Mobile Device and Platform Security – Part II

John Mitchell

Guest Lecture Tuesday June 6

- **Diogo Mónica,** Director of security at Docker
- Topics include container security and what it is like working as a security engineer
- Diogo a very good speaker. Please show your appreciating and come to class.
- We will include topics on the final exam

Two lectures on mobile security

- Introduction: platforms and trends
- Threat categories
  - Physical, platform malware, malicious apps
- Defense against physical theft
- Malware threats
- System architecture and defenses
  - Apple iOS security features and app security model
  - Android security features and app security model
- Security app development
  - WebView – secure app and web interface dev
  - Device fragmentation

Tues

Thurs

ANDROID

History and early decisions

Android history

- Android, Inc founded by Andy Rubin around 2005
  - Worked with HTC-built device with a physical keyboard
  - Scrapped Blackberry-like phone when iPhone came out
  - First Android phone HTC Dream, Oct 2008 (T-Mobile G1): touchscreen and keyboard
- Open-source software project
- Backed and acquired by Google

HTC Dream

- First phone had
  - Android 1.6 (Donut)
  - 3.15 megapixel rear camera with auto-focus
  - 3.2 inch touchscreen
  - Gmail, Google Maps, Search, Google Talk, You Tube, calendar, contacts, alarm

Android ecosystem

- Open-source software distributed by Google
  - Increase number of users and devices linked to core Google products
- Multiple hardware vendors
  - Can customize software for their products
- Open marketplace for apps

App market

- Self-signed apps
- App permissions
  - granted on user installation
- Open market
  - Bad apps may show up on market
  - Shifts focus from remote exploit to privilege escalation

ANDROID PLATFORM

Device locking and permissions

Device lock and unlock

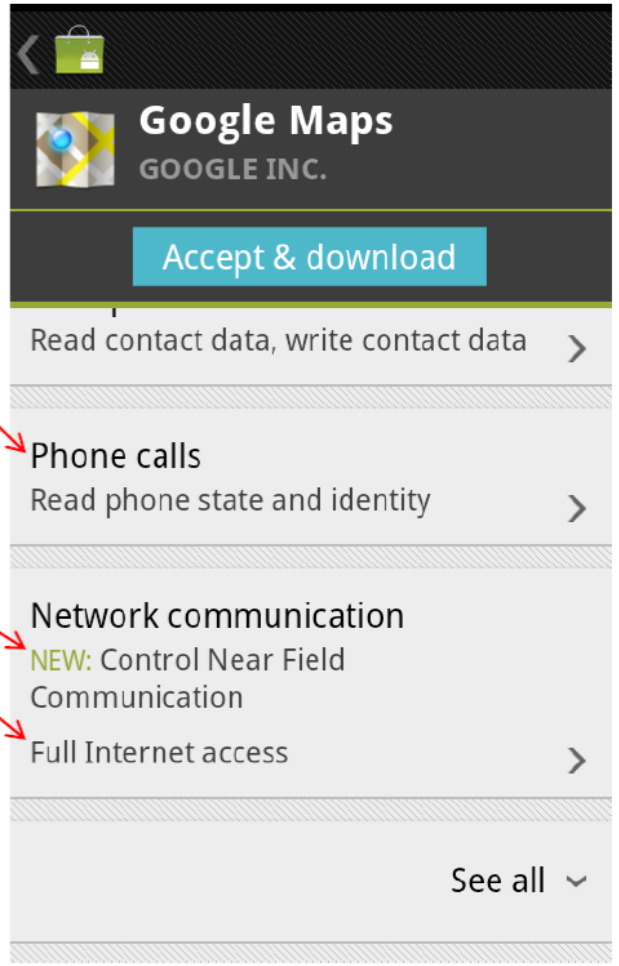
- Similar PIN and fingerprint
- Fingerprint API lets users
  - Unlock device
  - Securely sign in to apps
  - Use Android Pay
  - Purchase on Play Store

Android permissions

- Example of permissions provided by Android
  - “android.permission.INTERNET”
  - “android.permission.READ_EXTERNAL_STORAGE
  - “android.permission.SEND_SMS”
  - “android.permission.BLUETOOTH”
- Also possible to define custom permissions

Android permission model

```
...
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.NFC" />
<uses-permission android:name="android.permission.INTERNET" />
...
```



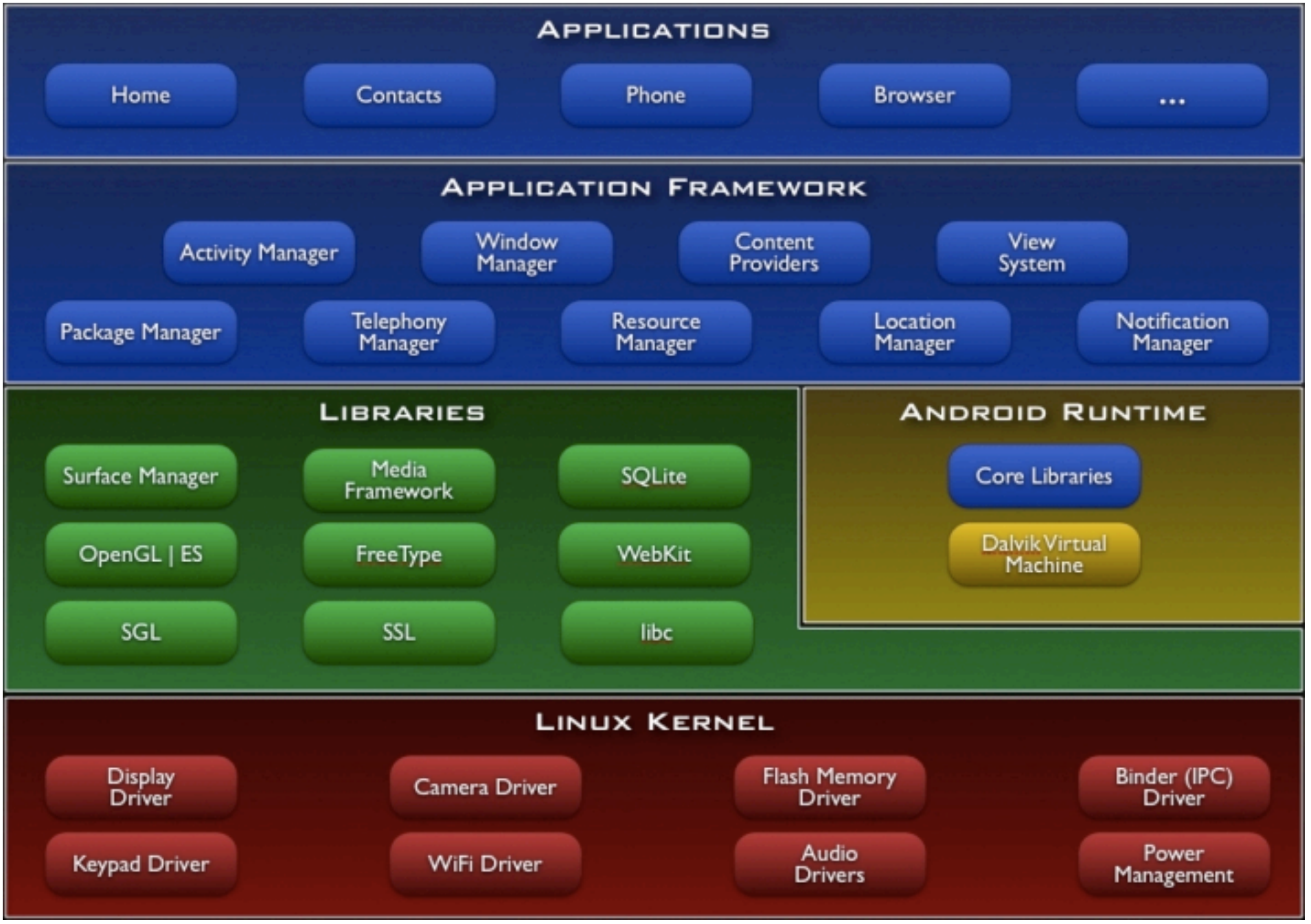
https://www.owasp.org/images/3/3e/Danelon_OWASP_EU_Tour_2013.pdf

ANDROID PLATFORM

Platform security features

Android

- Platform outline:
  - Linux kernel, browser, SQL-lite database
  - Software for secure network communication
    - Open SSL, Bouncy Castle crypto API and Java library
  - C language infrastructure
  - Java platform for running applications
    - Dalvik bytecode, virtual machine


Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Exploit prevention

- Open source: public review, no obscurity
- Goals
  - Prevent remote attacks, privilege escalation
  - Secure drivers, media codecs, new and custom features
- Overflow prevention
  - ProPolice stack protection
    - First on the ARM architecture
  - Some heap overflow protections
    - Chunk consolidation in DL malloc (from OpenBSD)
- ASLR
  - Avoided in initial release
    - Many pre-linked images for performance
  - Later developed and contributed by Bojinov, Boneh

dlmalloc (Doug Lea)

- Stores meta data in band
- Heap consolidation attack
  - Heap overflow can overwrite pointers to previous and next unconsolidated chunks
  - Overwriting these pointers allows remote code execution
- Change to improve security
  - Check integrity of forward and backward pointers
    - Simply check that back-forward-back = back, f-b-f=f
  - Increases the difficulty of heap overflow

ANDROID PLATFORM

App execution environment

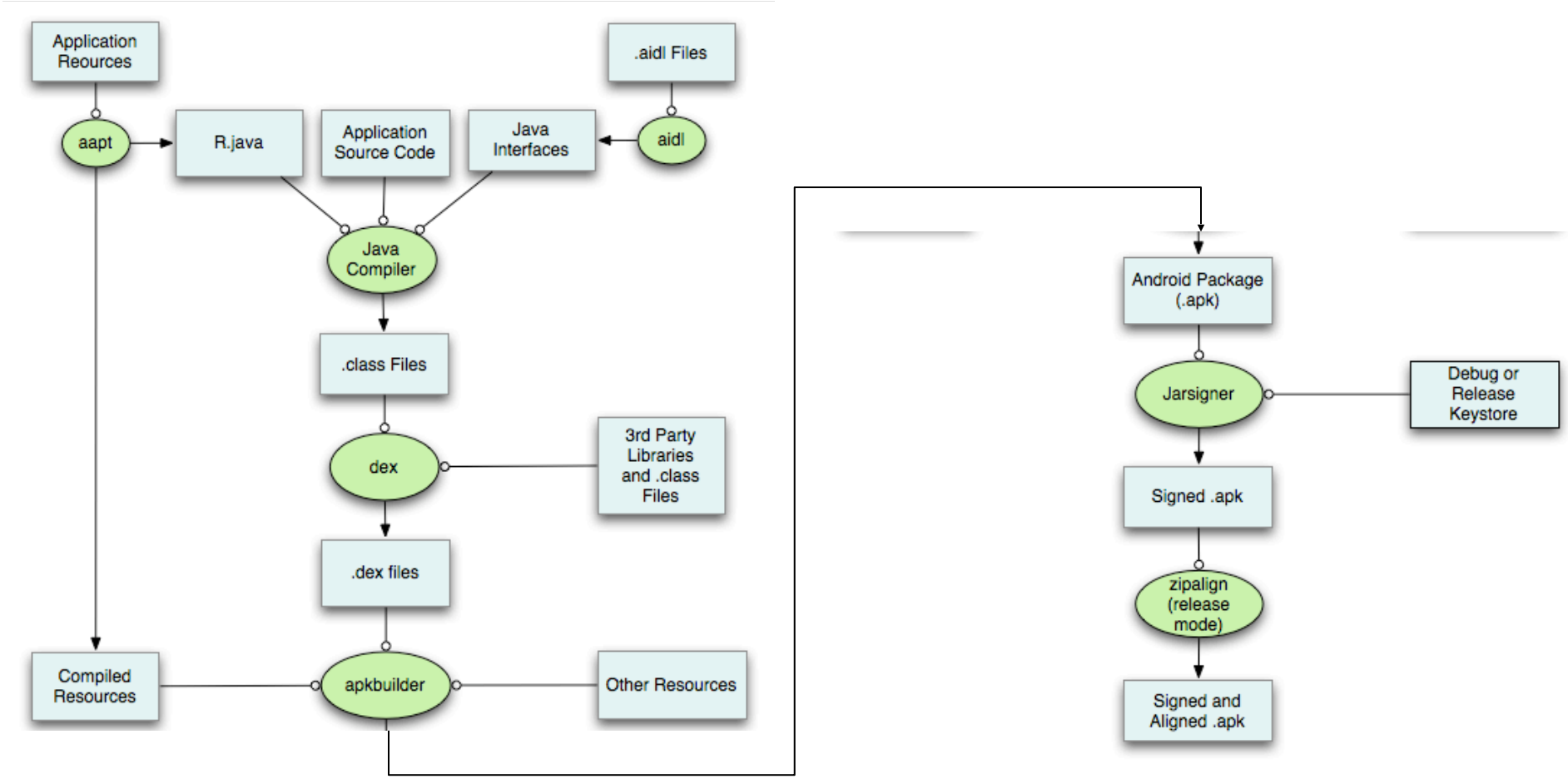
Application development concepts

- Activity – one-user task
  - Example: scroll through your inbox
  - Email client comprises many activities
- Service – Java daemon that runs in background
  - Example: application that streams an mp3 in background
- Intents – asynchronous messaging system
  - Fire an intent to switch from one activity to another
  - Example: email app has inbox, compose activity, viewer activity
    - User click on inbox entry fires an intent to the viewer activity, which then allows user to view that email
- Content provider
  - Store and share data using a relational database interface
- Broadcast receiver
  - "mailboxes" for messages from other applications

