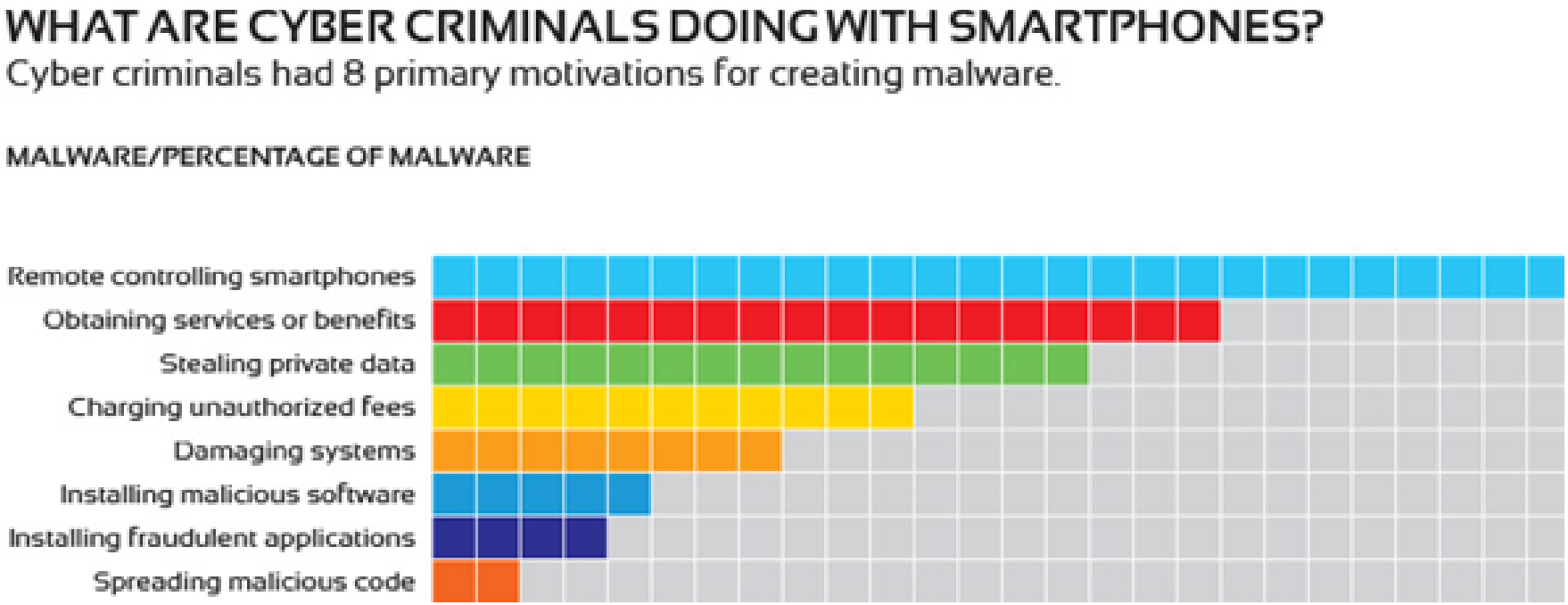
Mobile Malware

John Mitchell

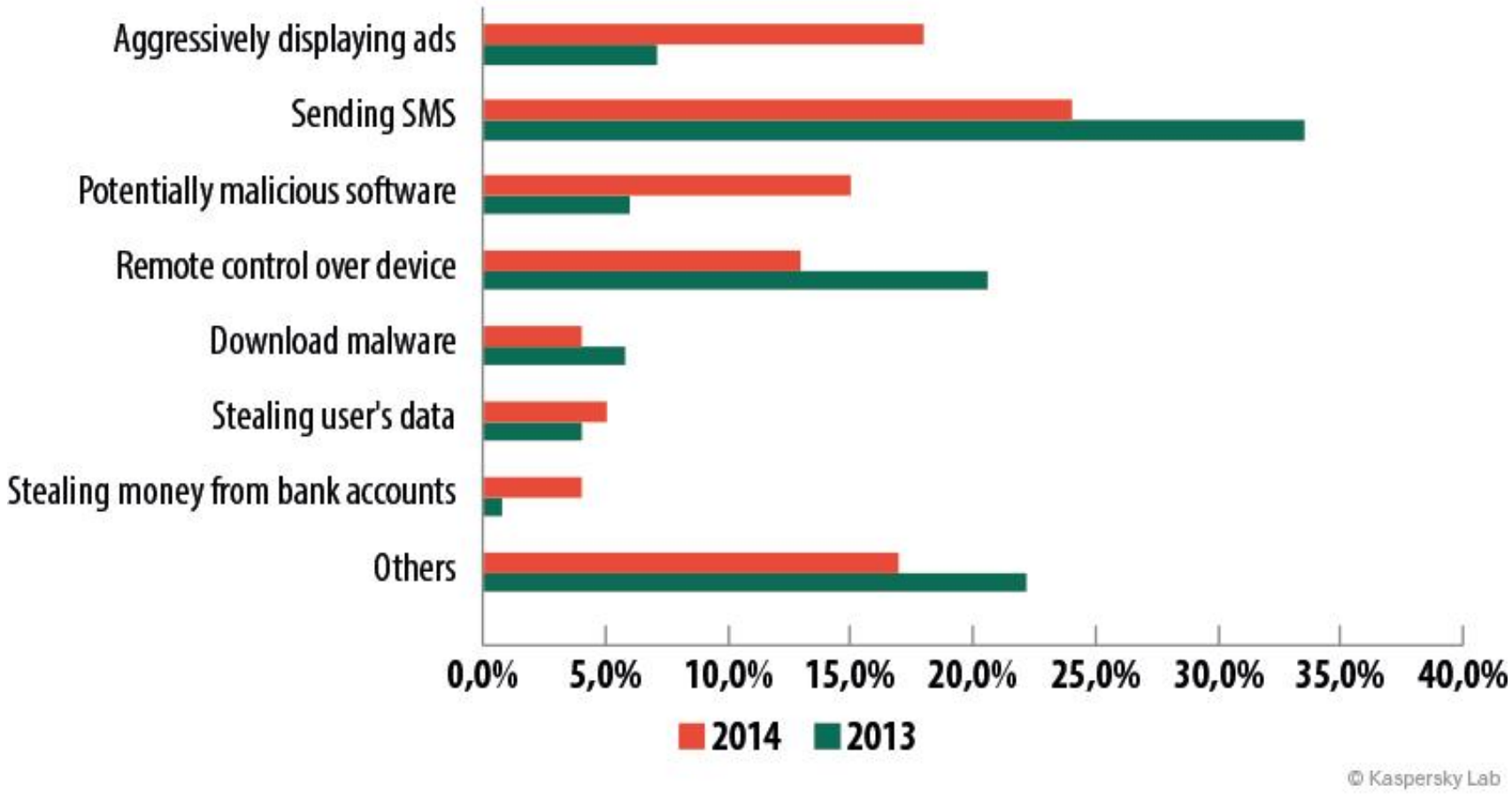
Outline

- Mobile malware
  - Common cases involve command and control, information theft
- Identifying malware
  - Detect at app store rather than on platform
- Classification study of mobile web apps
  - Entire Google Play market as of 2014
  - 85% of approx 1 million apps use web interface

Mobile Malware



Some Trends


Aggressively displaying ads
Sending SMS
Potentially malicious software
Remote control over device
Download malware
Stealing user's data
Stealing money from bank accounts
Others
0,0%
5,0%
10,0%
15,0%
20,0%
25,0%
30,0%
35,0%
40,0%
2014
2013
© Kaspersky Lab

iPhone: Operation Pawn Storm

blog.trendmicro.com/.../pawn-storm-update-ios-espionage-app-found/

- Trend Micro:
  - “an active economic and political cyber-espionage operation that targets … military, governments, defense industries, and the media.”
  - Infects individuals to get to organizations
- Xagent
  - iOS 7: app icon is hidden, runs in background, restarts if terminated
  - iOS 8: app icon is visible; doesn’t automatically restart
- Apparently, iOS device needs to be jailbroken
  - Exact install process unknown
  - May require social engineering.

