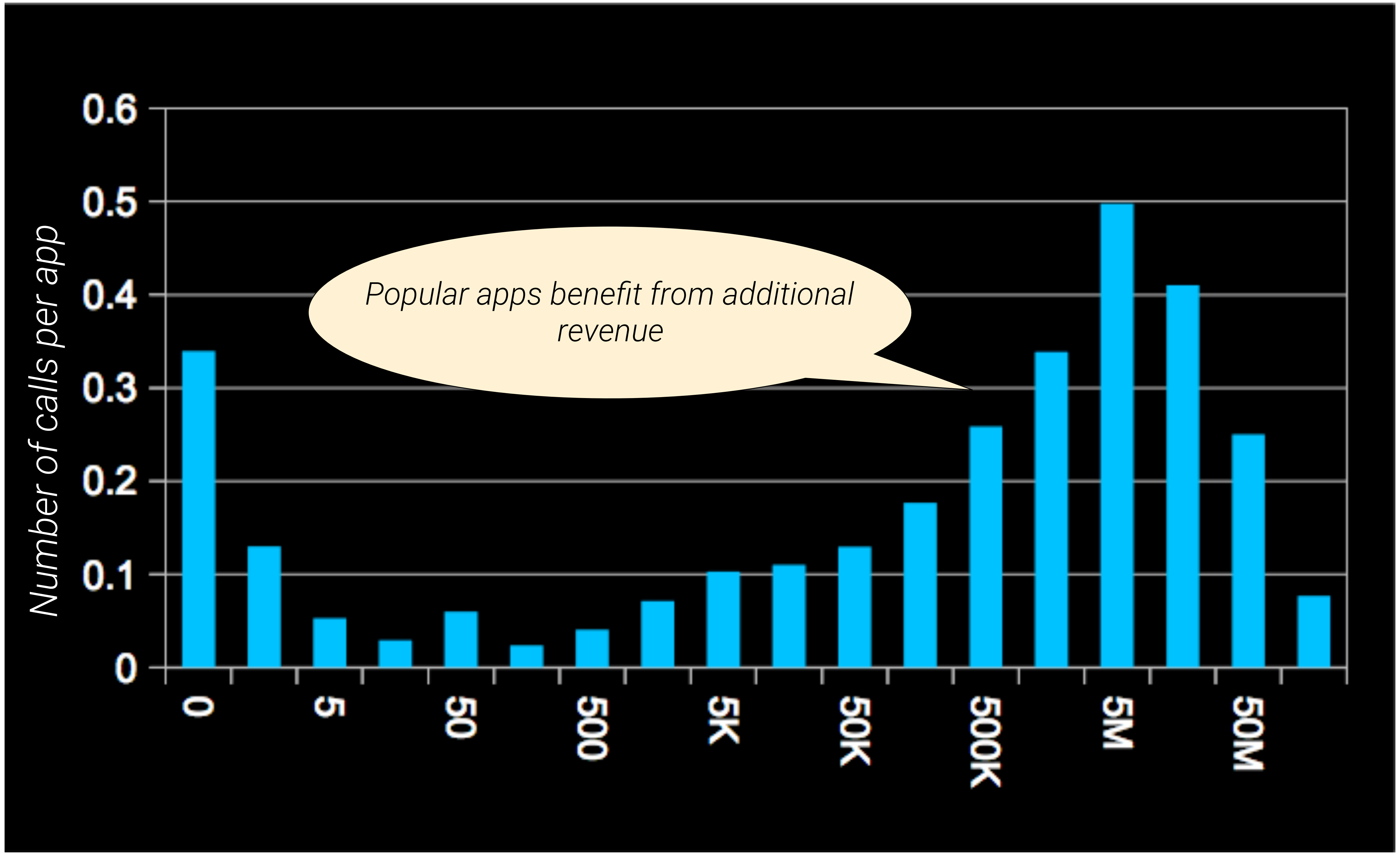


Classification	Percent of Apps	Percent of Installs
Arbitrary Data	3.06%	9.13%
Keywords	2.50%	5.87%
Gender	2.03%	3.06%
Location	1.64%	3.38%
Age	1.50%	2.66%
Multiple Factors	0.50%	1.99%
Postal Code	0.42%	0.49%
Enable Location	0.34%	0.32%
Income	0.12%	0.07%
Interests	0.01%	0.01%
Area Code	0.01%	0.01%
Country	0.01%	0.12%
Education	0.01%	0.01%
Ethnicity	0.00%	0.00%
Name	0.00%	0.00%
E-Mail	0.00%	0.00%