Security Features

- Isolation
  - Multi-user Linux operating system
  - Each application normally runs as a different user
- Communication between applications
  - May share same Linux user ID
    - Access files from each other
    - May share same Linux process and Dalvik VM
  - Communicate through application framework
    - “Intents,” based on Binder, discussed in a few slides
- Battery life
  - Developers must conserve power
  - Applications store state so they can be stopped (to save power) and restarted – helps with DoS

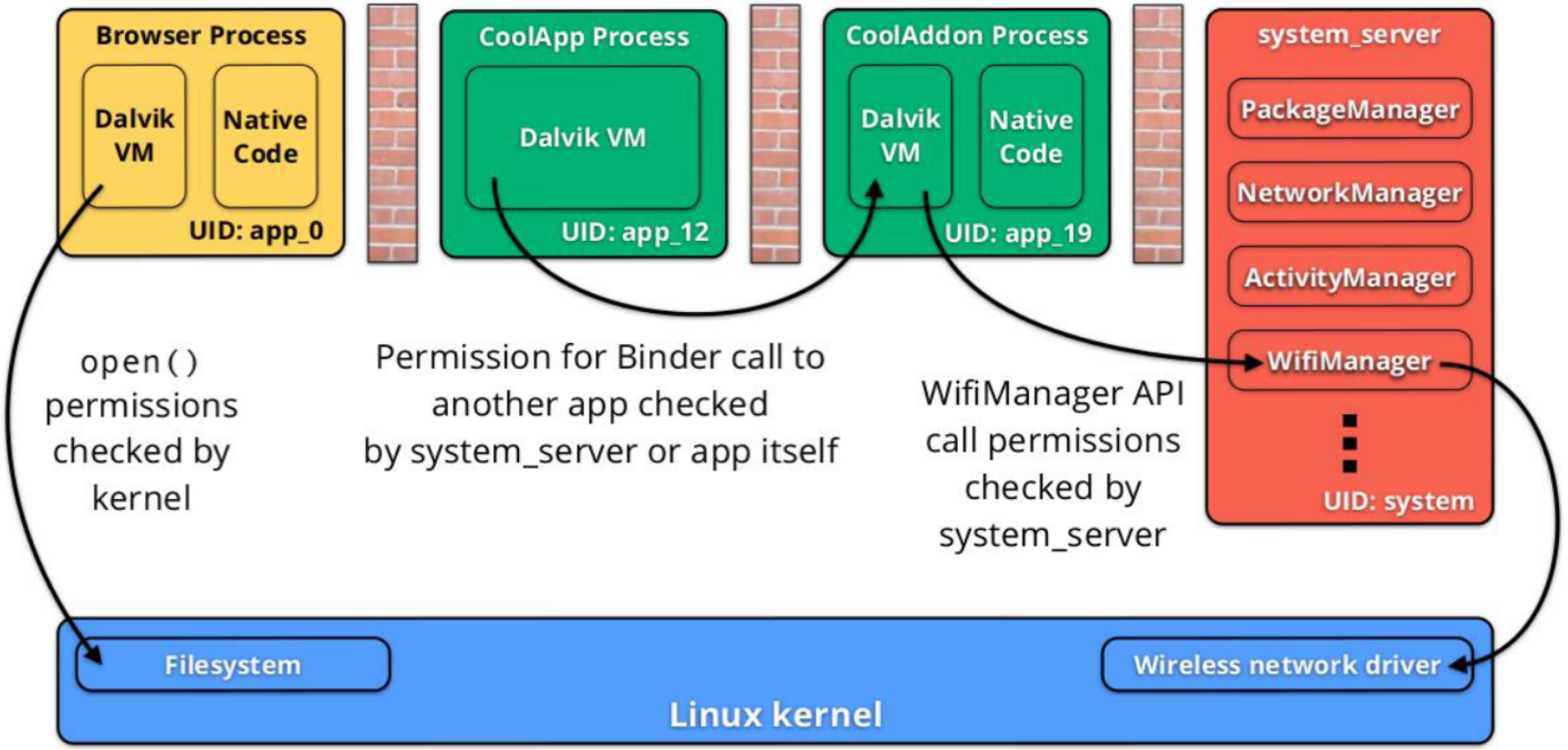
Application development process



Application sandbox

- Application sandbox
  - Each application runs with its UID in its own Dalvik virtual machine
    - Provides CPU protection, memory protection
    - Authenticated communication protection using Unix domain sockets
    - Only ping, zygote (spawn another process) run as root
  - Applications announce permission requirement
    - Create a whitelist model – user grants access
      - Don’t interrupt user – all questions asked as install time
    - Inter-component communication reference monitor checks permissions

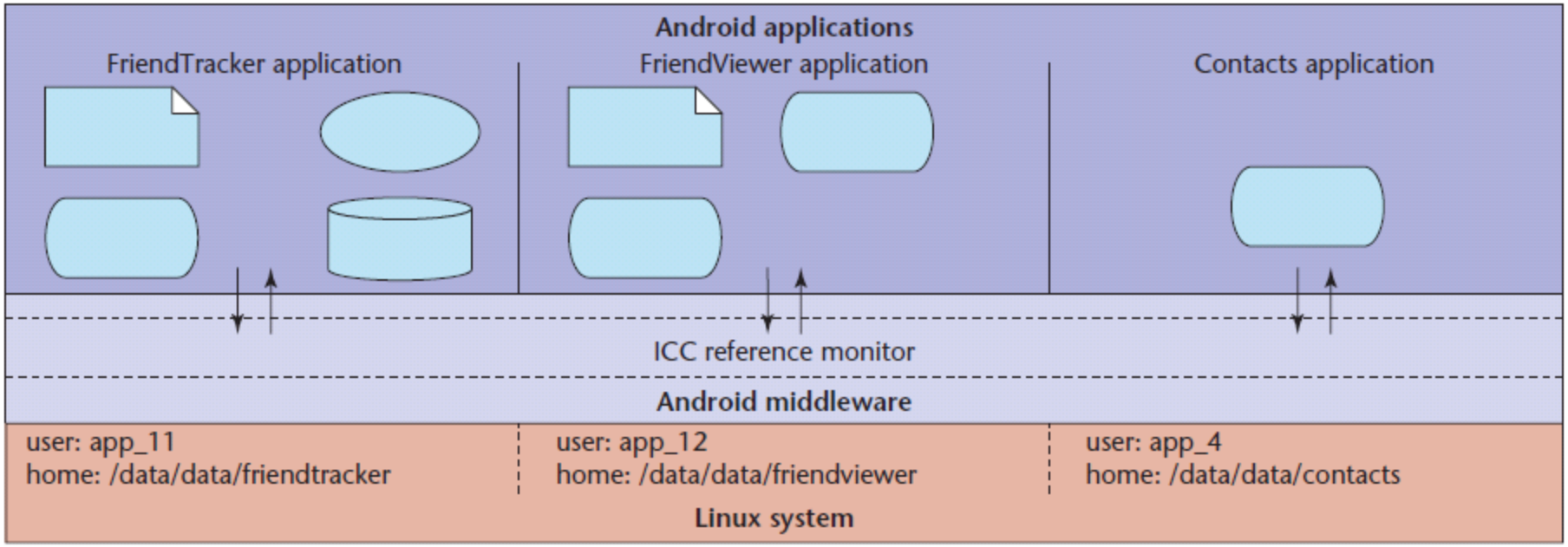
Android permission model



https://www.owasp.org/images/3/3e/Danelon_OWASP_EU_Tour_2013.pdf

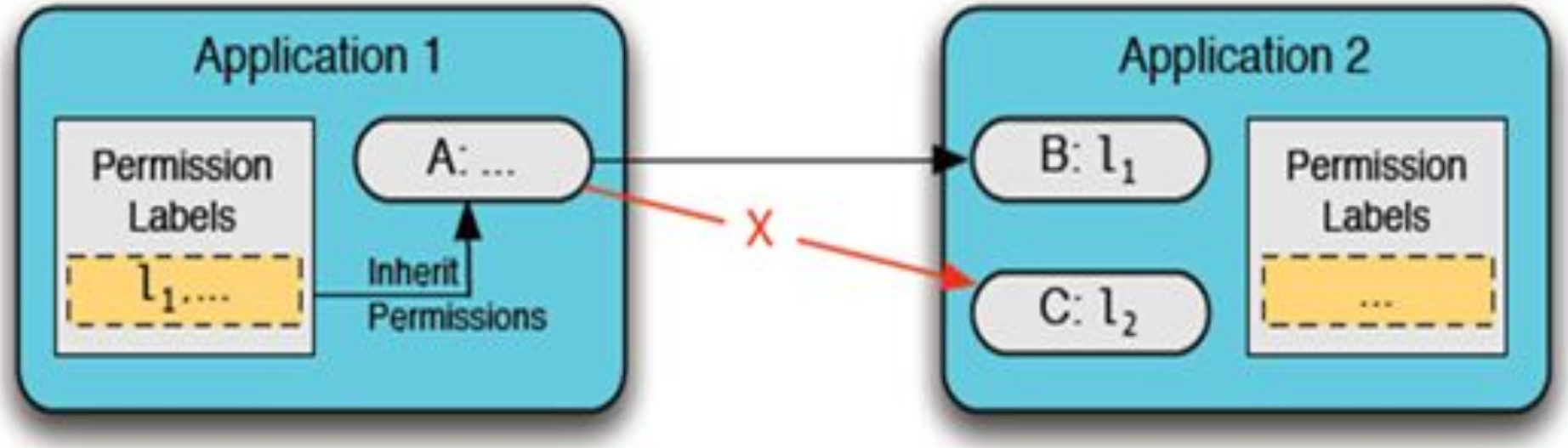
Android Intents

- Msg between components in same or different app
- Intent is a bundle of information, e.g.,
  - action to be taken
  - data to act on
  - category of component to handle the intent
  - instructions on how to launch a target activity
- Routing can be
  - Explicit: delivered only to a specific receiver
  - Implicit: all components that have registered to receive that action will get the message



- Layers of security
  - Each application executes as its own user identity
  - Android middleware has reference monitor that mediates the establishment of inter-component communication (ICC)

Source: Penn State group Android security paper



MAC Policy Enforcement in Android. This is how applications access components of other applications via the reference monitor. Component A can access components B and C if permission labels of application 1 are equal or dominate labels of application 2.