XAgent app

- Collects user information
  - Collect text messages
  - Access contact lists, pictures, geo-location data
  - Start voice recording, read WiFi status
  - Get a list of installed apps, list of processes
- Command and Control (C&C) Communication
  - HTTP POST request to send messages
  - GET request to receive commands

Android malware example

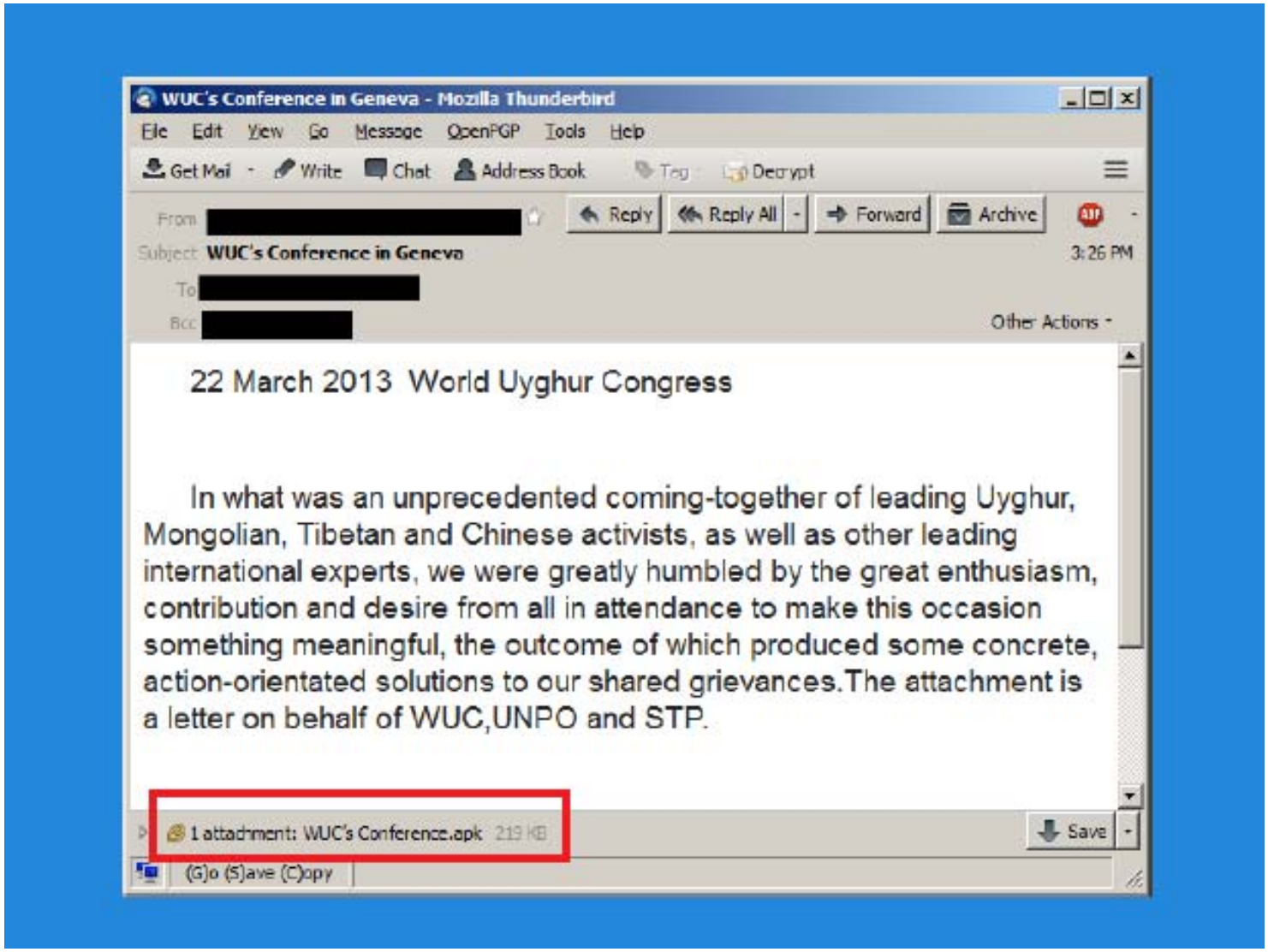
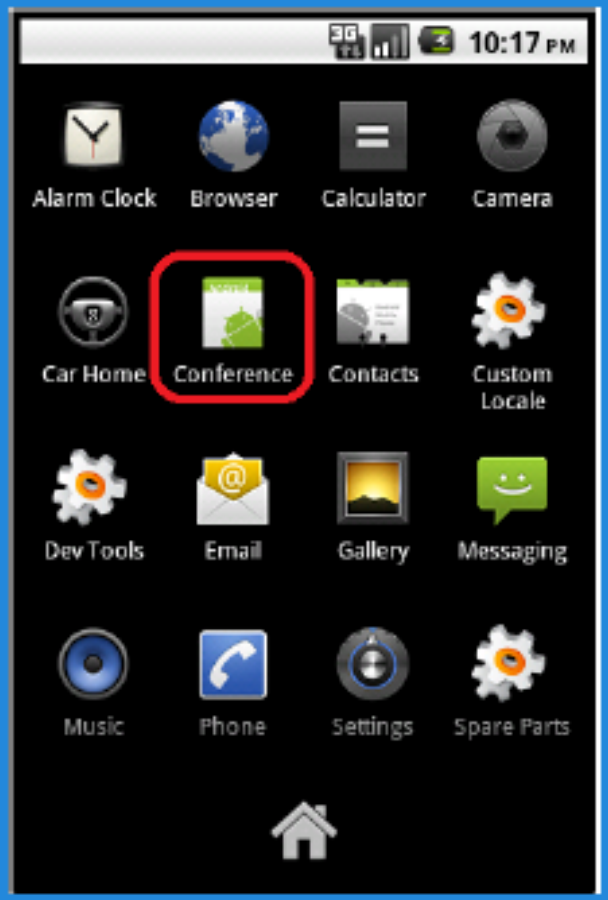


Image: Kaspersky Labs, https://www.securelist.com/en/blog/208194186/Android_Trojan_Found_in_Targeted_Attack, March 26th, 2013

Install malicious “conference app”



WUC’s Conference in Geneva

On behalf of all at the Word Uyghur Congress (WUC), the Unrepresented Nations and Peoples Organization (UNPO) and the Society for Threatened Peoples (STP), Human Rights in China: Implications for East Turkestan, Tibet and Southern Mongolia In what was an unprecedented

Image: Kaspersky Labs, https://www.securelist.com/en/blog/208194186/Android_Trojan_Found_in_Targeted_Attack, March 26th, 2013

Malware behavior triggered by C&C server (Chuli)