Calls vs. Install Count

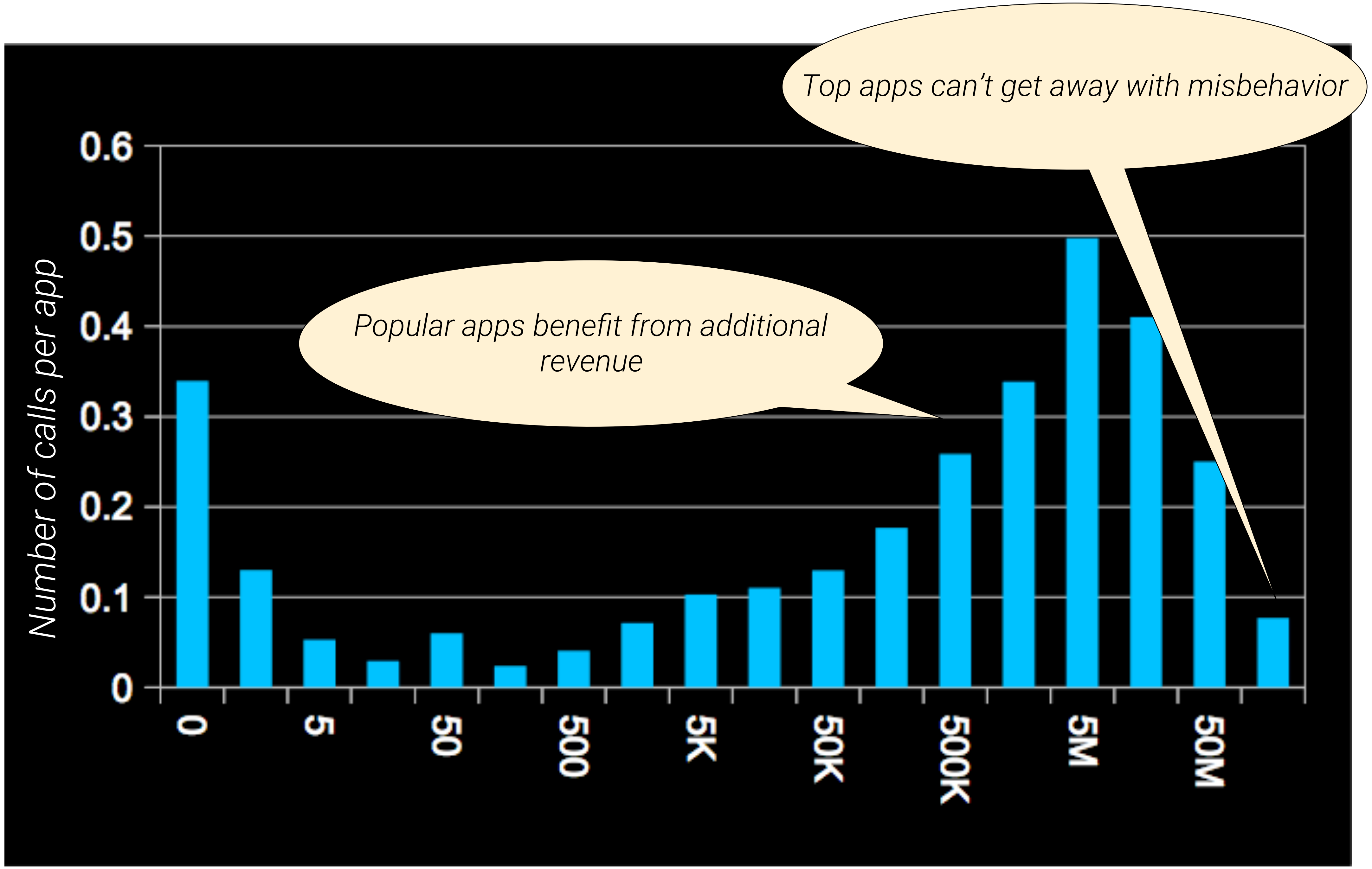


Calls vs. Install Count



Popular apps benefit from additional revenue

Calls vs. Install Count



Popular apps benefit from additional revenue

Top apps can't get away with misbehavior

Web Ad Security = Mashup Security

Web mashup security

Advertisements usually hosted in <*IFRAME*>

➔ Application code separation

Same origin policy restricts network access

➔ Harder for web page to forge clicks

Android has no equivalent mechanisms

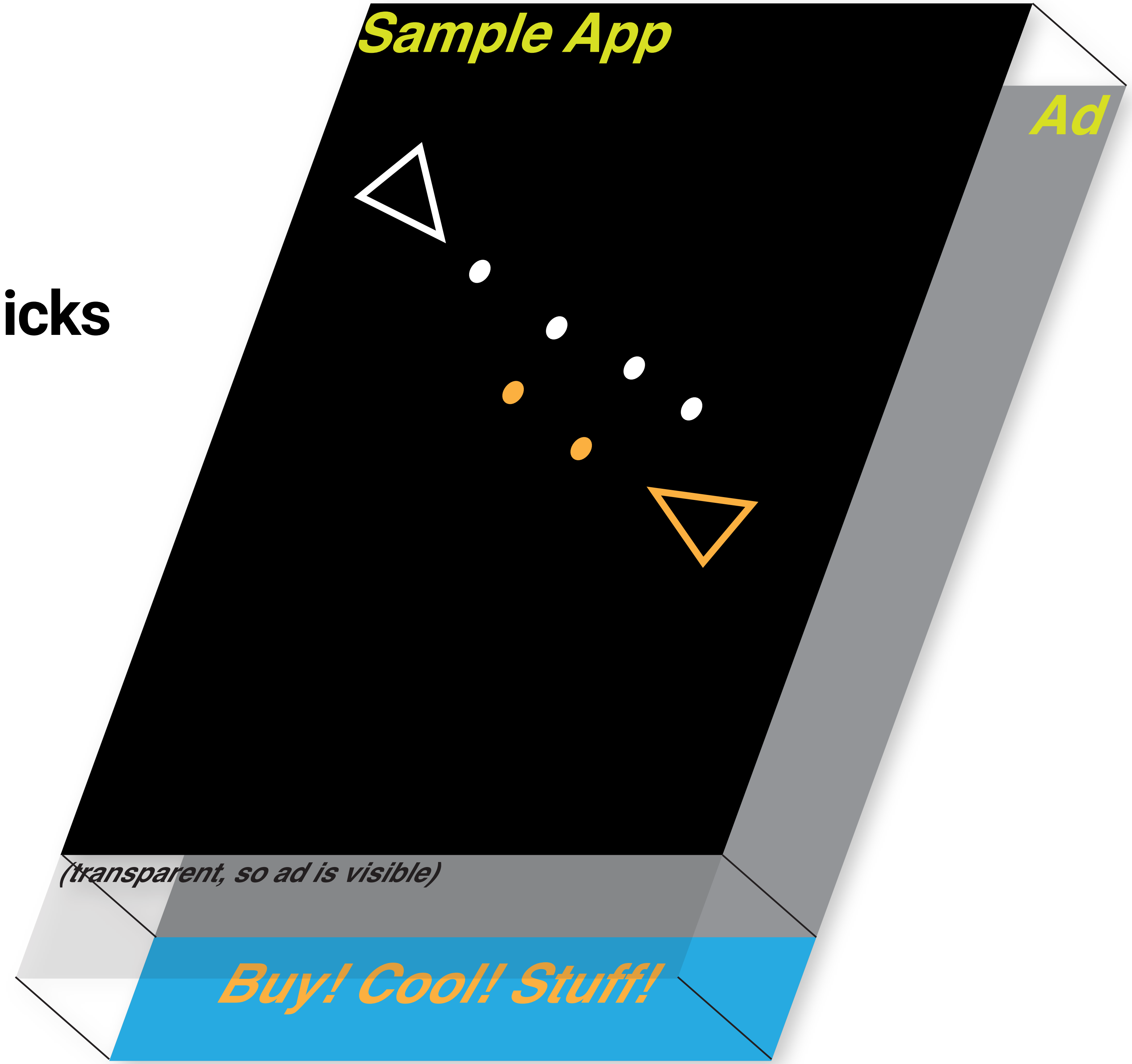
AdSplit Architecture

App layering/separation

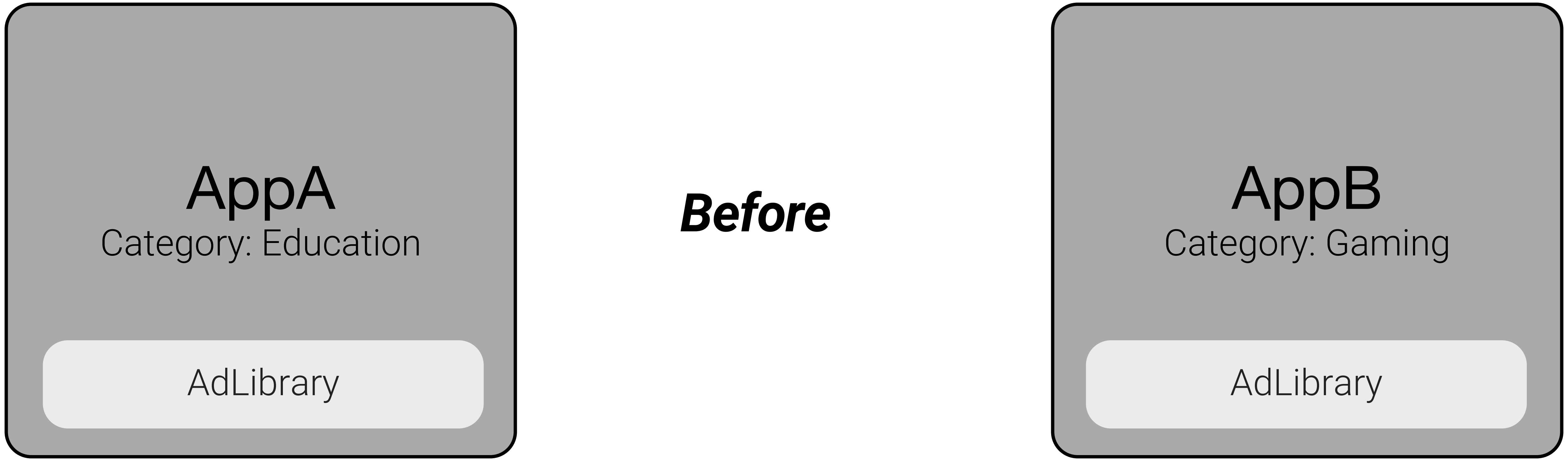