Source: Penn State group, Android security tutorial

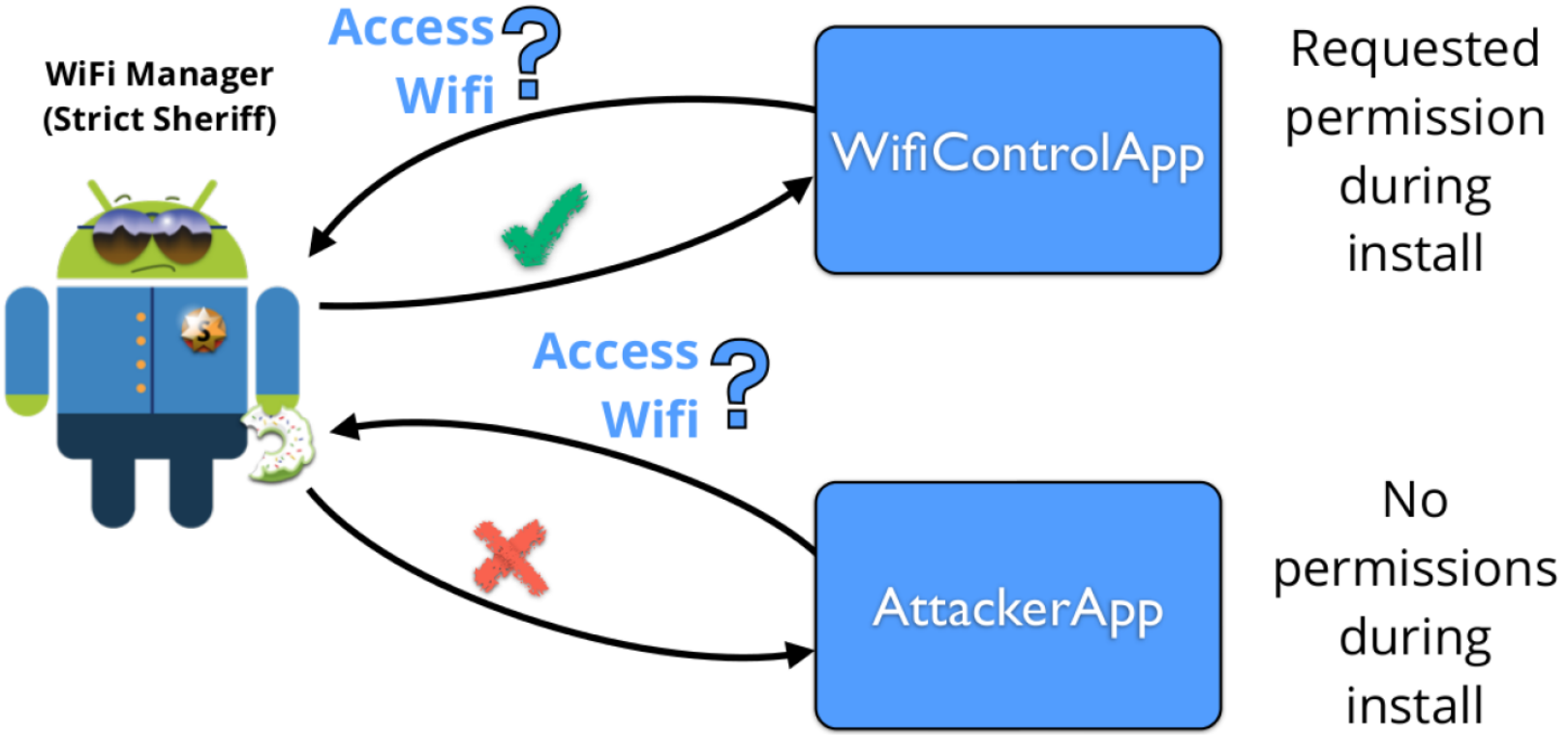
Security issues with intents

- Sender of an intent can verify that the recipient has a permission by specifying a permission with the method call
- Senders can use explicit intents to send the message to a single component (avoiding broadcasting)
- Receivers have to handle malicious intents

Attack: Permission redelegation

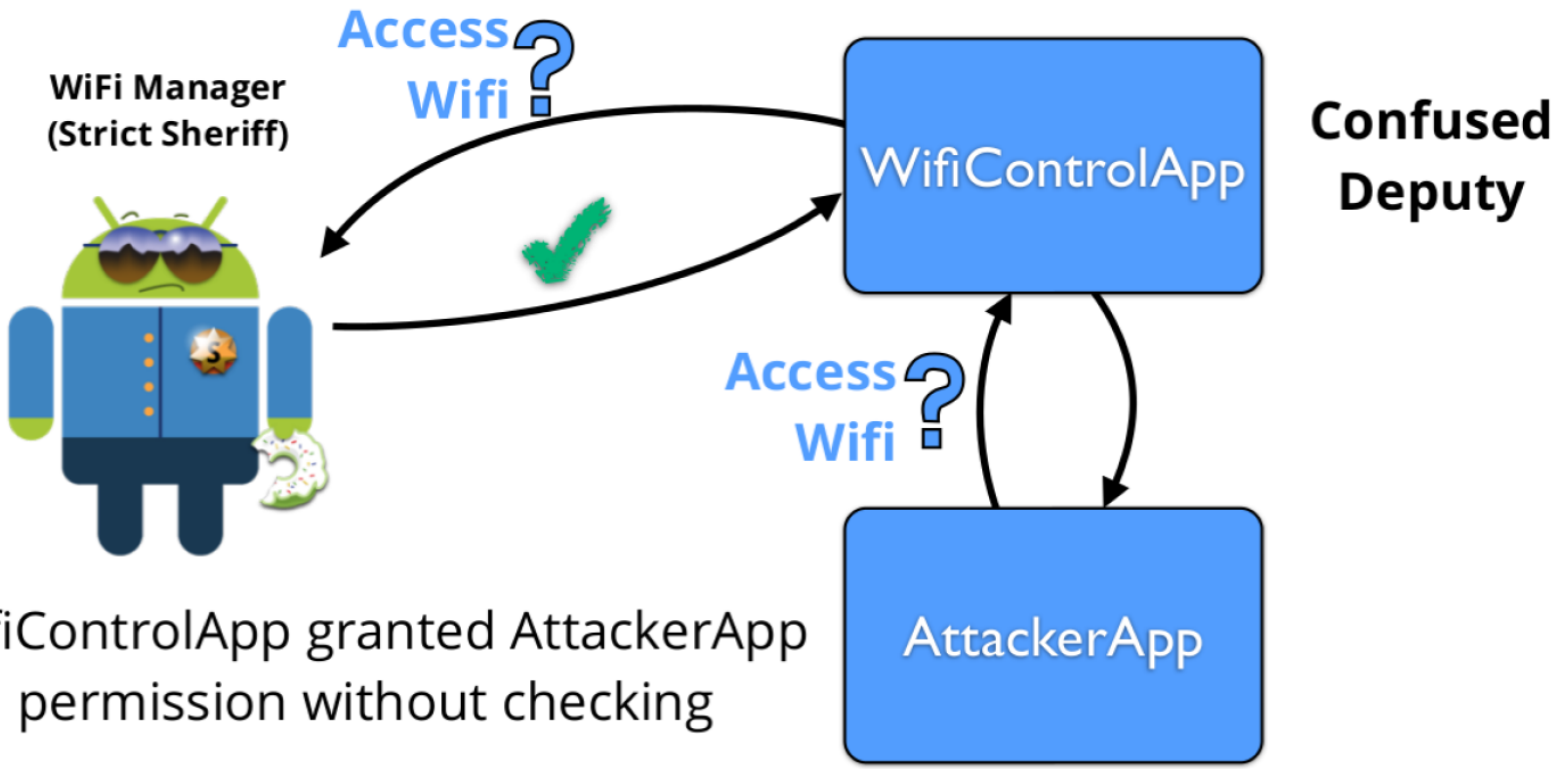
- Definition: an application without a permission gains additional privileges through another application
- Example of the “confused deputy” problem

Permission redelegation



https://www.owasp.org/images/3/3e/Danelon_OWASP_EU_Tour_2013.pdf

Permission redelegation



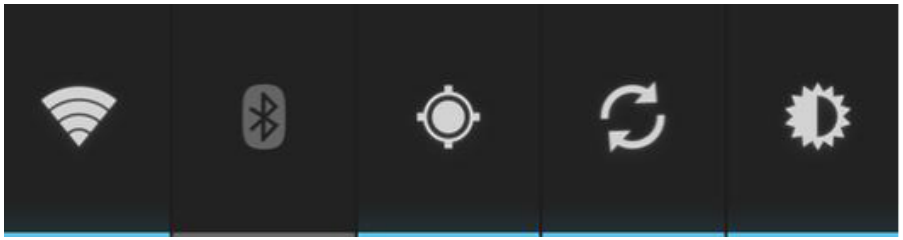
https://www.owasp.org/images/3/3e/Danelon_OWASP_EU_Tour_2013.pdf

How could this happen?

- App w/ permissions exposes a public interface
- Study in 2011
  - Examine 872 apps
  - 320 of these (37%) have permissions and at least one type of public component
  - Construct attacks using 15 vulnerabilities in 5 apps
- Reference
  - Permission Re-Delegation: Attacks and Defenses, Adrienne Felt, Helen Wang, Alexander Moshchuk, Steven Hanna, Erika Chin, Usenix 2011

Example: power control widget

- Default widgets provided by Android, present on all devices



- Can change Wi-fi, BT, GPS, Data Sync, Screen Brightness with only one click
- Uses Intent to communicate the event of switching settings
- A malicious app without permissions can send a fake Intent to the Power Control Widget, simulating click to switch settings

https://www.owasp.org/images/3/3e/Danelon_OWASP_EU_Tour_2013.pdf

Vulnerable versions (in red)

Version	Codename	API	Distribution
1.6	**Donut**	**4**	**0.10%**
2.1	**Eclair**	**7**	**1.50%**
2.2	**Froyo**	**8**	**3.20%**
2.3 - 2.3.2	Gingerbread	9	0.10%
2.3.3 - 2.3.7		10	36.40%
3.2	Honeycomb	13	0.10%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	25.60%
4.1.x	**Jelly Bean**	16	29.00%
4.2.x		**17**	**4.00%**

- Principle of least privilege helps but is not a solution
- Apps with permissions need to manage security

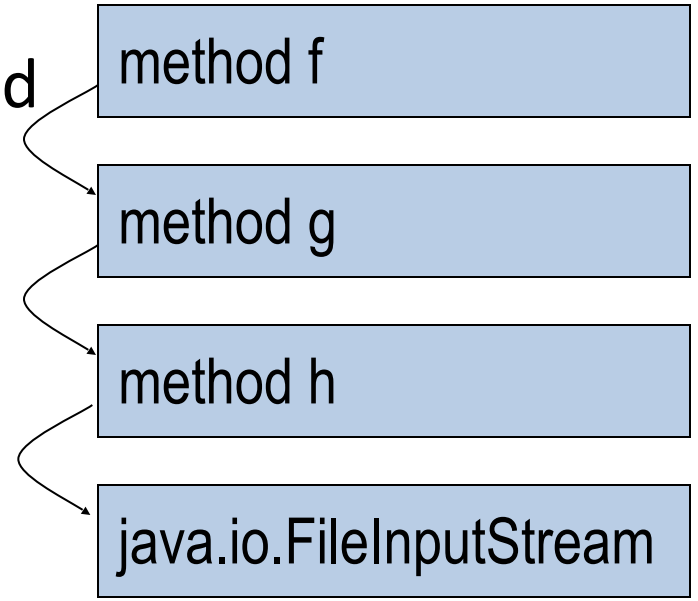
https://www.owasp.org/images/3/3e/Danelon_OWASP_EU_Tour_2013.pdf