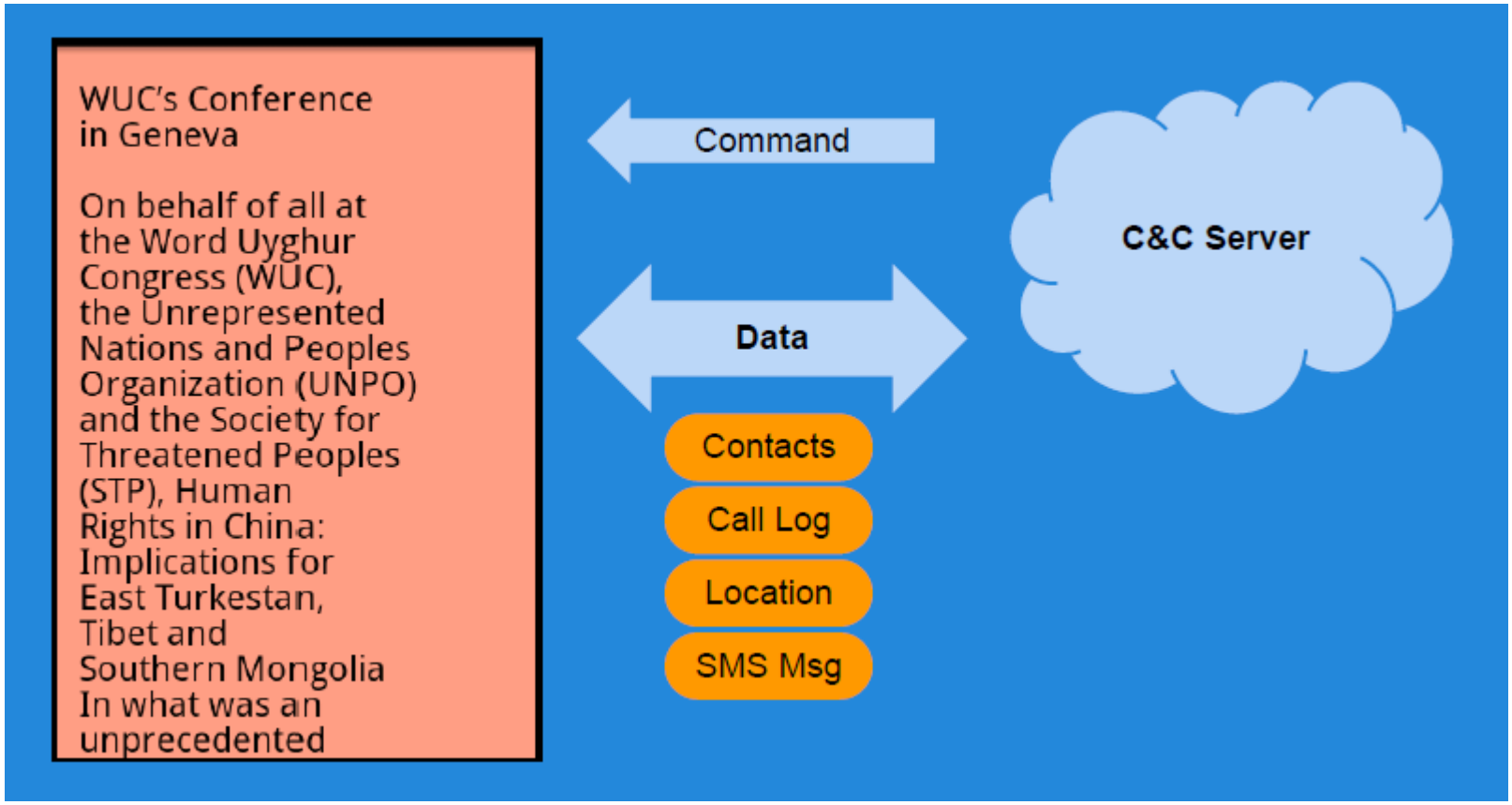


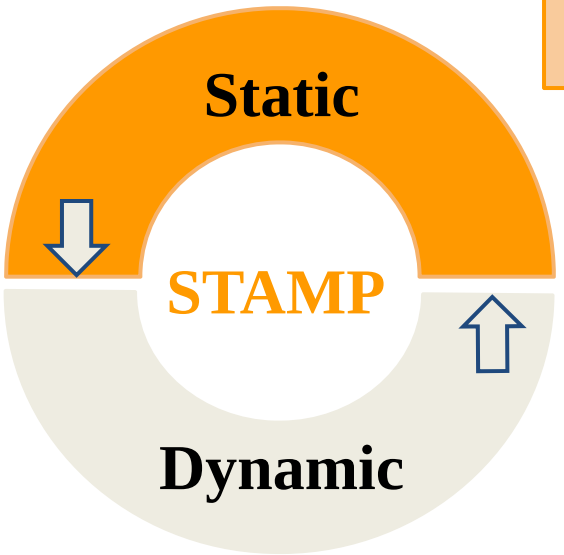
Image: Kaspersky Labs, https://www.securelist.com/en/blog/208194186/Android_Trojan_Found_in_Targeted_Attack, March 26th, 2013

Outline

- Mobile malware
  - Common cases involve command and control, information theft
- ➡ Identifying malware
  - Detect at app store rather than on platform
- Classification study of mobile web apps
  - Entire Google Play market as of 2014
  - 85% of approx 1 million apps use web interface

STAMP Admission System

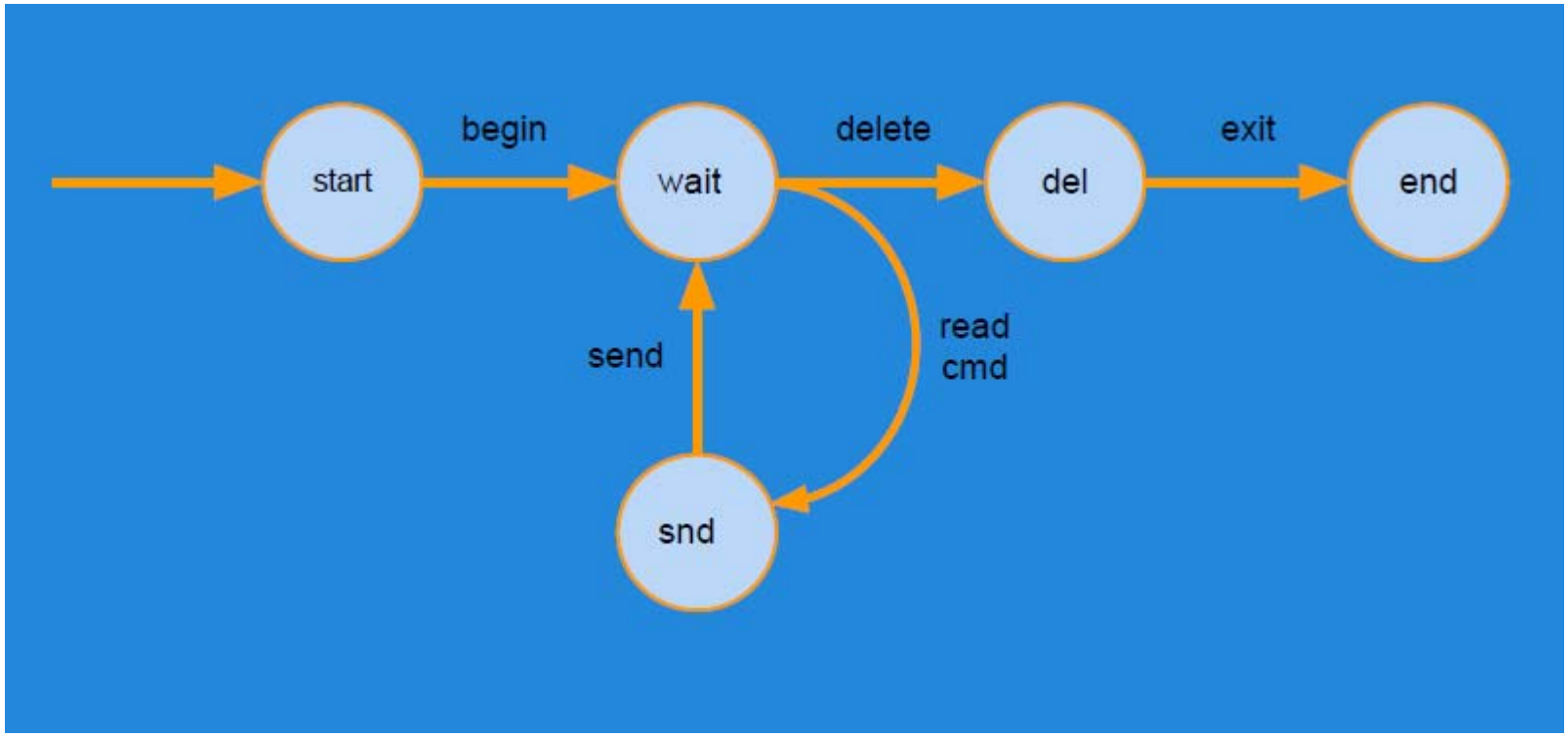
Static Analysis
More behaviors,
fewer details



Dynamic Analysis
Fewer behaviors,
more details

Alex Aiken,
John Mitchell,
Saswat Anand,
Jason Franklin
Osbert Bastani,
Lazaro Clapp,
Patrick Mutchler,
Manolis Papadakis

Abstract program execution



- States: mapping of variable names to values
- Transitions: relation on pairs of states
- Traces: sequence of states or state, transition pairs

Analysis

Step 1

Convert bytecode to intermediate format (called Quads)

Step 2

Compute call graph using Class Hierarchy Analysis

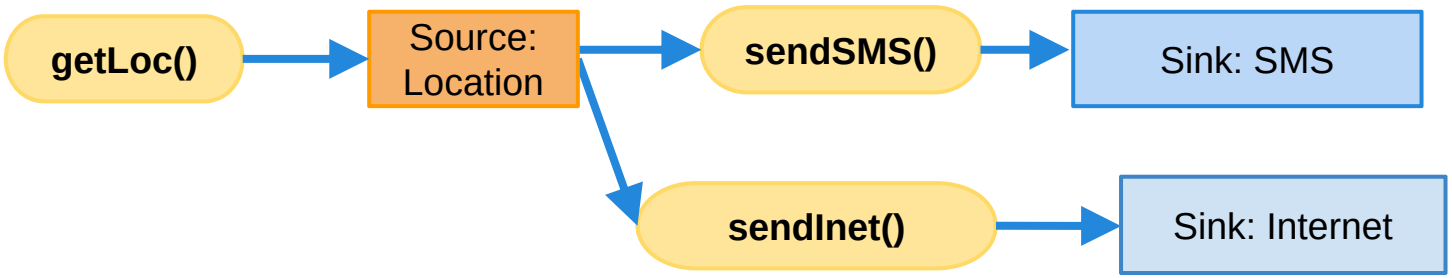
Step 3

Build an edge-labeled graph G by processing Quads of each class

Step 4

Add new edges to G as per a set of rules until no rules apply

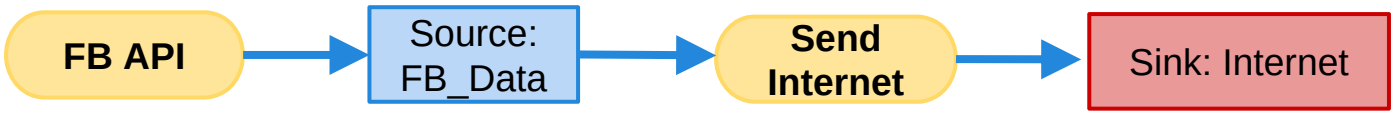
Data Flow Analysis



- Source-to-sink flows
  - Sources: Location, Calendar, Contacts, Device ID etc.
  - Sinks: Internet, SMS, Disk, etc.