Click delegation

OS attaches verifiable statements to clicks

Ads verify their visibility



Process separation



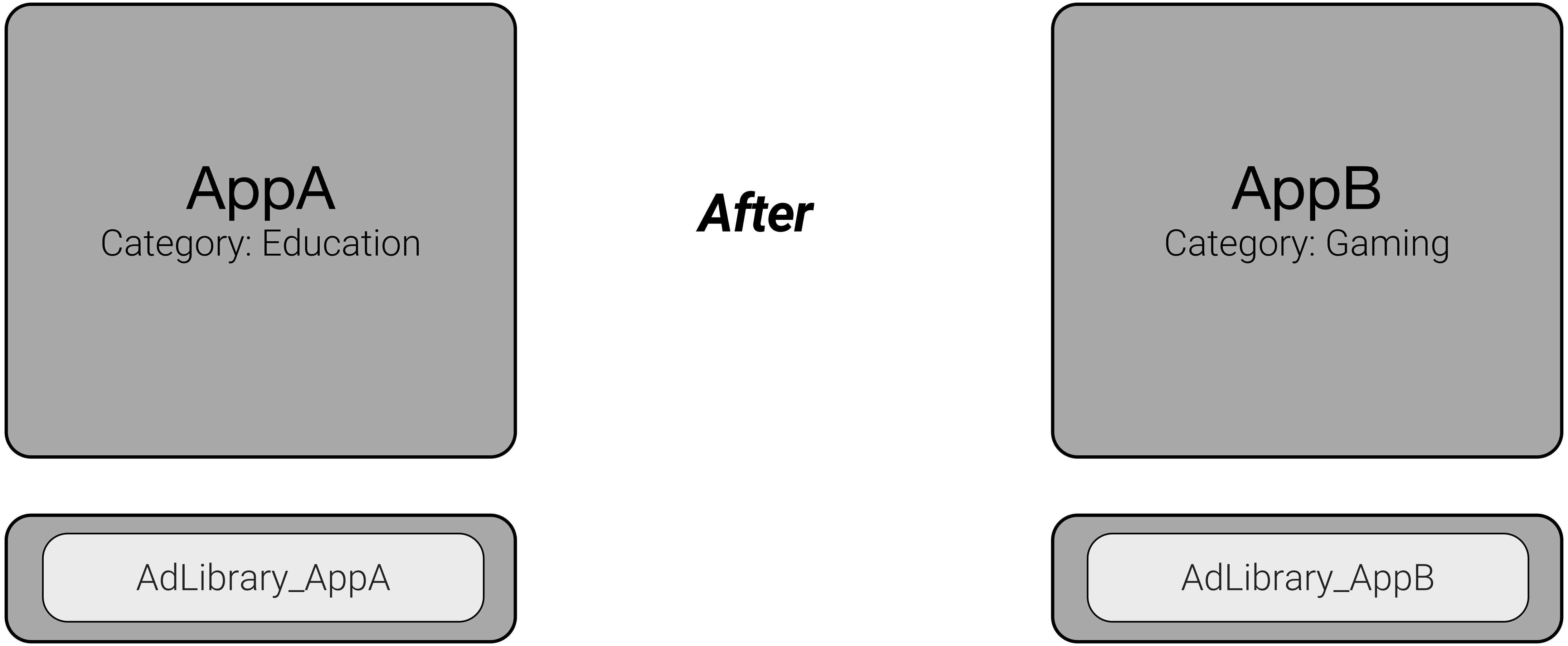
AppA
Category: Education
AdLibrary

Before

AppB
Category: Gaming
AdLibrary

AppA and AppB include same ad library but can see different ads based on category.

Process separation



A distinct instance of ad library per host app.

Lifecycle management

Launch ad activity with host activity

Change activity stack to insert and remove ad activity with host activity

User Input

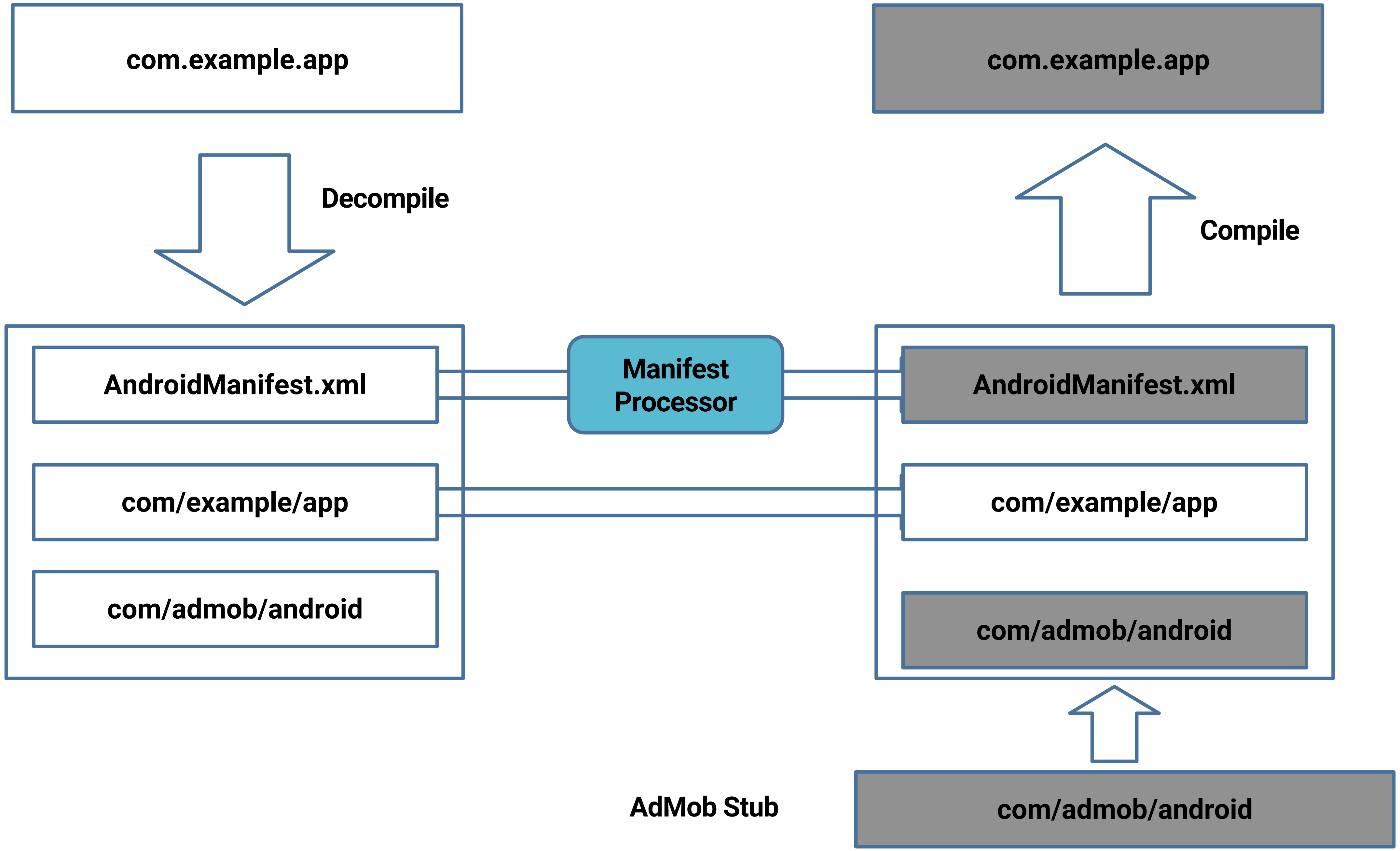
User input validation based on Quire [UsenixSec ’11]