Java Sandbox

- Four complementary mechanisms
  - Class loader
    - Separate namespaces for separate class loaders
    - Associates *protection domain* with each class
  - Verifier and JVM run-time tests
    - NO unchecked casts or other type errors, NO array overflow
    - Preserves private, protected visibility levels
  - Security Manager
    - Called by library functions to decide if request is allowed
    - Uses protection domain associated with code, user policy

Stack Inspection

- Permission depends on
  - Permission of calling method
  - Permission of all methods above it on stack
    - Up to method that is trusted and asserts this trust



Stories: Netscape font / passwd bug; Shockwave plug-in

Many details omitted here

ANDROID MALWARE

Android malware example

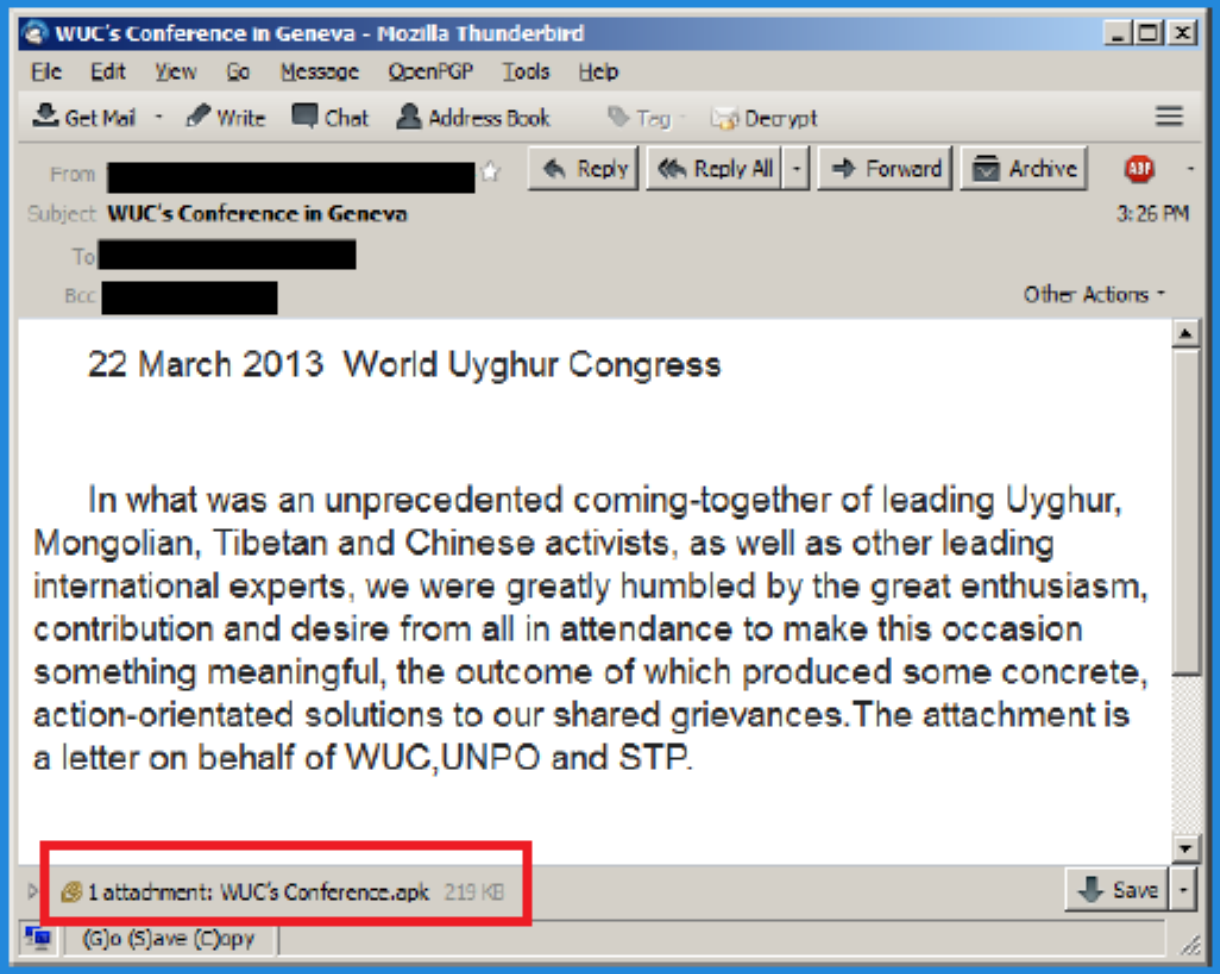
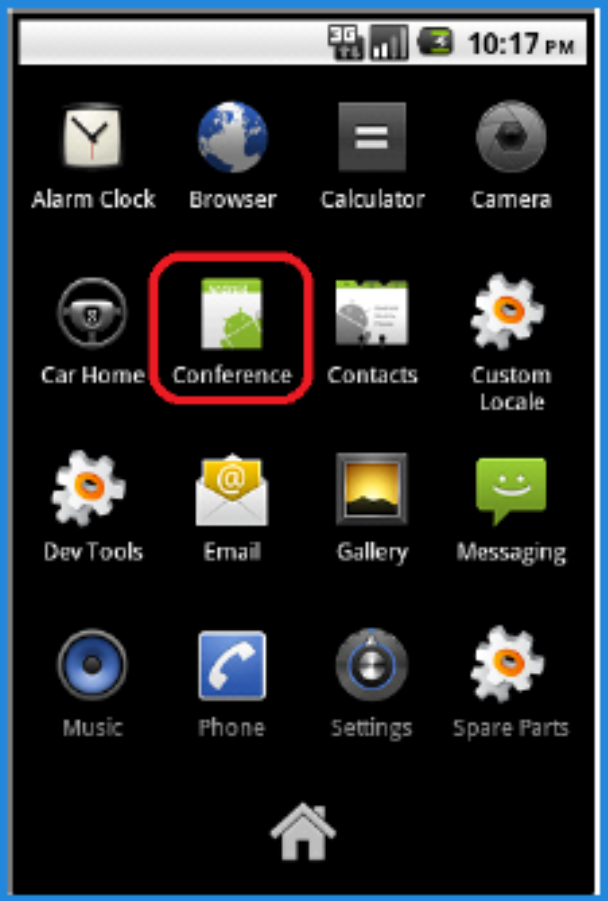


Image: Kaspersky Labs, https://www.securelist.com/en/blog/208194186/Android_Trojan_Found_in_Targeted_Attack, March 26th, 2013

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)

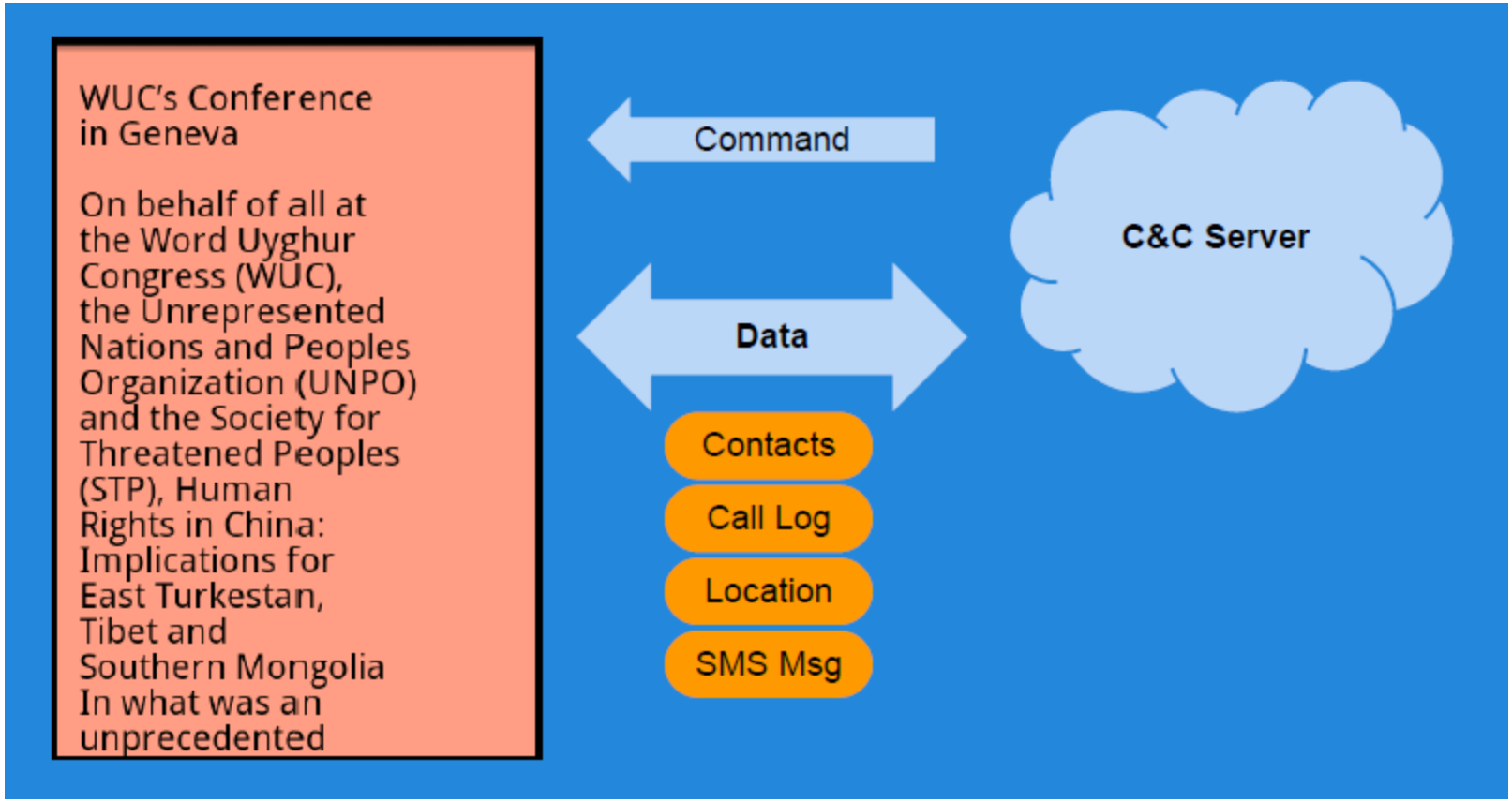
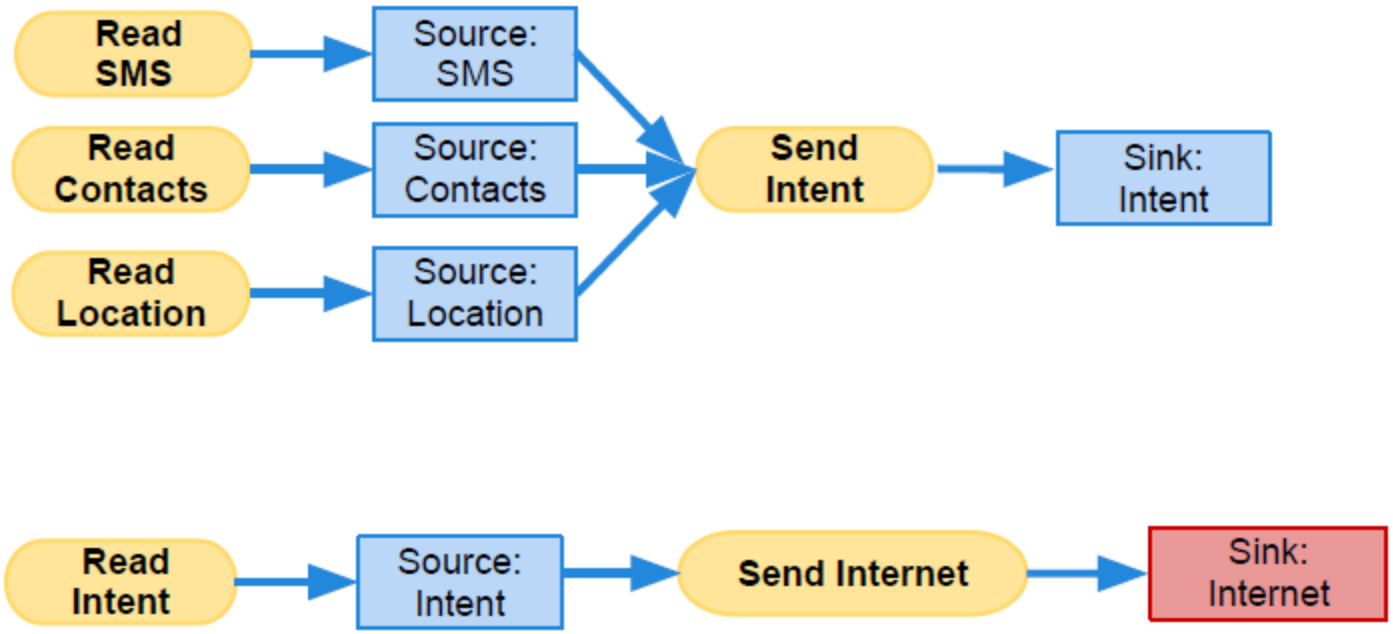