Data Flow Analysis in Action

- Malware/Greyware Analysis
  - Data flow summaries enable enterprise-specific policies
- API Misuse and Data Theft Detection



- Automatic Generation of App Privacy Policies
  - Avoid liability, protect consumer privacy

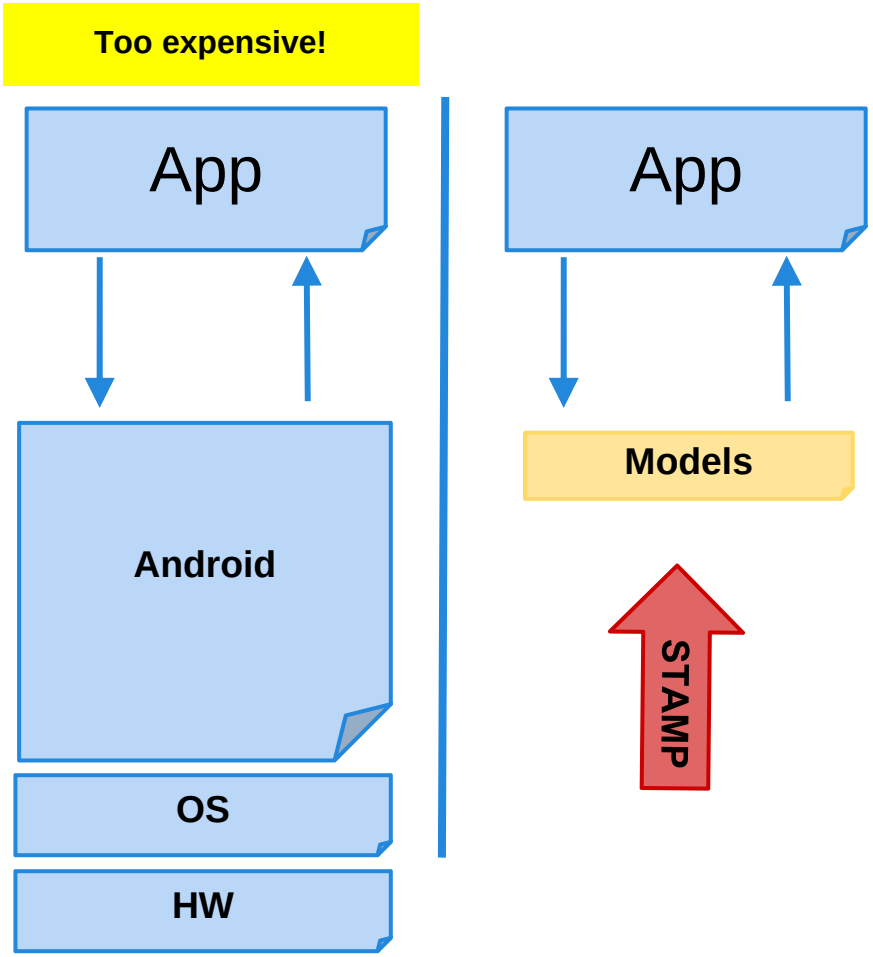
Privacy Policy
This app collects your:
Contacts
Phone Number
Address

- Vulnerability Discovery

Challenges

- Android is 3.4M+ lines of complex code
  - Uses reflection, callbacks, native code
- **Scalability:** Whole system analysis impractical
- **Soundness:** Avoid missing flows
- **Precision:** Minimize false positives

STAMP Approach



- Model Android/Java
  - Sources and sinks
  - Data structures
  - Callbacks
  - 500+ models
- Whole-program analysis
  - Context sensitive

Building Models

- 30k+ methods in Java/Android API
  - 5 mins x 30k = 2500 hours

- Follow the permissions
  - 20 permissions for sensitive sources
    - ACCESS_FINE_LOCATION (8 methods with source annotations)
    - READ_PHONE_STATE - (9 methods)
  - 4 permissions for sensitive sinks
    - INTERNET, SEND_SMS, etc.

Identifying Sensitive Data

```
android.Telephony.TelephonyManager: String getDeviceId()
```

- Returns device IMEI in String
- Requires permission GET_PHONE_STATE

```
@STAMP(
   SRC ="$GET_PHONE_STATE.deviceid",
   SINK ="@return"
)
```

Data We Track (Sources)

- Account data
- Audio
- Calendar
- Call log
- Camera
- Contacts
- Device Id
- Location
- Photos (Geotags)
- SD card data
- SMS

30+ types of sensitive data

Data Destinations (Sinks)

- Internet (socket)
- SMS
- Email
- System Logs
- Webview/Browser
- File System
- Broadcast Message

10+ types of exit points

Currently Detectable Flow Types

396 Flow Types

Unique Flow Types = Sources x Sink

Example Analysis

Contact Sync for Facebook (unofficial)

Description:

This application allows you to synchronize your Facebook contacts on Android.

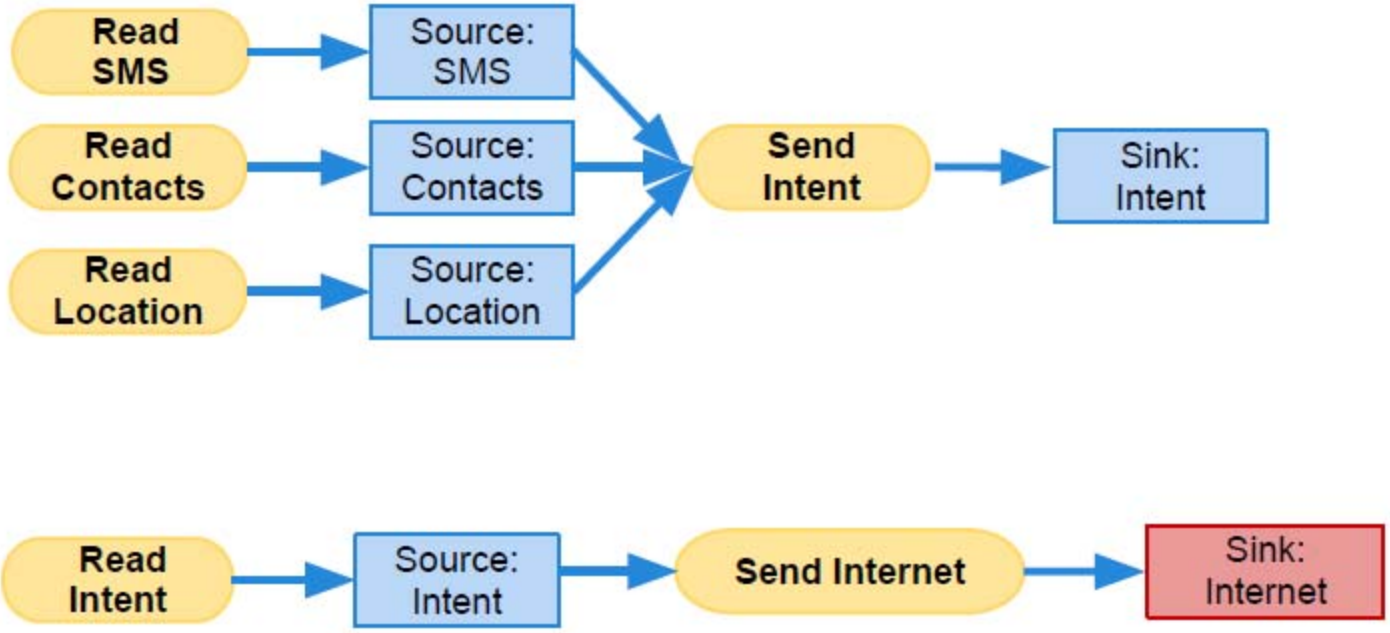
IMPORTANT:

* "Facebook does not allow [sic] to export phone numbers or emails. Only names, pictures and statuses are synced."

* "Facebook users have the option to block one or all apps. If they opt for that, they will be EXCLUDED from your friends list."

Privacy Policy: (page not found)