HMAC-signed touch events with timestamps

Ad service can verify visibility & layout

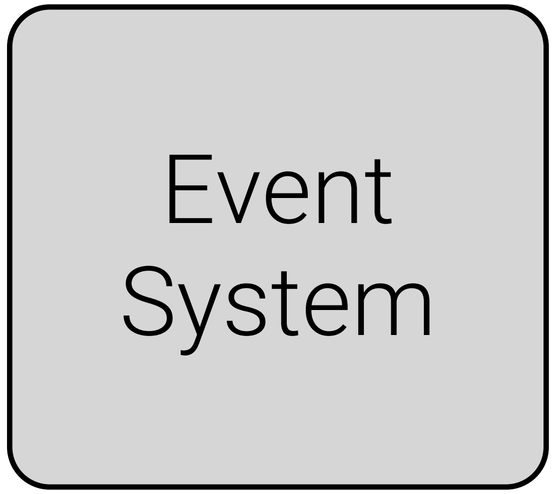
Supported by stock Android queries

Automated separation



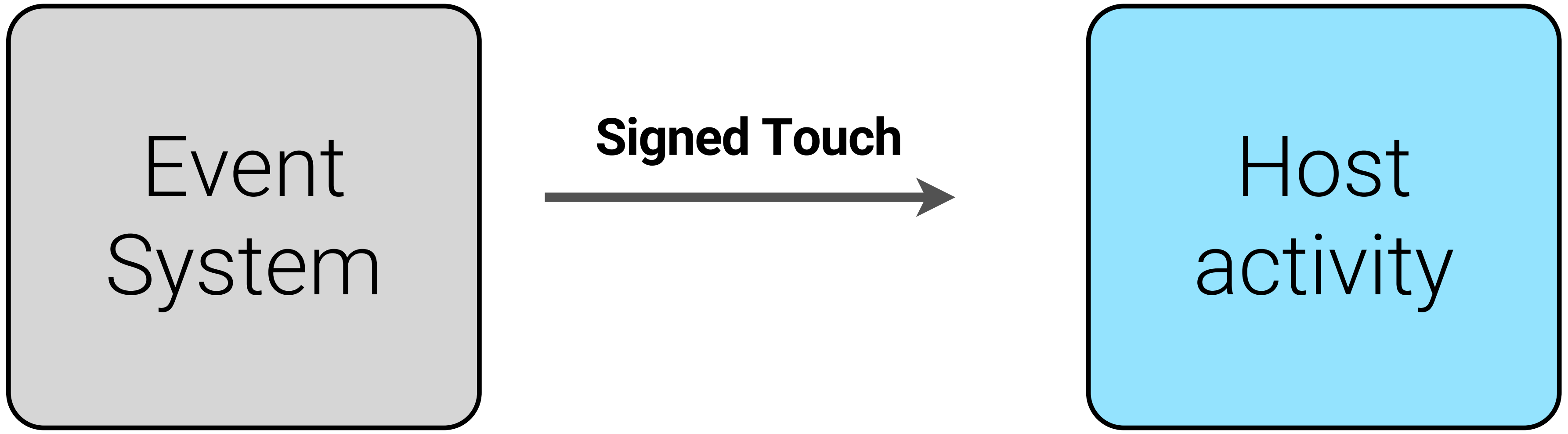
Touch event validation

Touch event validation

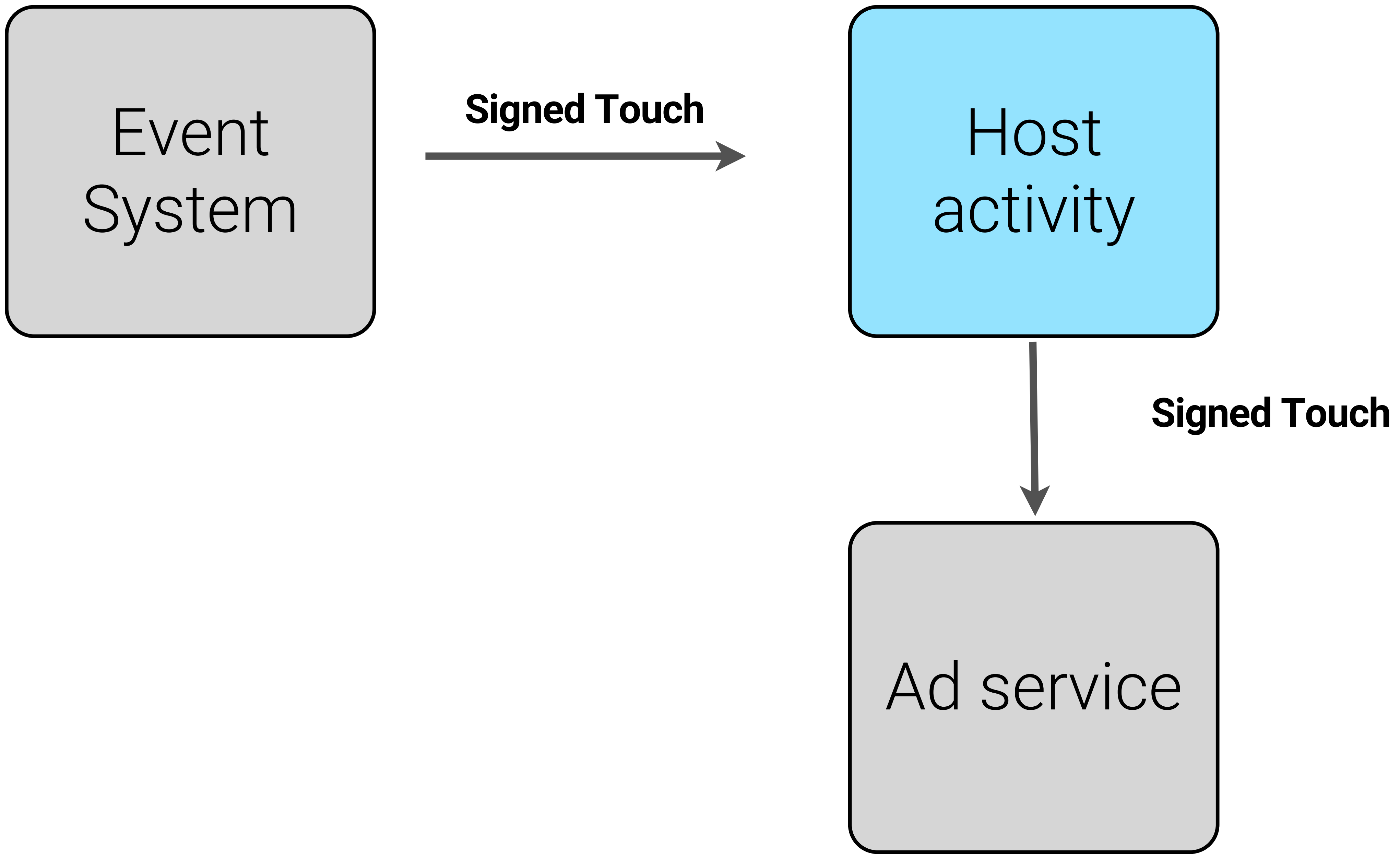


Event System

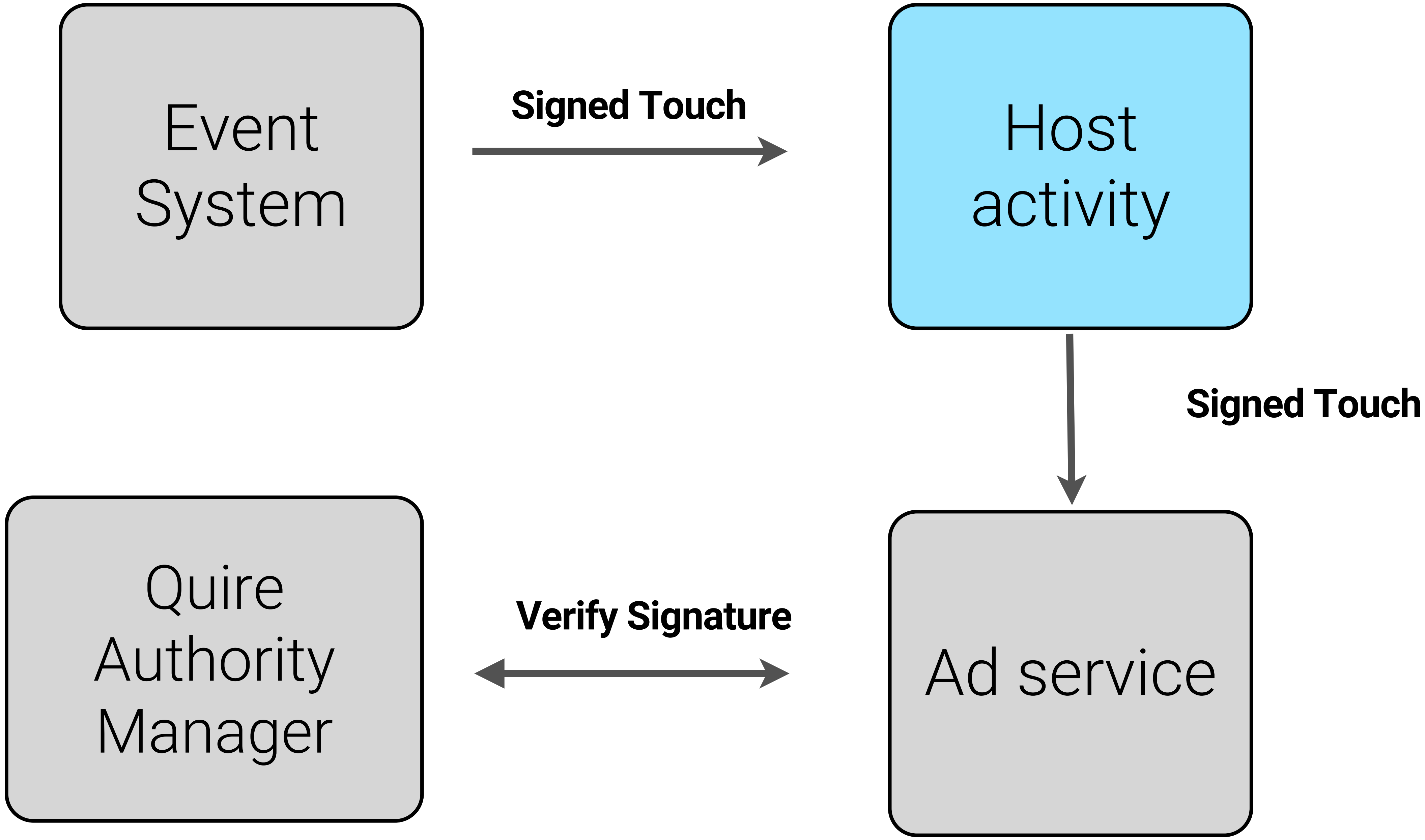
Touch event validation



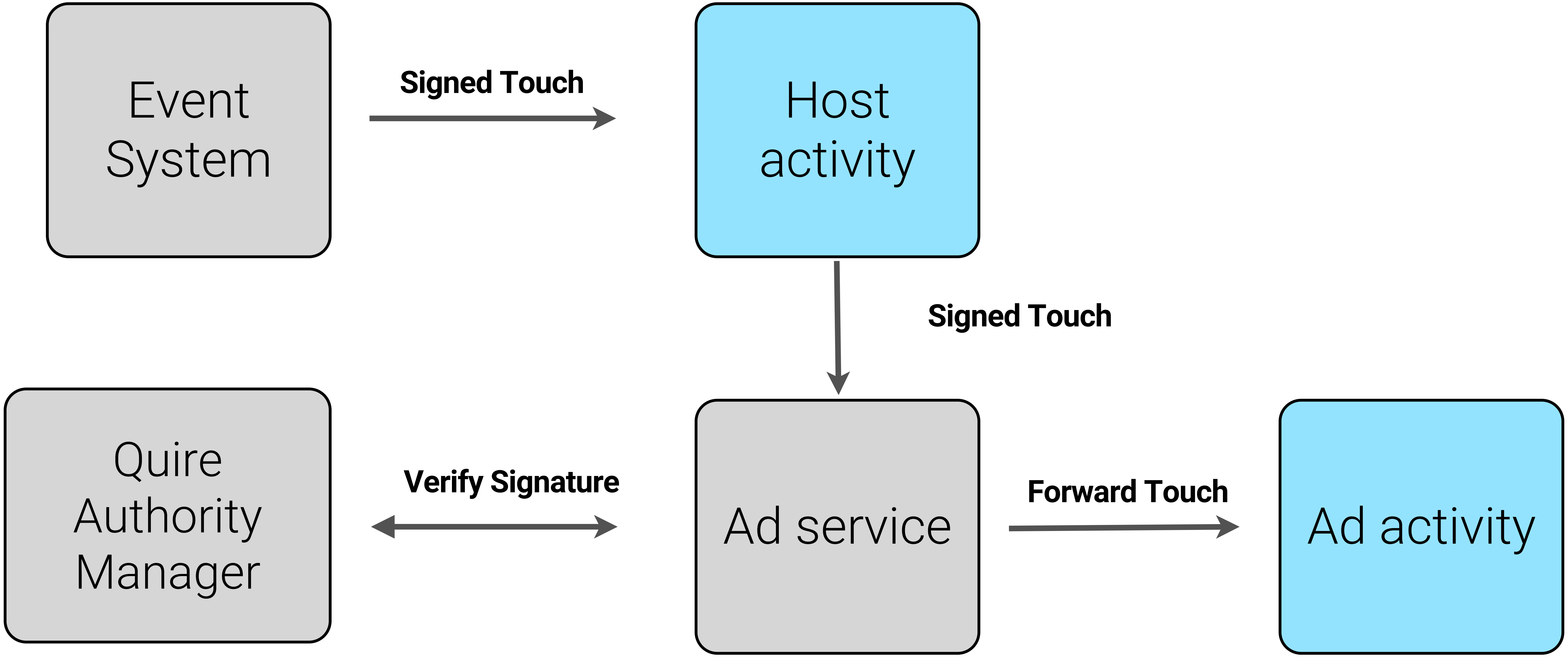
Touch event validation



Touch event validation



Touch event validation