Image: Kaspersky Labs, https://www.securelist.com/en/blog/208194186/Android_Trojan_Found_in_Targeted_Attack, March 26th, 2013

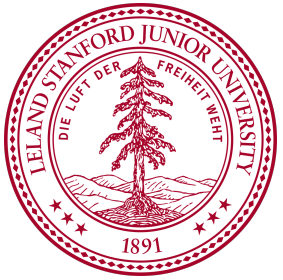
Chuli source-to-sink flows



ANDROID WEB APPS

A Large-Scale Study of Mobile Web App Security

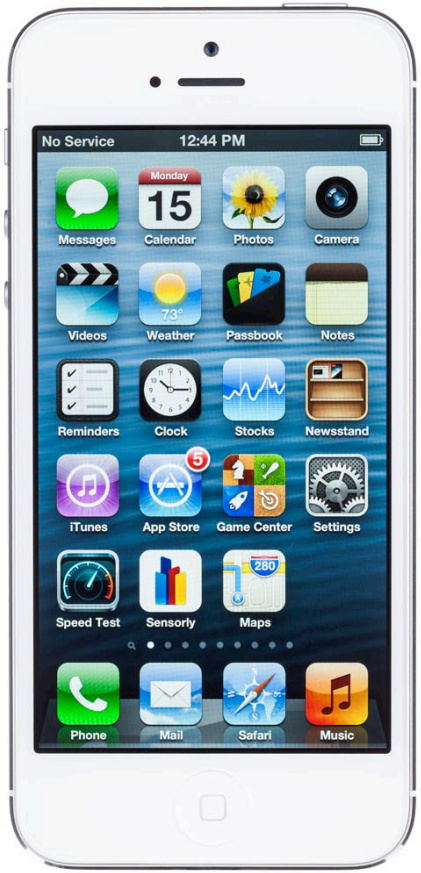
Patrick Mutchler, Adam Doupe,
John Mitchell, Chris Kruegel, Giovanni Vigna



Mobile Apps

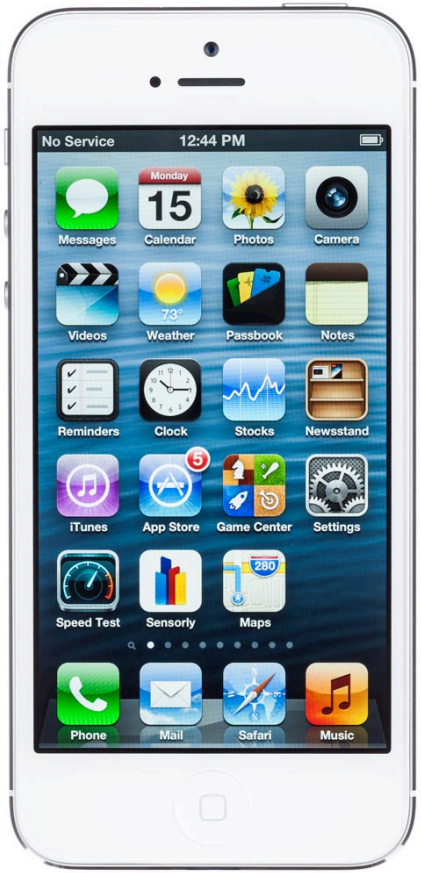
Mobile Apps


android
SAMSUNG


No Service
12:44 PM
Messages
Monday
15
Calendar
Photos
Camera
Videos
73°
Weather
Passbook
Notes
Reminders
Clock
Stocks
Newsstand
iTunes
App Store
Game Center
Settings
Speed Test
Sensorly
280
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
4
Phone
9
1
Photos

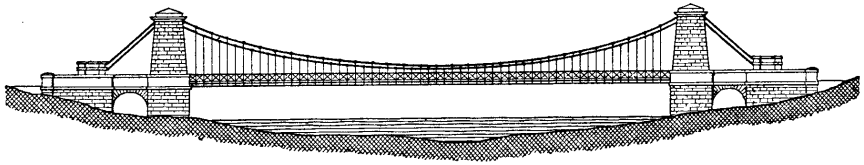
Mobile Web Apps

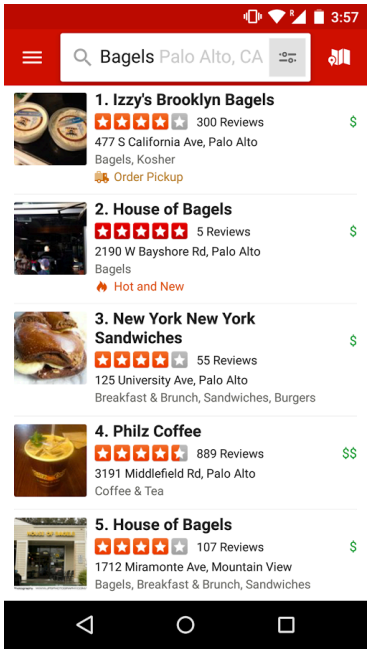
- Mobile web app: embeds a fully functional web browser as a UI element

JavaScript Bridge

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

Java



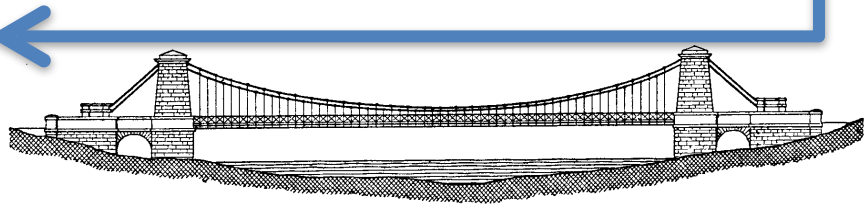


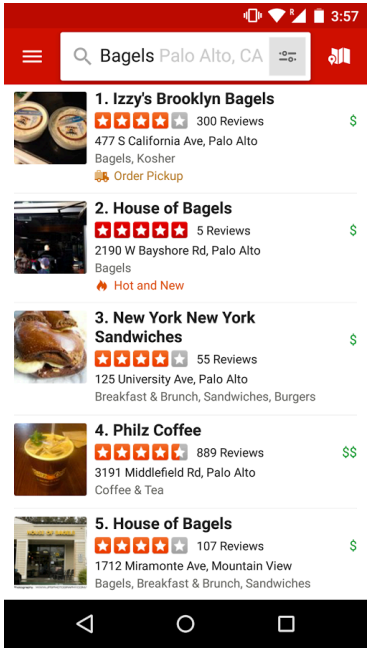
JavaScript

JavaScript Bridge

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

Java





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

Isolated in Browser

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

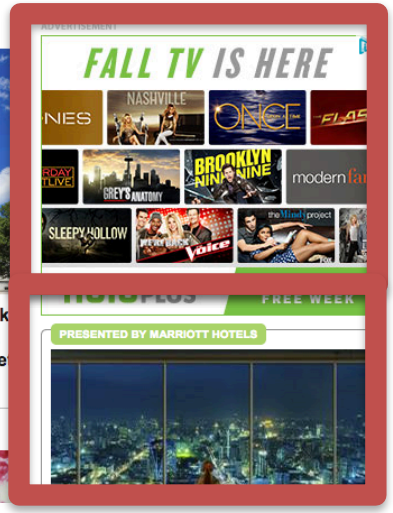
The way we talk about incidents of gun violence in this country -- and the solutions we propose to stem future acts of violence -- seems to be dramatically different depending on the race of those involved. Consider the tragic death of 25 year-old African-American Kajieme Powell in St. Louis this summer. 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

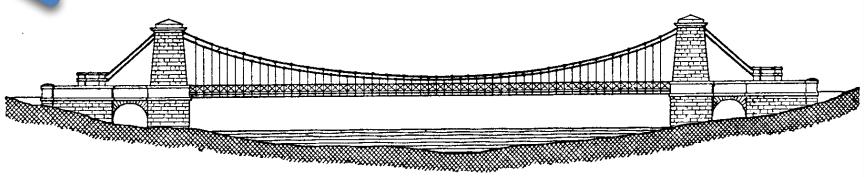
Spanish Nurse With Ebola Worsens



No origin distinction in WebView

```
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


}
```