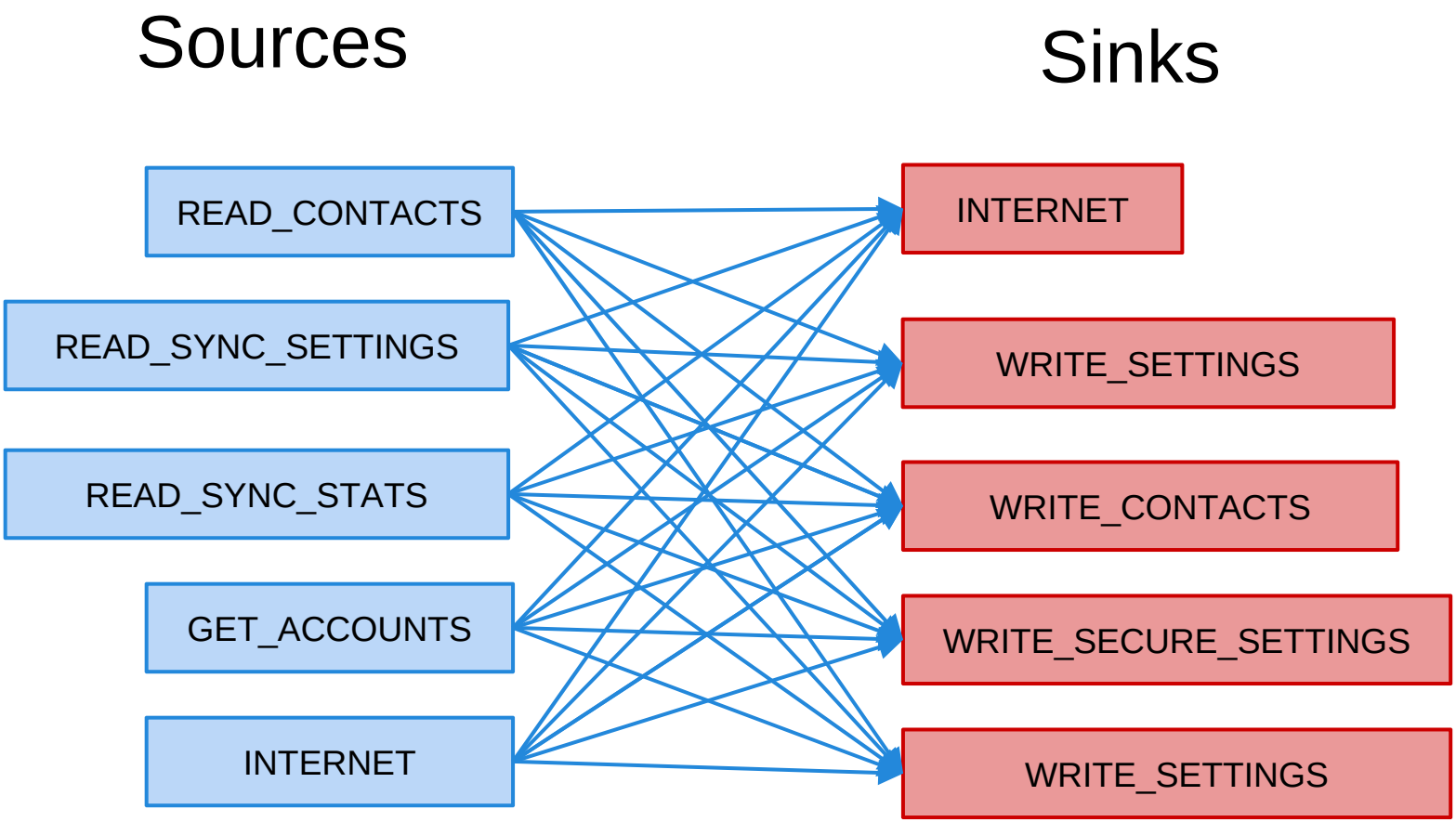
Chuli source-to-sink flows



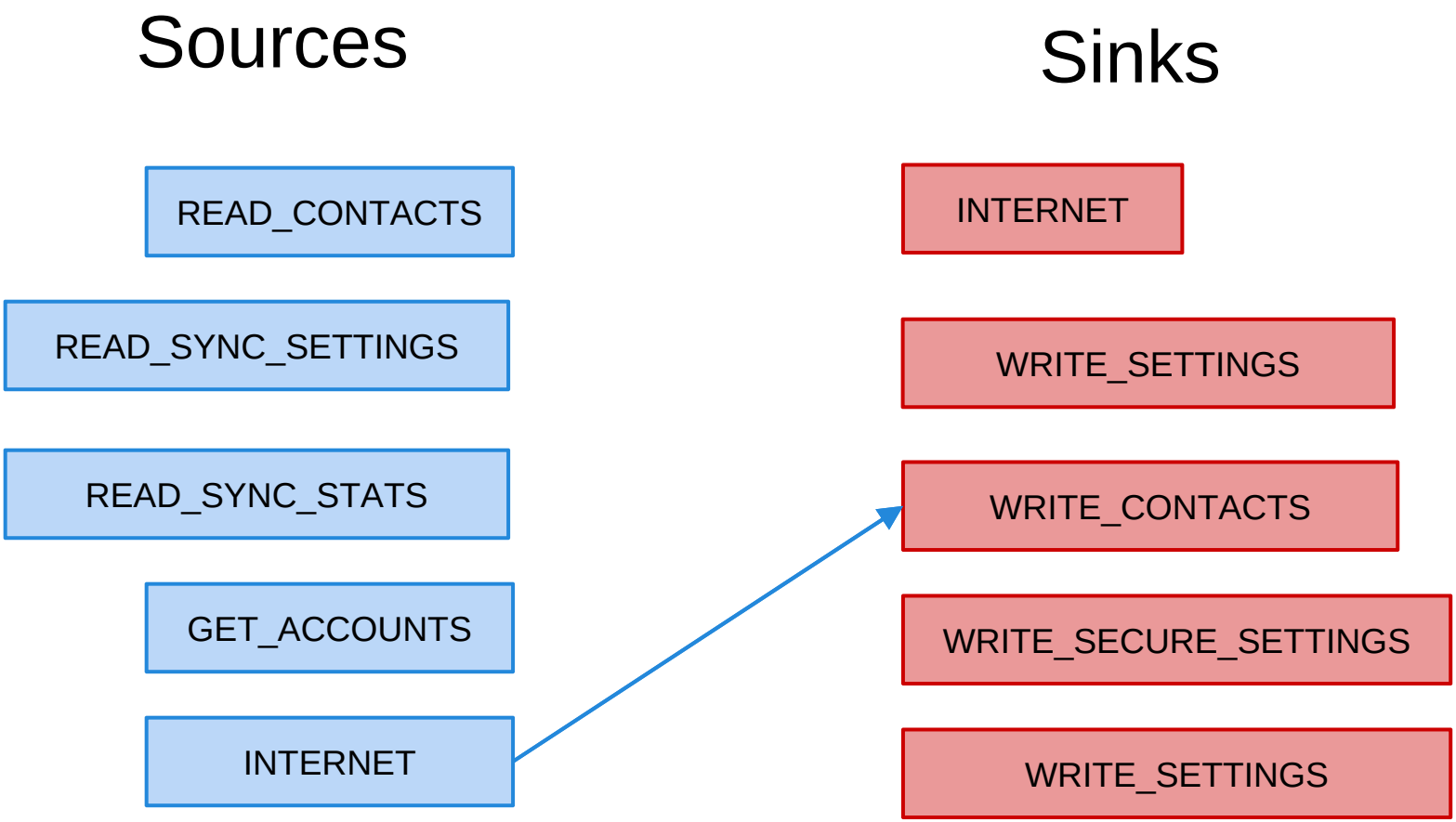
Contact Sync Permissions

Category	Permission	Description
Your Accounts	AUTHENTICATE_ACCOUNTS	Act as an account authenticator
	MANAGE_ACCOUNTS	Manage accounts list
	USE_CREDENTIALS	Use authentication credentials
Network Communication	**INTERNET**	**Full Internet access**
	ACCESS_NETWORK_STATE	View network state
Your Personal Information	**READ_CONTACTS**	**Read contact data**
	WRITE_CONTACTS	Write contact data
System Tools	WRITE_SETTINGS	Modify global system settings
	WRITE_SYNC_SETTINGS	Write sync settings (e.g. Contact sync)
	READ_SYNC_SETTINGS	Read whether sync is enabled
	READ_SYNC_STATS	Read history of syncs
Your Accounts	GET_ACCOUNTS	Discover known accounts
Extra/Custom	WRITE_SECURE_SETTINGS	Modify secure system settings

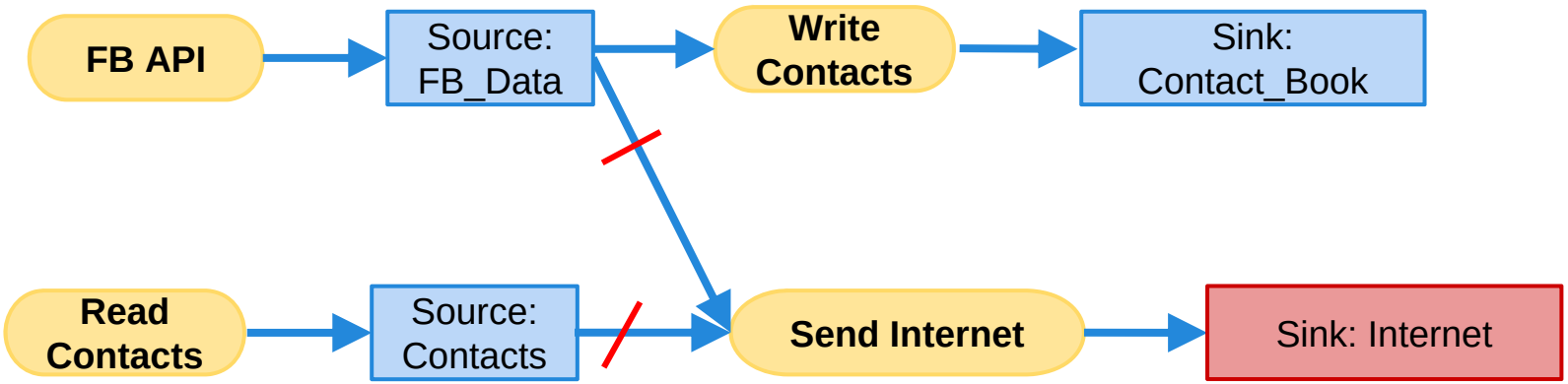
Possible Flows from Permissions



Expected Flows



Observed Flows



Outline

- Mobile malware
  - Common cases involve command and control, information theft
- Identifying malware
  - Detect at app store rather than on platform
- ➡ Classification study of mobile web apps
  - Entire Google Play market as of 2014
  - 85% of approx 1 million apps use web interface
  - 28% have at least one vulnerability

A Large-Scale Study of Mobile Web App Security

Patrick Mutchler, Adam Doupe,
John Mitchell, Chris Kruegel, Giovanni Vigna

Mobile Apps

Mobile Apps


android
SAMSUNG


No Service
12:44 PM
Messages
Monday
15
Calendar
Photos
Camera
Videos
Weather
Passbook
Notes
Reminders
Clock
Stocks
Newsstand
iTunes
App Store
Game Center
Settings
Speed Test
Sensorly
Maps
Phone
Mail
Safari
Music