Event System

Signed Touch

Host activity

Signed Touch

Quire Authority Manager

Verify Signature

Ad service

Forward Touch

Ad activity

Why not automatic?

Need to re-engineer for every ad library

Missed opportunity to have verifiable provenance

Local crypto service could sign event messages from ads

Remote server could verify event authenticity, deter click fraud

(Details: see our *Quire* paper in Usenix Security 2011)

Insight: Ads use HTML!

Most ad libraries embed an HTML WebView widget

Advertisers like HTML + JavaScript (portability, etc.)

System-provided ad widget could be a substitute!

AdWebView Benefits

No advertising native-code installation required

Browser logic enforces same-origin privileges

No permissions required!

Ads still run in a separate activity, separate UID

Defense in depth

Policy questions

Sensitive privileges (geolocation, etc.)?

Host apps might leak sensitive data to ads

Ad blocking?

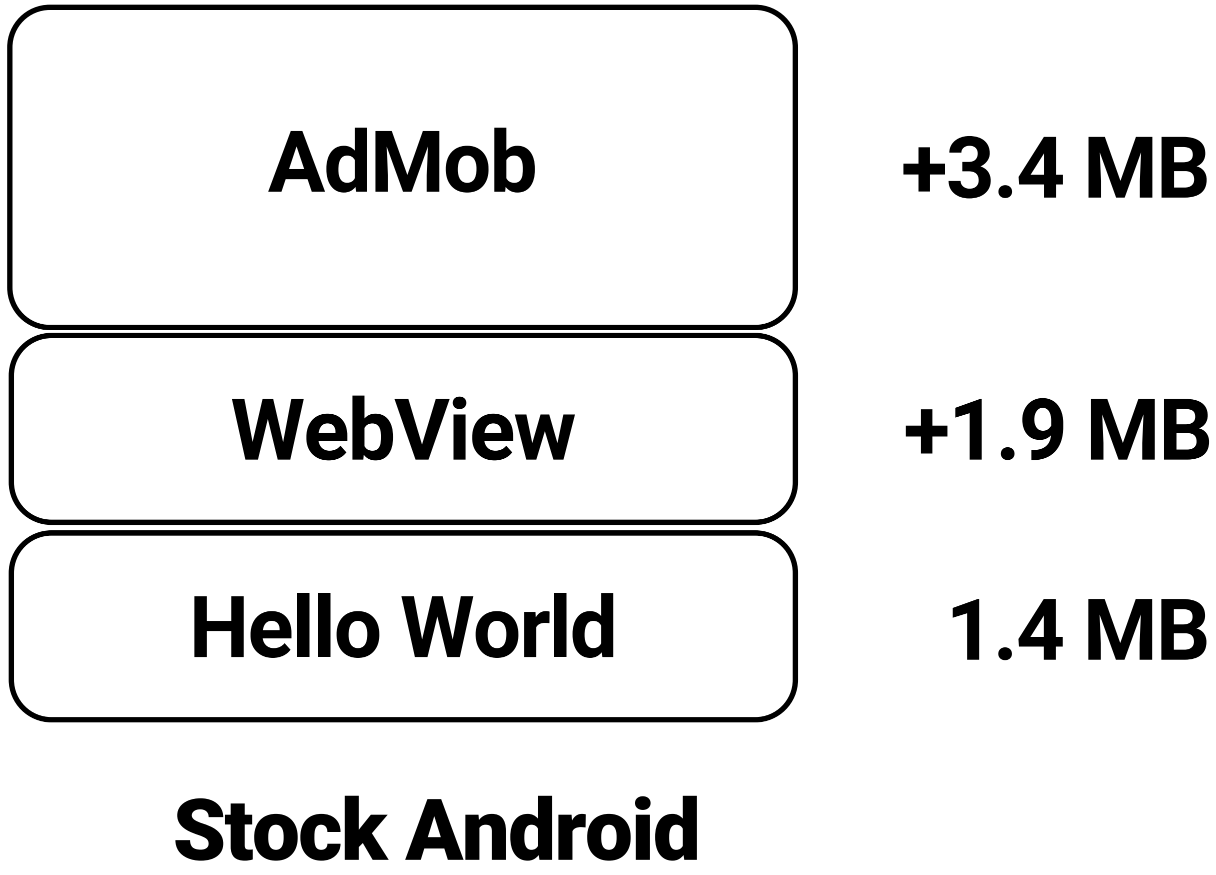
Power rationing?