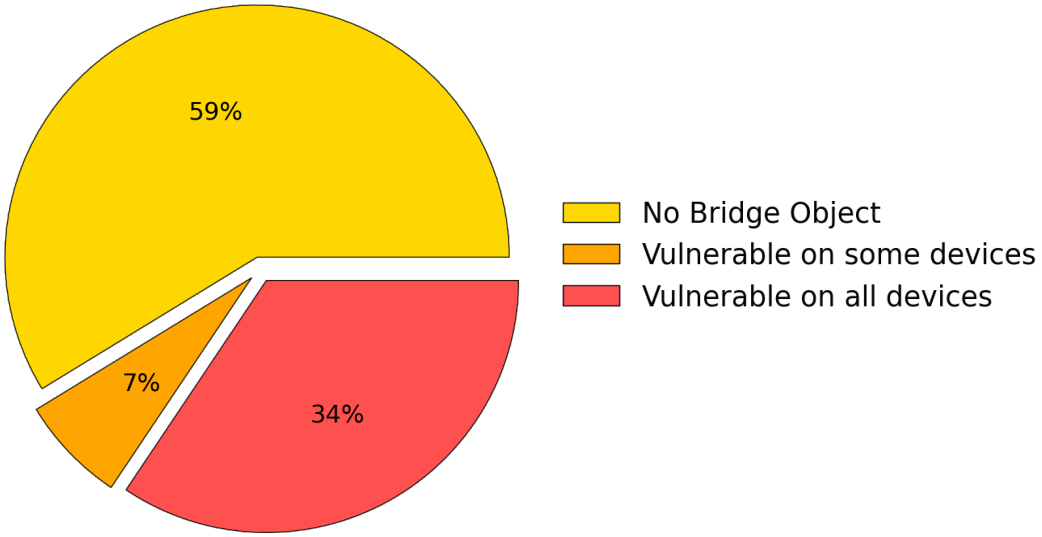
```
public boolean shouldOverrideUrlLoading(
        WebView view, String url){

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

Reach Untrusted Content?

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

Use HTTPS?

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



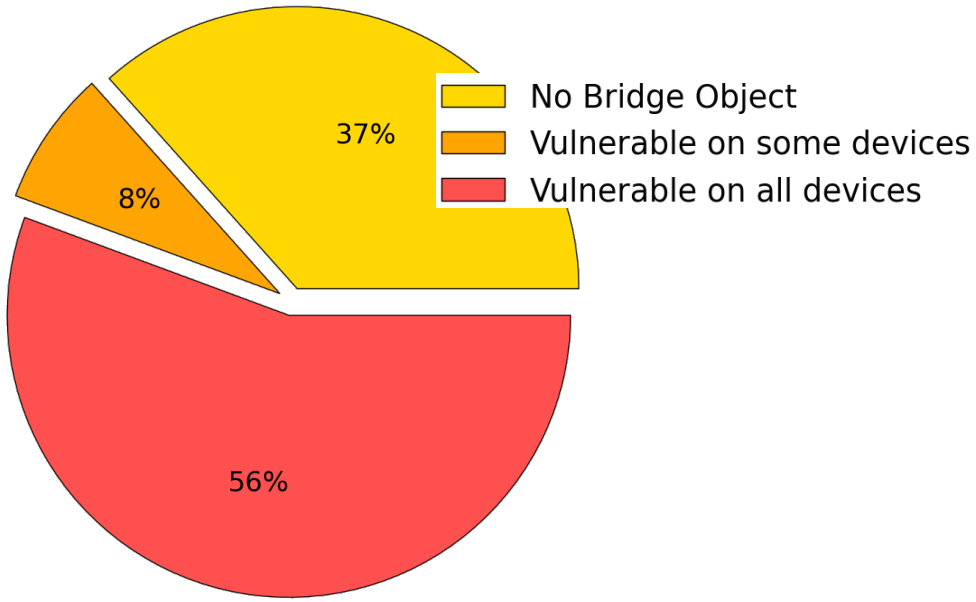
Handling SSL Errors

`onReceivedSslError`

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

Mishandling SSL Errors

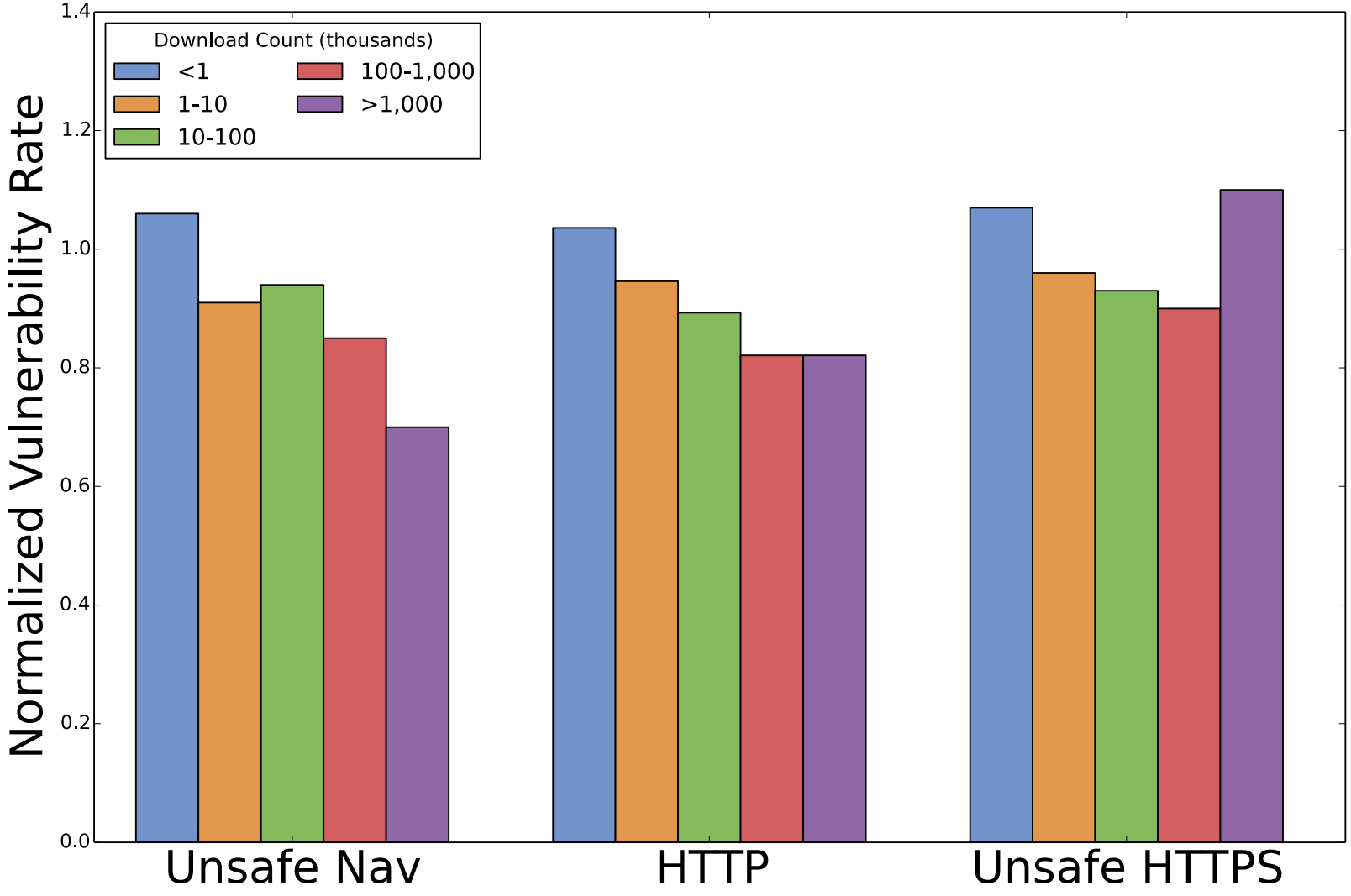
- 117,974 apps implement `onReceivedSslError`
- 29,652 apps (25%) **must** ignore errors



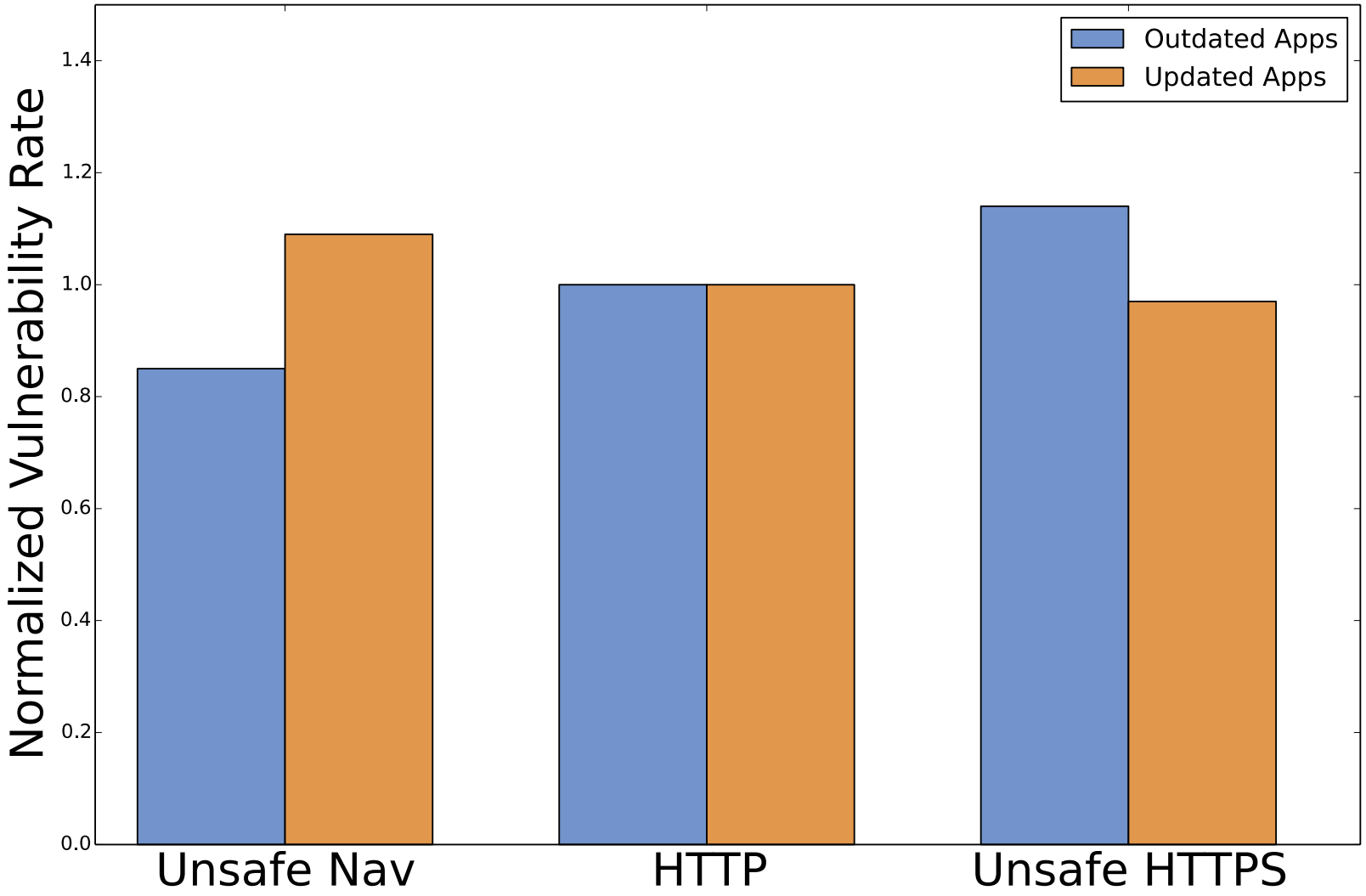
Primary results

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
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
Unsafe Nav
HTTP
Unsafe HTTPS

Outdated Apps



Libraries

29%
unsafe nav

51%
HTTP

53%
unsafe HTTPS

Additional security issues

Based on 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

ANDROID VERSIONING

Target Fragmentation in Android Apps

Patrick Mutchler
John Mitchell

Yeganeh Safaei
Adam Doupe

Takeaways

Android apps can run using outdated OS behavior

- The large majority of Android apps do this
- Including popular and well maintained apps

Outdated security code invisibly permeates the app ecosystem

- “Patched” security vulnerabilities still exist in the wild
- “Risky by default” behavior is widespread

Roadmap

What is target fragmentation?

Target fragmentation statistics

Security consequences

Roadmap

What is target fragmentation?

Target fragmentation statistics

Security consequences

"If the device is running Android 6.0 or higher… [the app] must request each dangerous permission that it needs while the app is running.

- Android Developer Reference

"If the device is running Android 6.0 or higher ***and your app's target SDK is 6.0 or higher*** *[the app] must request each dangerous permission that it needs while the app is running*.