Mobile Apps


android
SAMSUNG


No Service
12:44 PM
Messages
Calendar
Photos
Camera
Videos
Weather
Passbook
Notes
Reminders
Clock
Stocks
Newsstand
iTunes
App Store
Game Center
Settings
Speed Test
Sensorly
Maps
Phone
Mail
Safari
Music


4G
12:38
Phone
Photos

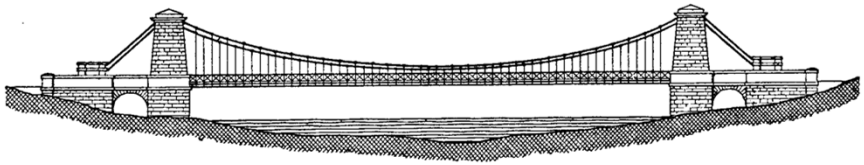
Mobile Web Apps

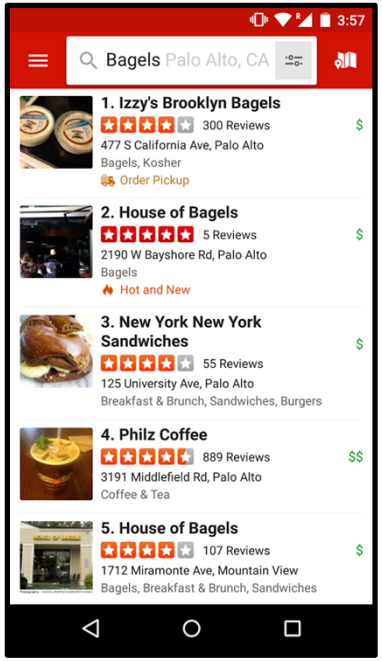
- Mobile web app: embeds a fully functional web browser as a UI element

JavaScript Bridge

```
Obj foo = new Object();
addJavascriptInterface(foo, ‘f’);
```

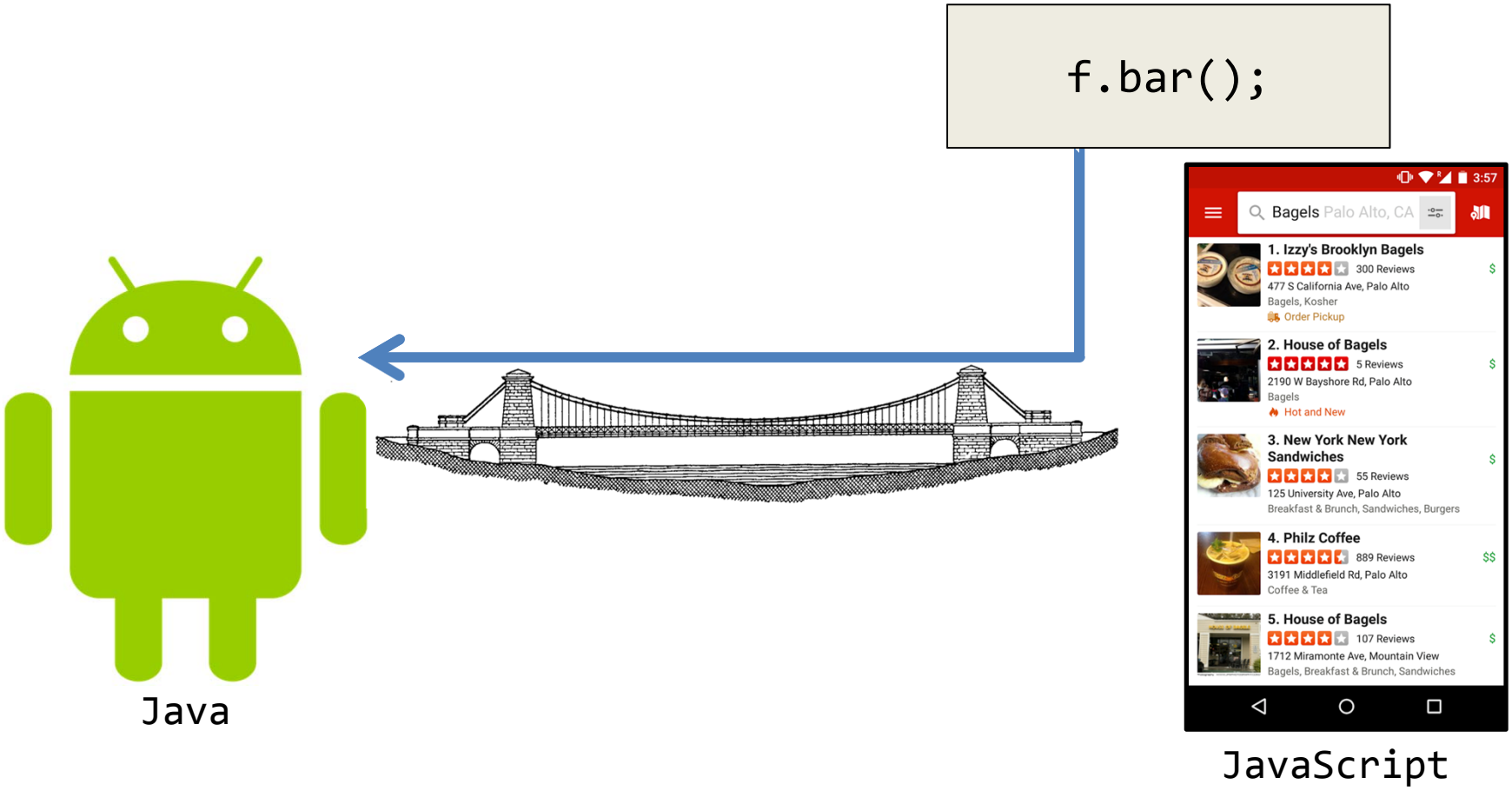
Java





JavaScript

JavaScript Bridge



Why?

- Full-featured mobile web apps
- Expose phone functionality to JavaScript

Security Concerns

- Who can access the bridge?
  - Everyone

October 9, 2014

THE HUFFINGTON POST

Edition: U.S. | Search The Huffington Post | Like 4.3m | Follow | Follow 3.1M

FRONT PAGE | POLITICS | BUSINESS | ENTERTAINMENT | TECH | MEDIA | WORLDPOST | HEALTHY LIVING | COMEDY | HUFFPOST LIVE | ALL SECTIONS

Black Voices • Gay Voices • Sports • Crime • Science • Religion • Celebrity • Green • Style • Horoscopes • Third Metric • OWN • Dr Phil • GPS for the Soul

WATCH LIVE: Dove's 'Legacy' From Mother to Daughter | COMING UP FRIDAY: Top Stories For Friday, Oct. 10 | COMING UP | Enter email address | Subscribe

46 U.S. CRUISE MISSILES 'ONE OR TWO' KEY KHORASAN KILLED 'A DOZEN' CIVILIANS DEAD

Complete Isolation

Comments | Shares (56) | Syria

FEATURED BLOG POSTS

Desmond Tutu... Vivek Wadhwa... Alex Ebert...

Josh Horwitz
Executive Director, Coalition to Stop Gun Violence

The Racial Double Standard on Gun Violence

The way we talk about incidents of gun violence in this country -- and the solutions we propose to stem future acts of violence -- seems to be dramatically different depending on the race of those involved. Consider the tragic death of 25 year-old African-American Kajieme Powell in St. Louis this summer. It was a textbook example of suicide-by-cop. And yet very little of the subsequent national

Marriage Equality... In West Virginia!

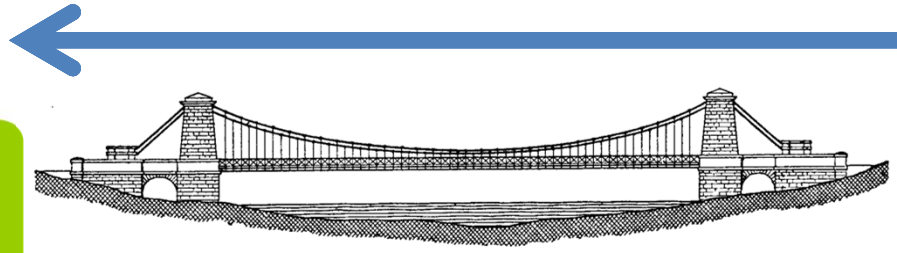
MORE POLITICS Seriously, GOP?!.. Rick Scott Lawsuit.. Harsh Law Explained.. SCOTUS: Our Bad!.. GOPer Goes Quie