Memory Overhead

Hello World 1.4 MB

Stock Android

Memory Overhead



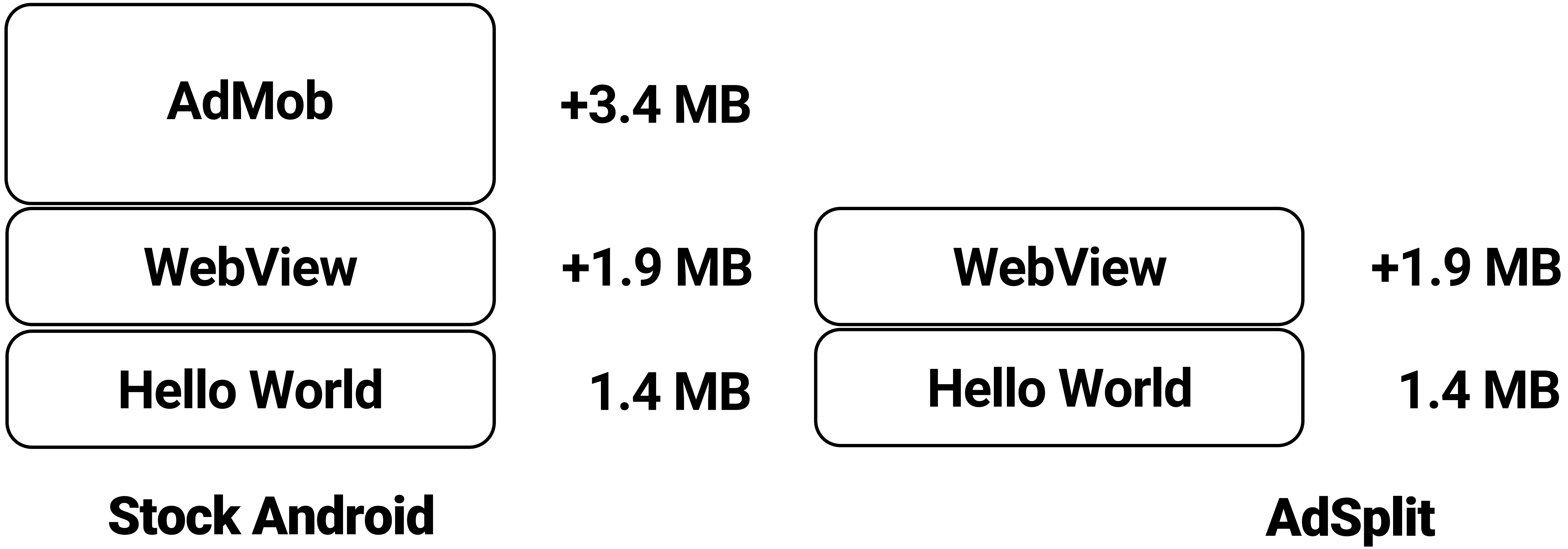
AdMob +3.4 MB

WebView +1.9 MB

Hello World 1.4 MB

Stock Android

Memory Overhead



AdMob +3.4 MB

WebView +1.9 MB

Hello World 1.4 MB

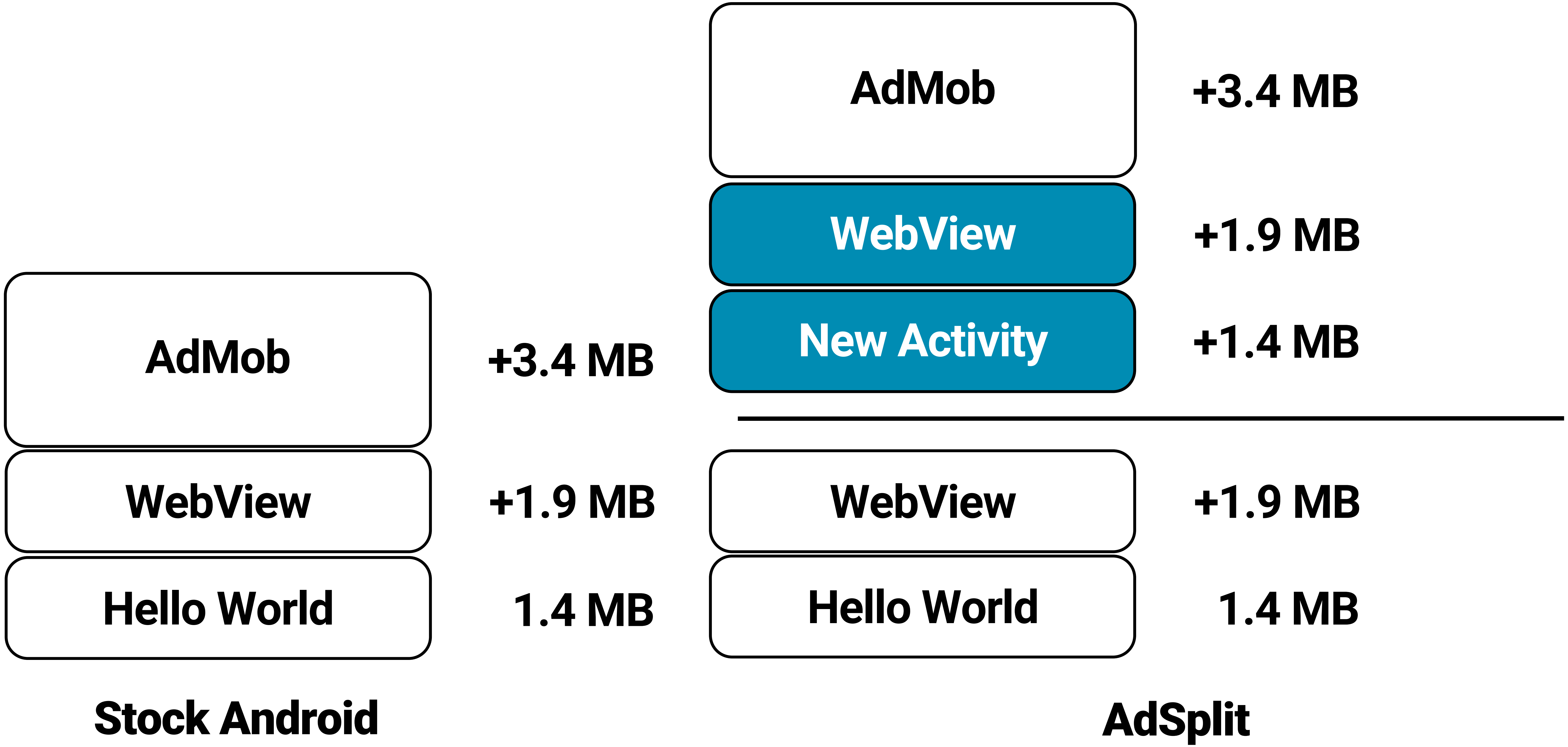
Stock Android

WebView +1.9 MB

Hello World 1.4 MB

AdSplit

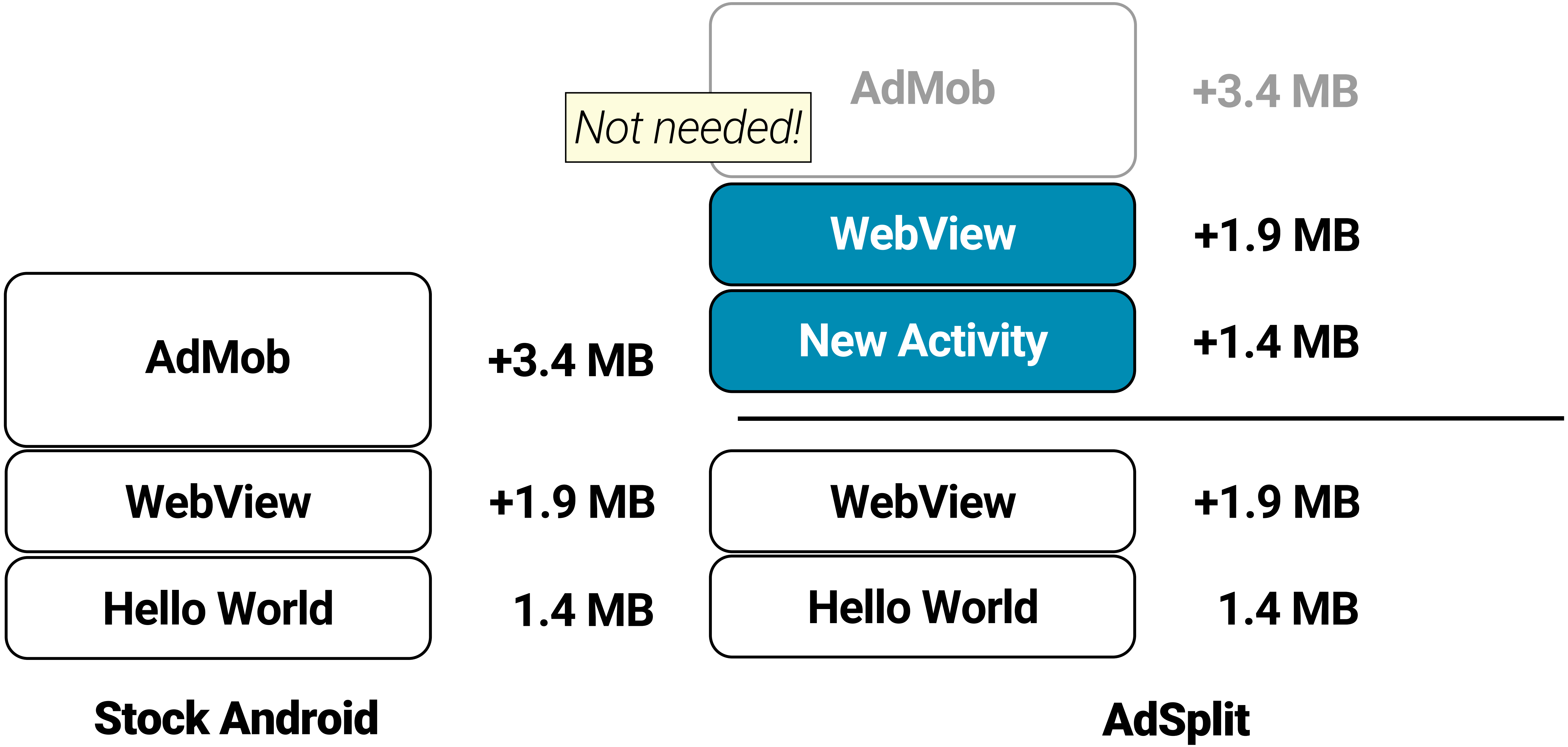
Memory Overhead



AdMob
+3.4 MB
WebView
+1.9 MB
New Activity
+1.4 MB
WebView
+1.9 MB
Hello World
1.4 MB
AdSplit

AdMob
+3.4 MB
WebView
+1.9 MB
Hello World
1.4 MB
Stock Android

Memory Overhead



AdMob +3.4 MB

WebView +1.9 MB

Hello World 1.4 MB

Stock Android

Not needed!

AdMob +3.4 MB

WebView +1.9 MB

New Activity +1.4 MB

WebView +1.9 MB

Hello World 1.4 MB

AdSplit

Aside: WebView security

Browsers need security updates

WebView is yet another WebKit derivative

Software updates broken out from core Android in 4.4

Google has abandoned backports for older versions

>50% of Android market no longer supported

Official advice: always use local sourced data or HTTPS

Advertising libraries use WebView

Generally over vanilla HTTP

AdMob notably encrypts the ***location*** field

Related Work

Confused Deputy Issues

Felt et al., **Permission Re-Delegation: Attacks and Defenses** (USENIX Security 2011)

Bugiel et al., **Towards Taming Privilege-Escalation Attacks on Android** (NDSS 2012)

Grace et al., **Systematic Detection of Capability Leaks in Stock Android Smartphones** (NDSS 2012)

Advertising

Pearce et al., **AdDroid: Privilege Separation for Applications and Advertisers in Android** (AsiaCCS 2012)

Grace et al., **Unsafe Exposure Analysis of Mobile In-App Advertisements** (WiSec 2012)

Leontiadis et al., **Don't kill my ads! Balancing Privacy in an Ad-Supported Mobile Application Market** (HotMobile 2012)

and lots more being published every day...