- Android Developer Reference

"If the [operating system version of the device] is higher than the version declared by your app's `targetSdkVersion`*, the system* ***may enable compatibility behaviors*** *to ensure that your app continues to work the way you expect."*

- Android Developer Reference

Roadmap

What is target fragmentation?

Target fragmentation statistics

Security consequences

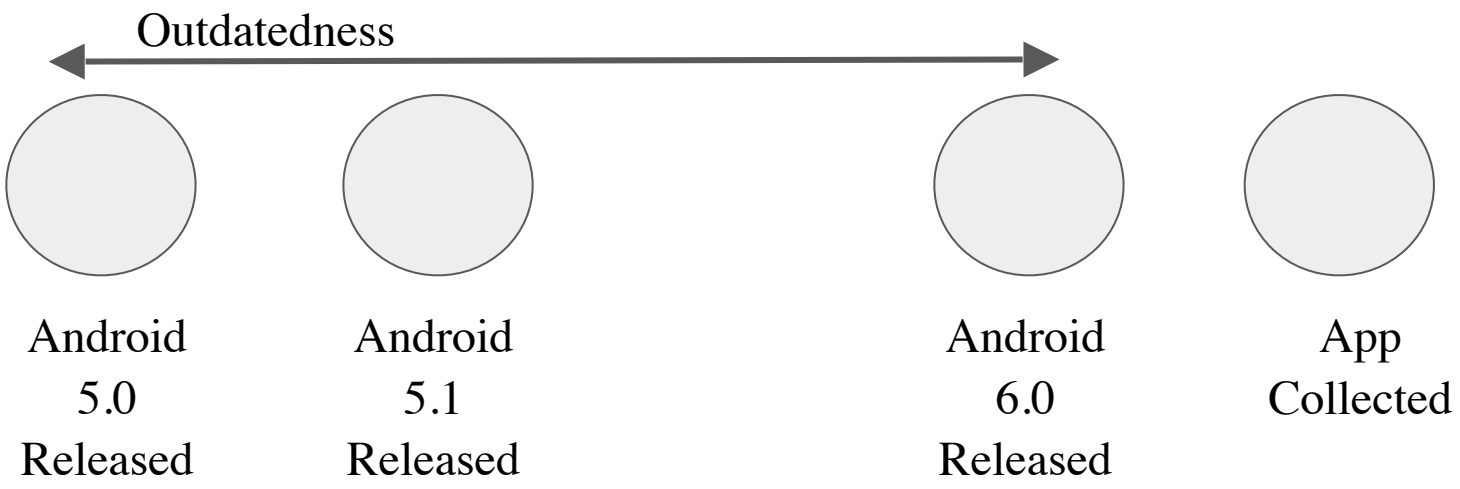
Dataset

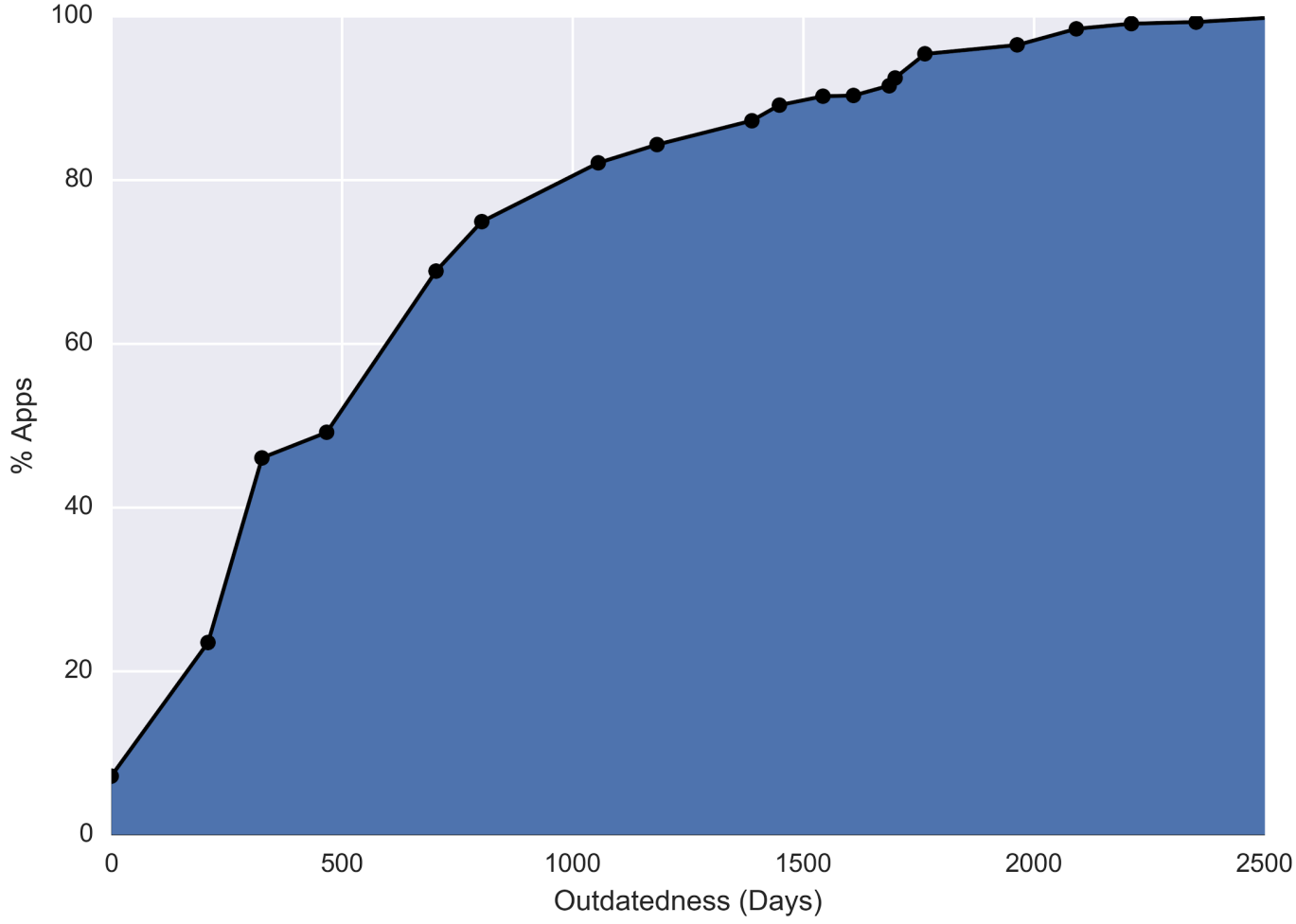
1,232,696 Android Apps

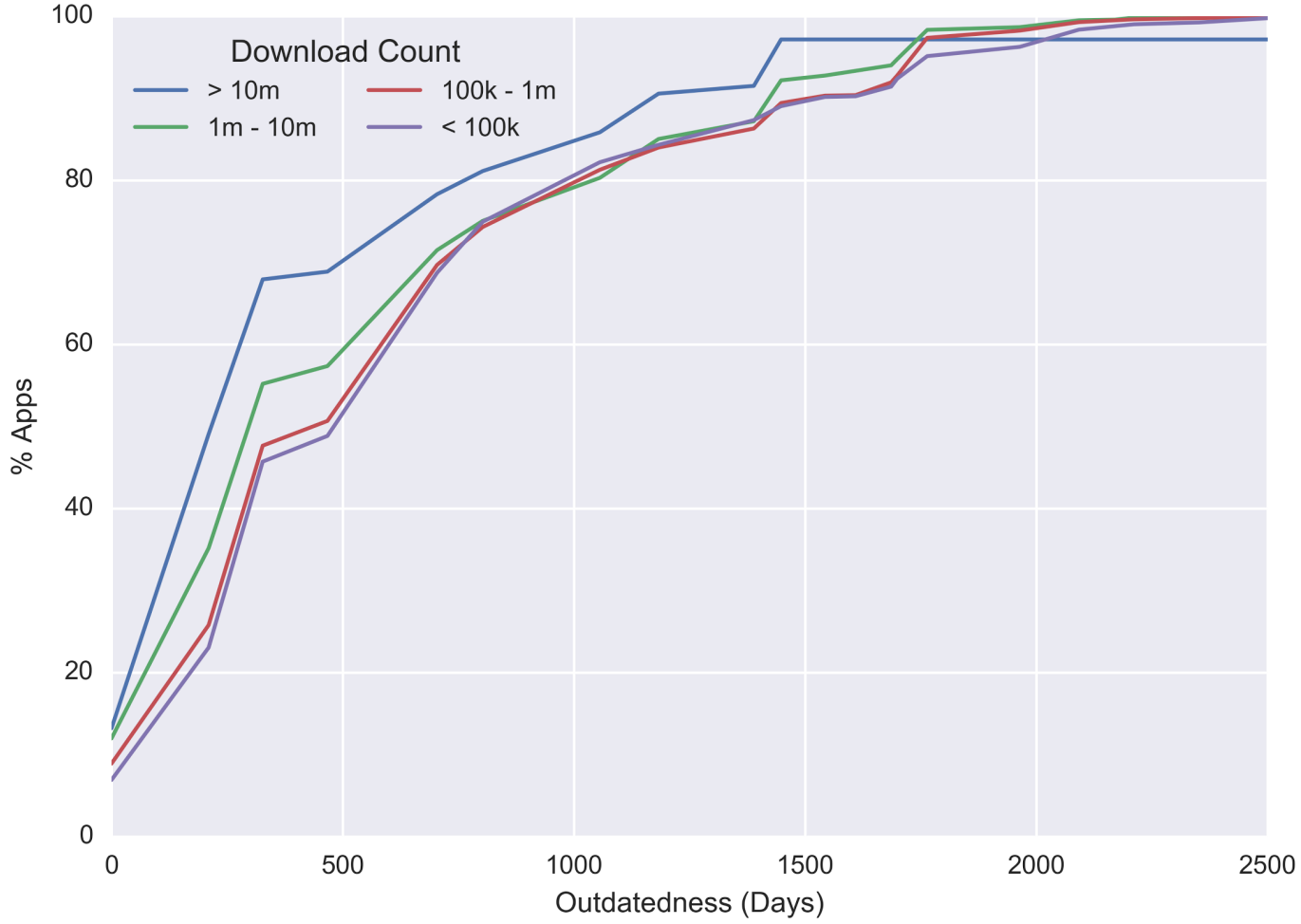
Popularity, Category, Update, and Developer metadata

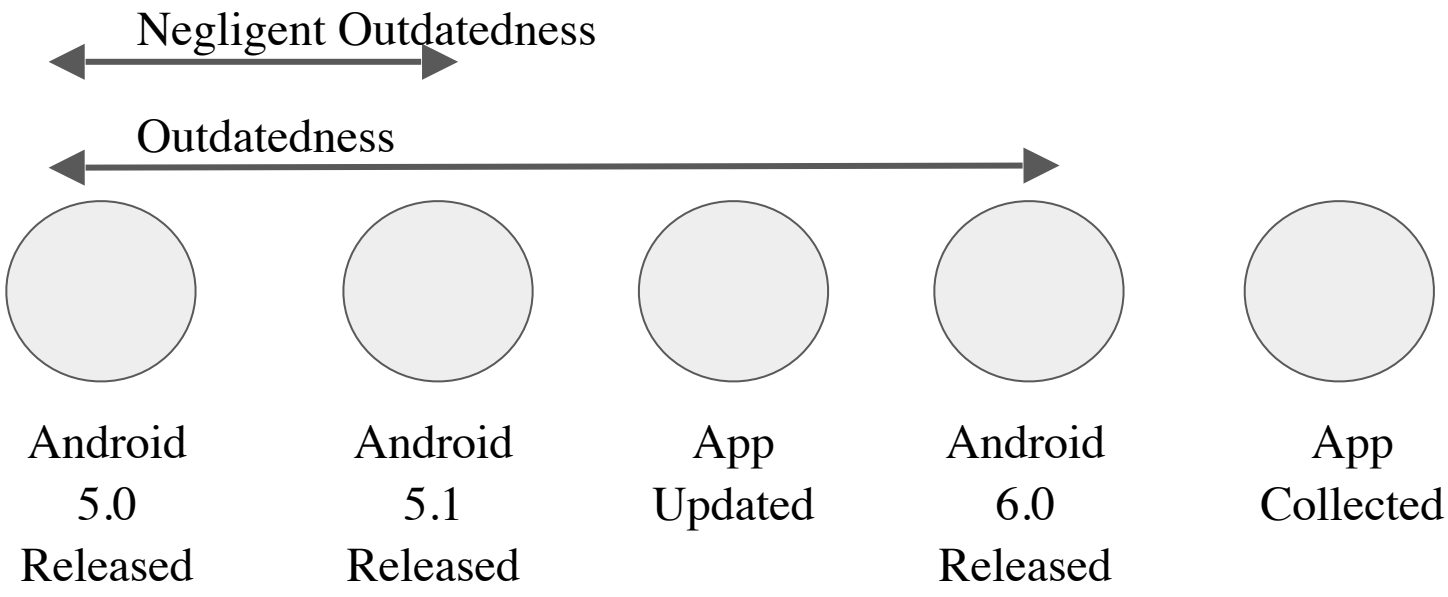
Collected between May 2012 and Dec 2015

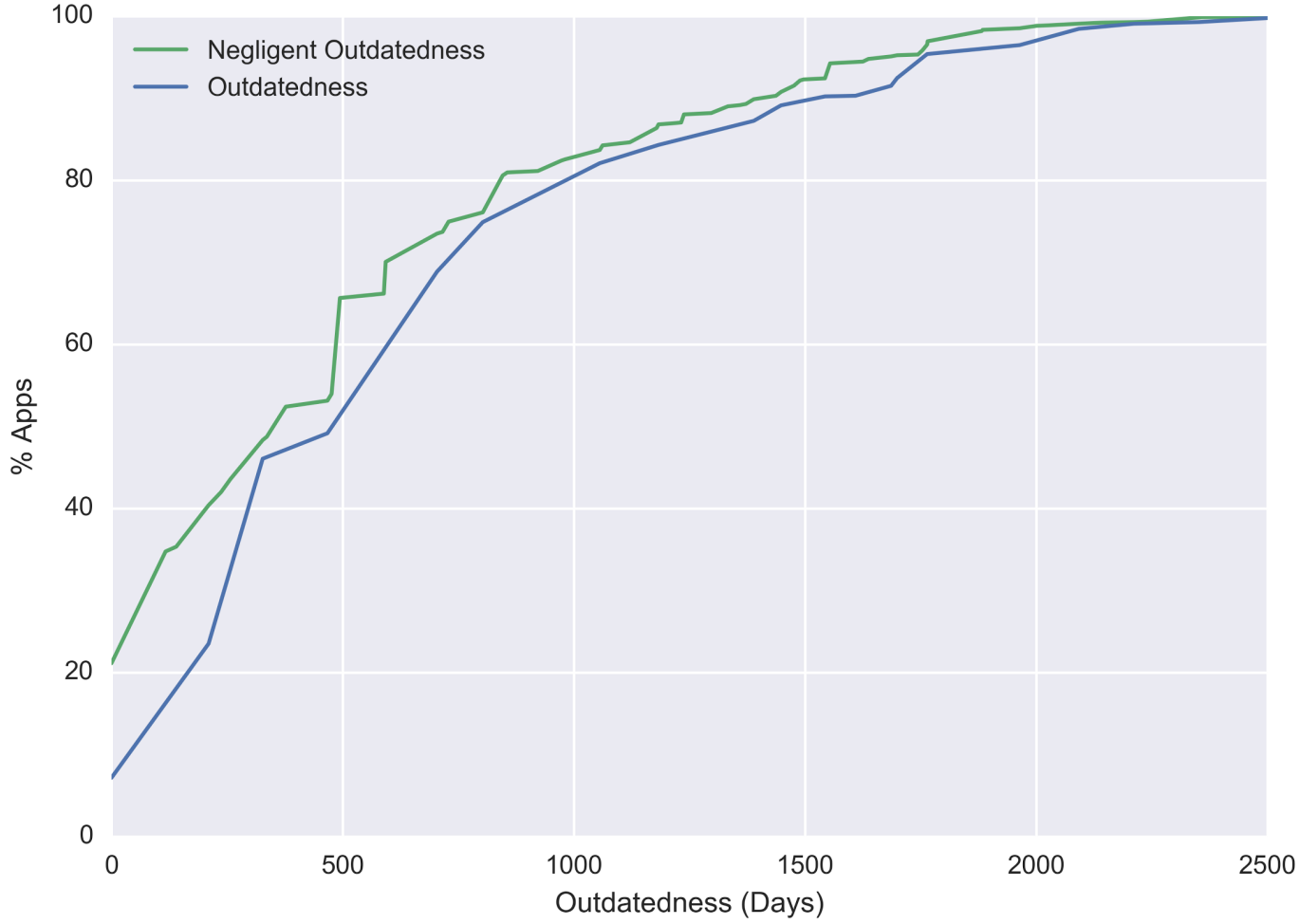
Broken into five datasets by collection date


Outdatedness
Android 5.0 Released
Android 5.1 Released
Android 6.0 Released
App Collected


100
80
60
40
20
0
% Apps
0
500
1000
1500
2000
2500
Outdatedness (Days)


Download Count
> 10m
1m - 10m
100k - 1m
< 100k
% Apps
Outdatedness (Days)
0
20
40
60
80
100
0
500
1000
1500
2000
2500


Negligent Outdatedness
Outdatedness
Android 5.0 Released
Android 5.1 Released
App Updated
Android 6.0 Released
App Collected


Negligent Outdatedness
Outdatedness
100
80
60
40
20
0
% Apps
0
500
1000
1500
2000
2500
Outdatedness (Days)

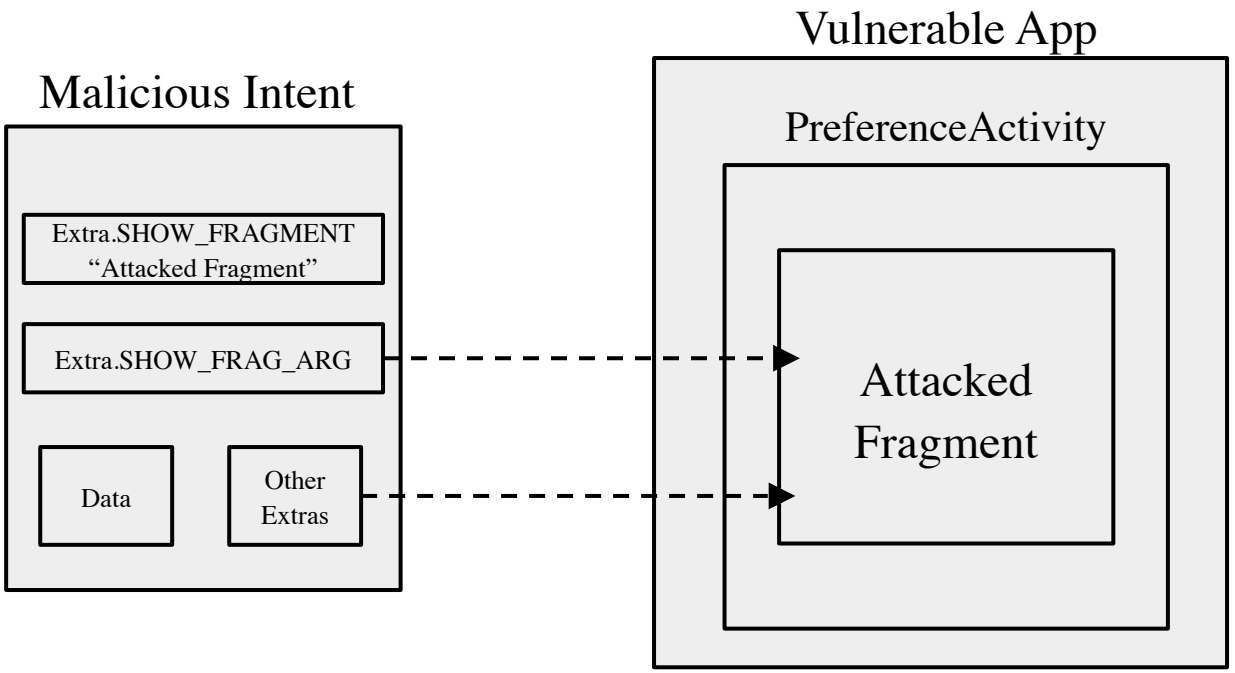
Roadmap

What is target fragmentation?

Target fragmentation statistics

Security consequences

Fragment Injection



A malicious application can invoke any exported *PreferenceActivity* class and supply it with an *:android:show_fragment* Intent extra in order to make it load an arbitrary class.

securityintelligence.com/new-vulnerability-android-framework-fragment-injection/

Fragment Injection

Fixed in Android 4.4

Developers implement `isValidFragment` to authorize fragments