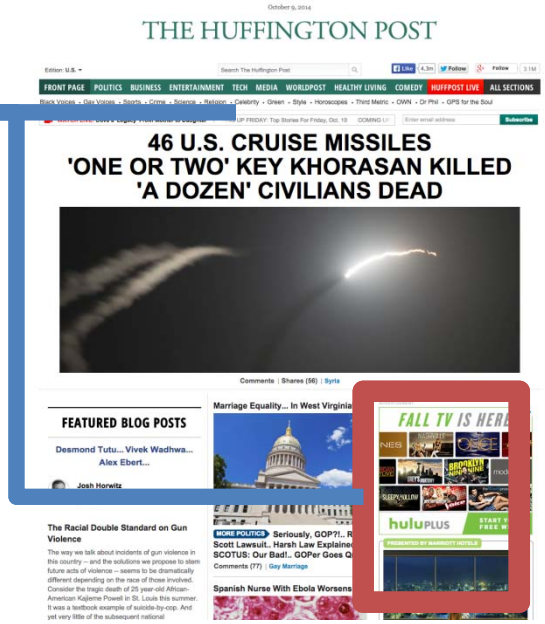
Comments (77) | Gay Marriage

Spanish Nurse With Ebola Worsens

```
f.bar();
```

Java

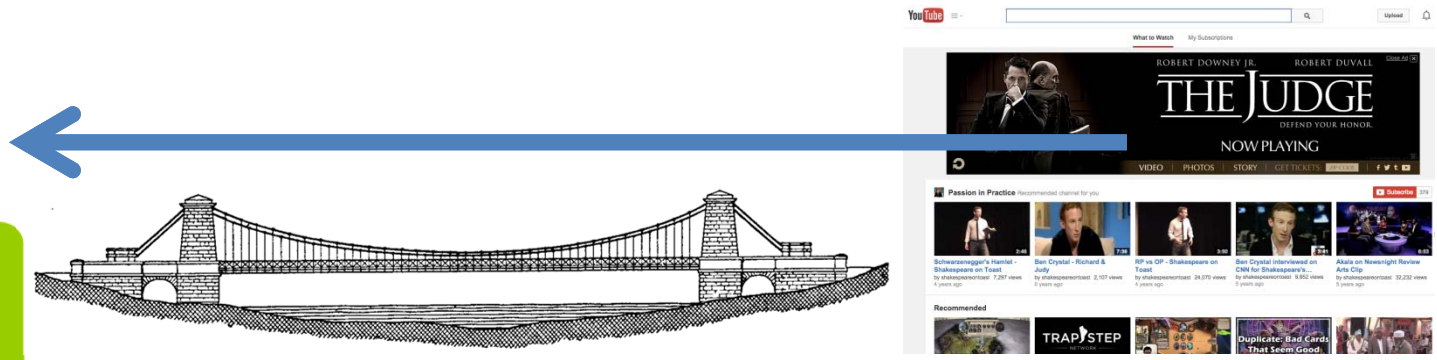




JavaScript

```
f.bar();
```

Java



JavaScript

Static Analysis

- How many mobile web apps?
- How many use JavaScript Bridge?
- How many vulnerable?

Experimental Results

- 737,828 free apps from Google Play (Oct ’13)
- 563,109 apps embed a browser
- 219,404 use the JavaScript Bridge
- 107,974 have at least one security violation

Most significant vulnerabilities

1. Loading untrusted web content
2. Leaking URLs to foreign apps
3. Exposing state changing navigation to foreign apps

1. Loading untrusted web content
2. Leaking URLs to foreign apps
3. Exposing state changing navigation to foreign apps

“You should restrict the web-pages that can load inside your WebView with a whitelist.”

- Facebook

“…only loading content from trusted sources into WebView will help protect users.”

- Adrian Ludwig, Google

1. Navigate to untrusted content

```
// In app code
myWebView.loadUrl(“foo.com”);
```

```
// In app code
myWebView.load("foo.com");

<!-- In HTML -->
<a href="foo.com">click!</a>
```

```
// In app code
myWebView.load("foo.com");

<!-- In HTML -->
<a href="foo.com">click!</a>

<!-- More HTML -->
<iframe src="foo.com"/>
```

```
// In app code
myWebView.loadUrl("foo.com");

<!-- In HTML -->
<a href="foo.com">click!</a>

<!-- More HTML -->
<iframe src="foo.com"/>

// In JavaScript
window.location = "foo.com";
```

```
public boolean shouldOverrideUrlLoading(
      WebView view, String url){

   // False -> Load URL in WebView
   // True  -> Prevent the URL load


}
```

```
public boolean shouldOverrideUrlLoading(
      WebView view, String url){

   String host = new URL(url).getHost();
   if(host.equals("stanford.edu"))
      return false;
   log("Overrode URL: " + url);
   return true;
}
```

```
public boolean shouldOverrideUrlLoading(
      WebView view, String url){

   String host = new URL(url).getHost();
   if(host.equals("stanford.edu"))
      return false;
   log("Overrode URL: " + url);
   return true;
}
```

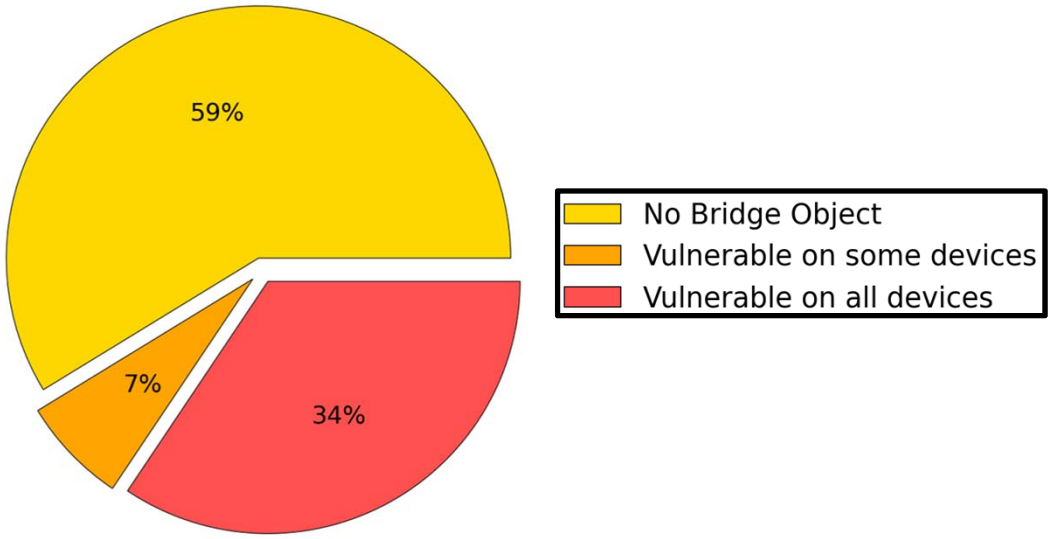
Reach Untrusted Content?

- 40,084 apps with full URLs and use JavaScript Bridge
- 13,683 apps (34%) can reach untrusted content

What does untrusted mean?

Use HTTPS?

- 152,706 apps with partially computed URLs
- 87,968 apps (57%) with HTTP URLs



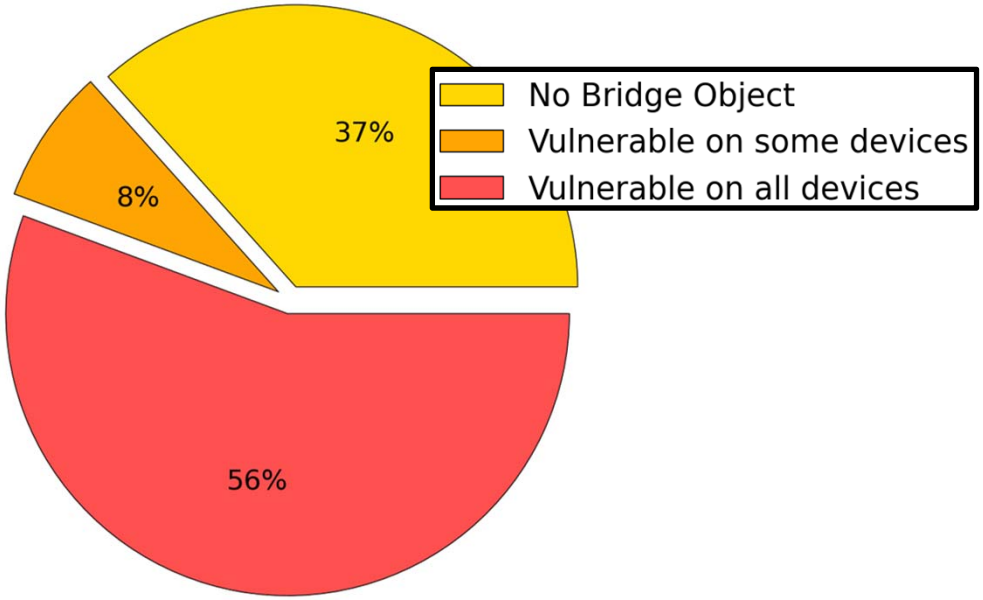
Handling SSL Errors

```
onReceivedSslError

1. handler.proceed()
2. handler.cancel()
3. view.loadUrl(...)
```

Mishandling SSL Errors

- 117,974 apps implement `onReceivedSslError`
- 29,652 apps (25%) **must** ignore errors