Other security topics

Nation-state attacks against your phone?

With the FBI asking Apple to produce custom signed firmware...

Could this be an issue with Android? Absolutely, but vendor-dependent.
Each vendor has their own release process, some will cooperate, others not.

The big unknown: the “baseband processor” and the ARM “TrustZone”
Separate operating systems, separate vulnerabilities.

Data-at-rest encryption / physical attacks

Standard support in Android 5.0, on by default in Android 6.0

Boot-time password requests, linked to decryption.

USB attacks?

Phone UI must be unlocked, user asked to approve computer’s public key.

“Boot locked” phone will refuse to install Android update that’s not Google-signed.

Boot unlocking will zero out the phone. Most users never do this.

USB-C allows for bidirectional data flow or even charging.

- UI support for charge-only.

Copy protection / DRM

Piracy is a huge issue

Apps can be reverse-engineered, tweaked, and redistributed.

Perhaps with added malware! Likely with added advertising.

APIs let you query the Play Store

Verify the user paid.

But don’t trust your local storage; user might have changed it.

Code obfuscators don’t seem to be a meaningful defense

ProGuard is a standard part of the Android SDK, very good at shrinking things.

Unsurprisingly, in-game-purchases and advertising are increasingly popular.

The future of Android?

Google is famously, amazingly secretive about whatever’s coming next.
Example: What about Java8 support? Dead silence.

The good news: Android’s market share is immense.
Massive tool support from industry & open source.
Example: You like Apple’s Swift? Try JetBrains’s Kotlin. Built into newer IntelliJ.

So, even if Google is unhelpful, there’s at least a huge dev community.
And some of their advice is occasionally helpful.