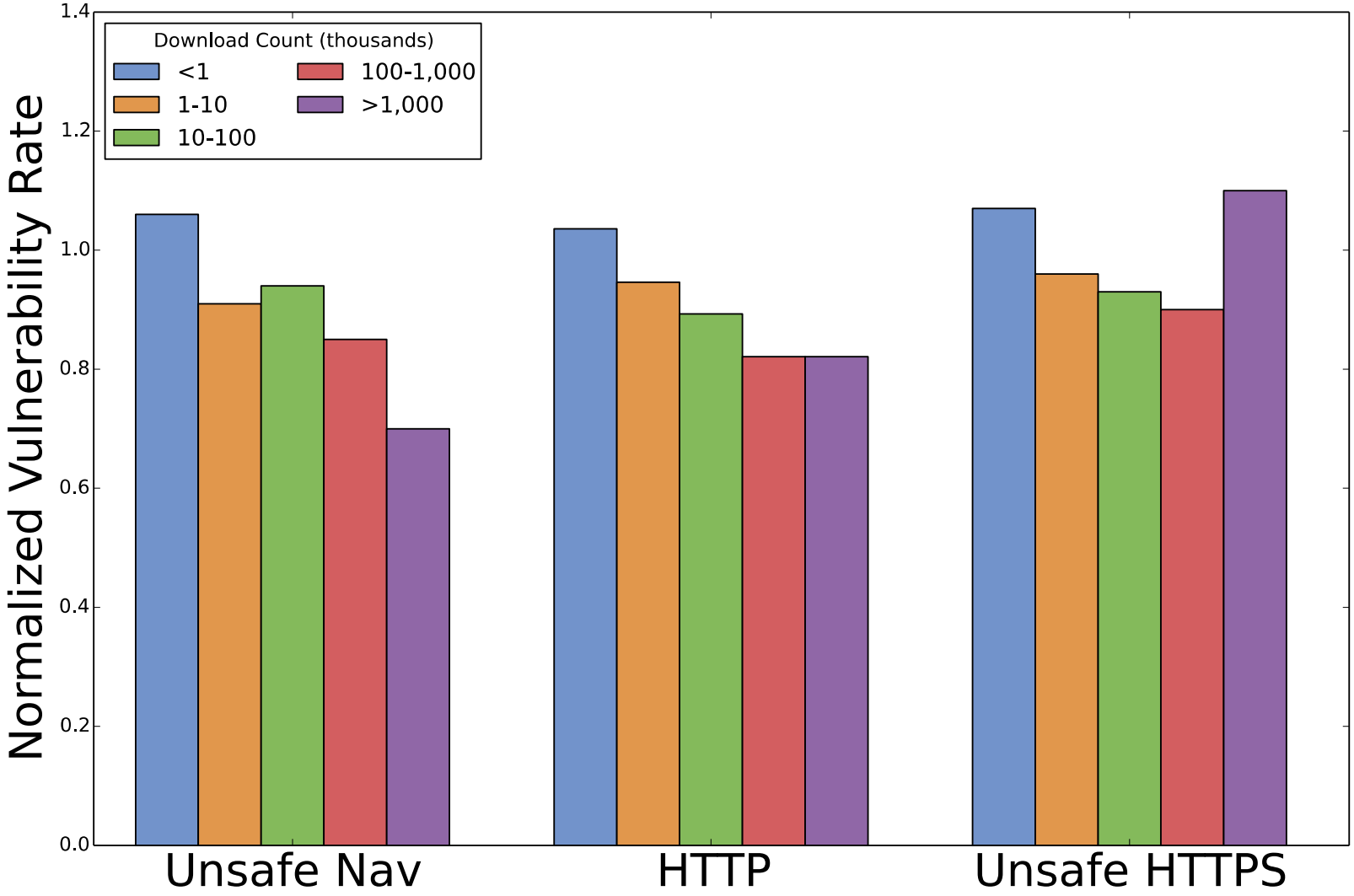


Results

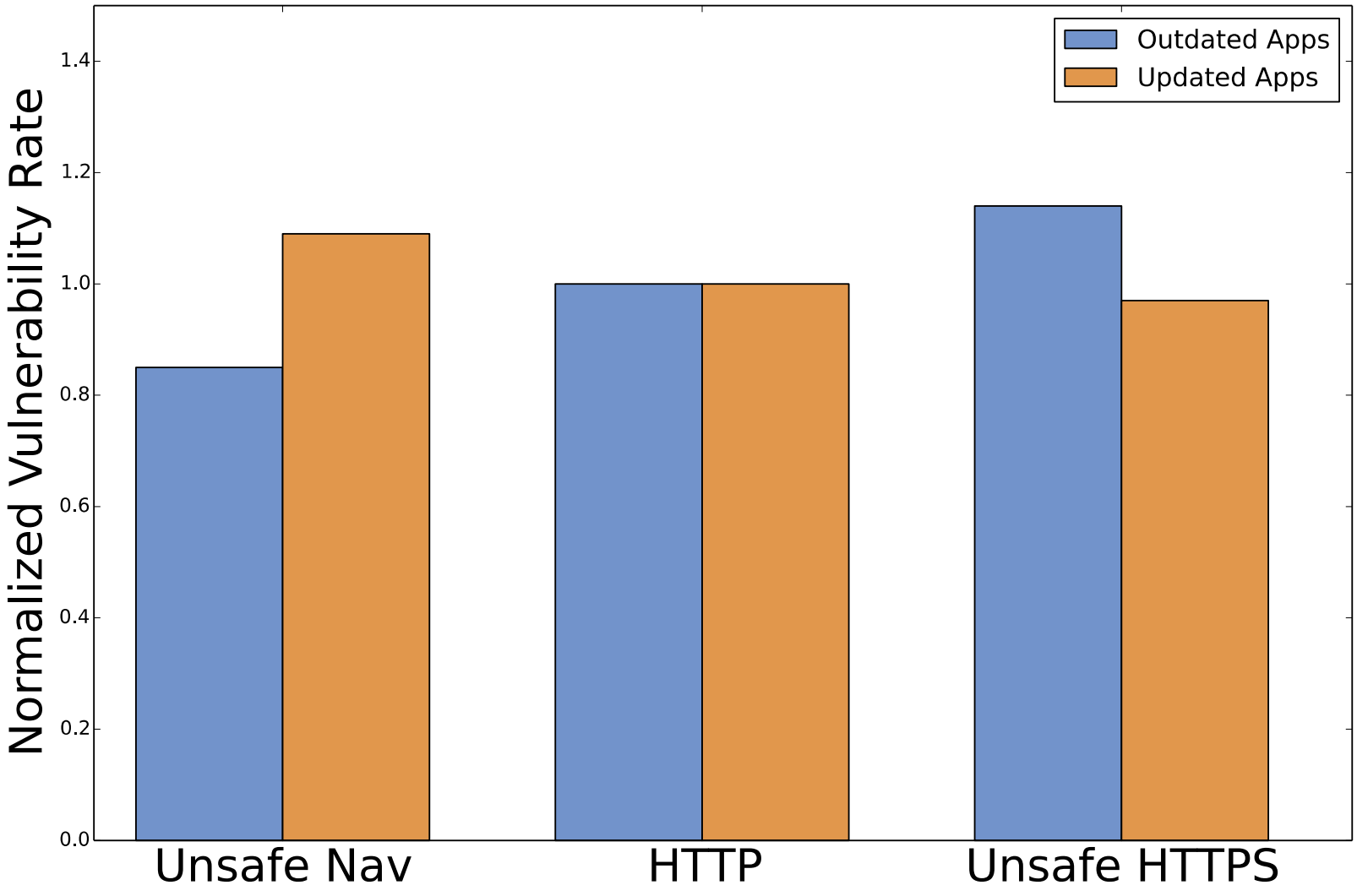
Primary results

Vulnerability	% Relevant	% Vulnerable
Unsafe Nav	15	34
HTTP	40	56
Unsafe HTTPS	27	29

Popularity


Download Count (thousands)
<1
1-10
10-100
100-1,000
>1,000
Normalized Vulnerability Rate
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Unsafe Nav
HTTP
Unsafe HTTPS

Outdated Apps



Libraries

29%
unsafe nav

51%
HTTP

53%
unsafe HTTPS

Additional security issues

Analyze 998,286 free web apps from June 2014

Mobile Web App Feature	% Apps
JavaScript Enabled	97
JavaScript Bridge	36
shouldOverrideUrlLoading	94
shouldInterceptRequest	47
onReceivedSslError	27
postUrl	2
Custom URL Patterns	10

Vuln	% Relevant	% Vulnerable
Unsafe Navigation	15	34
Unsafe Retrieval	40	56
Unsafe SSL	27	29
Exposed POST	2	7
Leaky URL	10	16

Takeaways

- Apps must not load untrusted content into WebViews
- Able to identify violating apps using static analysis
- Vulnerabilities are present in the entire app ecosystem

Outline

- Mobile malware
  - Common cases involve command and control, information theft
- Identifying malware
  - Detect at app store rather than on platform
- Classification study of mobile web apps
  - Entire Google Play market as of 2014
  - 85% of approx 1 million apps use web interface
  - 28% have at least one vulnerability

Summary

- Analyze a dataset of 737,828 Android apps
- Found large number of apps contain severe vulnerabilities
- 37,418 apps are vulnerable to a remote code execution exploit when run on any Android device, because of security oversight in older versions and slow adoption of safe versions
- 45,689 apps are vulnerable to a remote code execution exploit when run on 73% of the in-use Android devices.
- Offer recommendations for developers who wish to avoid these vulnerabilities.