```
// Put this in your app
protected boolean isValidFragment(String fName){
      return MyFrag.class.getName().equals(fName);
}
```

Fragment Injection

Vulnerable if:

- Targets 4.3 or lower (31%)
- Some class inherits from `PreferenceActivity` (4.8%)
- That class is exported (1.1%)
- That class does not override `isValidFragment` (0.55%)

4.2% of apps vulnerable if no fix was ever implemented

Mixed Content in WebView

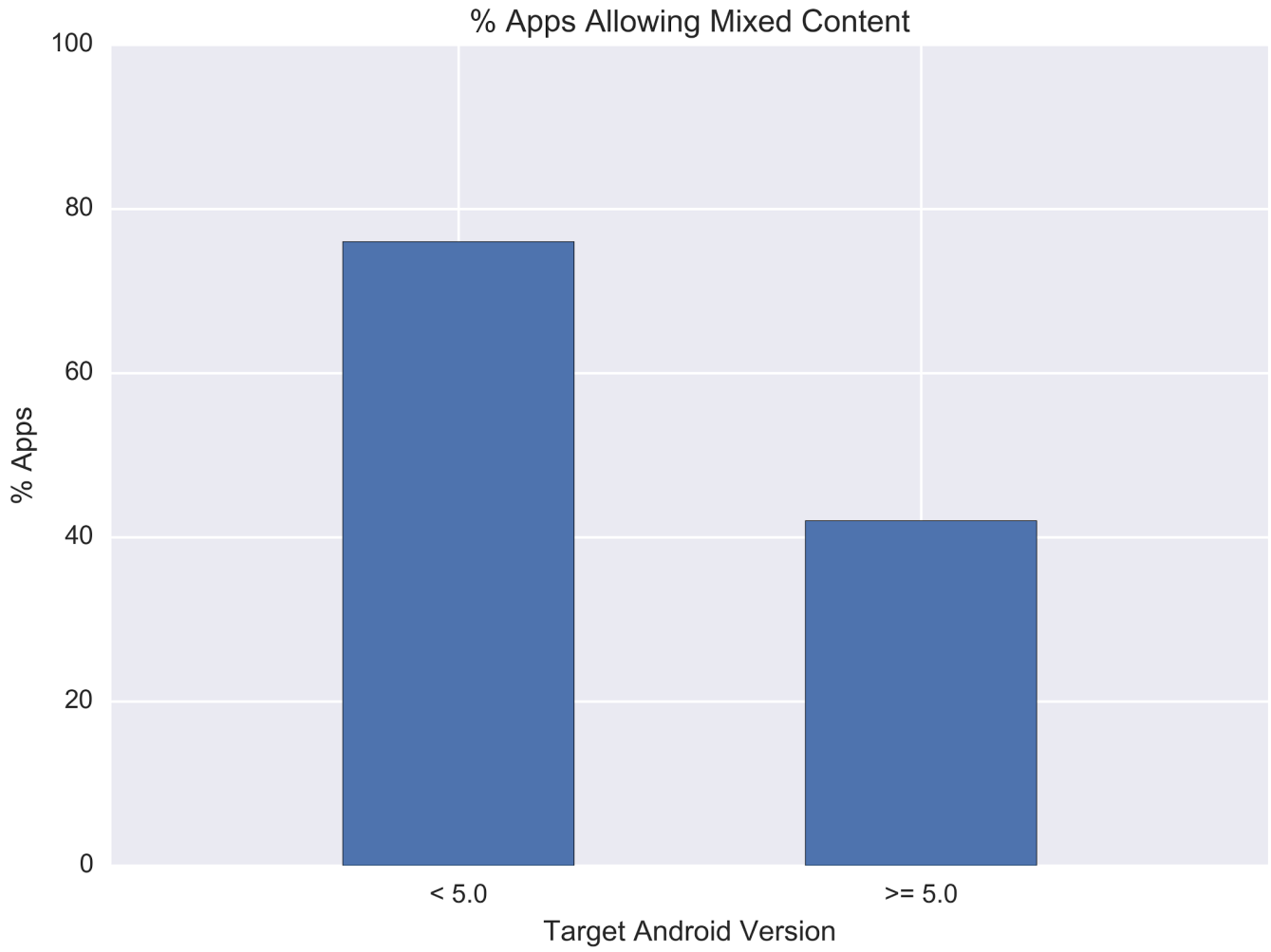
```
Mixed Content: The page at                                    simple-example.html:1
'https://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/simple-example.html' was loaded over HTTPS, but
requested an insecure script 'http://googlesamples.github.io/web-
fundamentals/samples/discovery-and-distribution/avoid-mixed-content/simple-
example.js'. This request has been blocked; the content must be served over HTTPS.
```

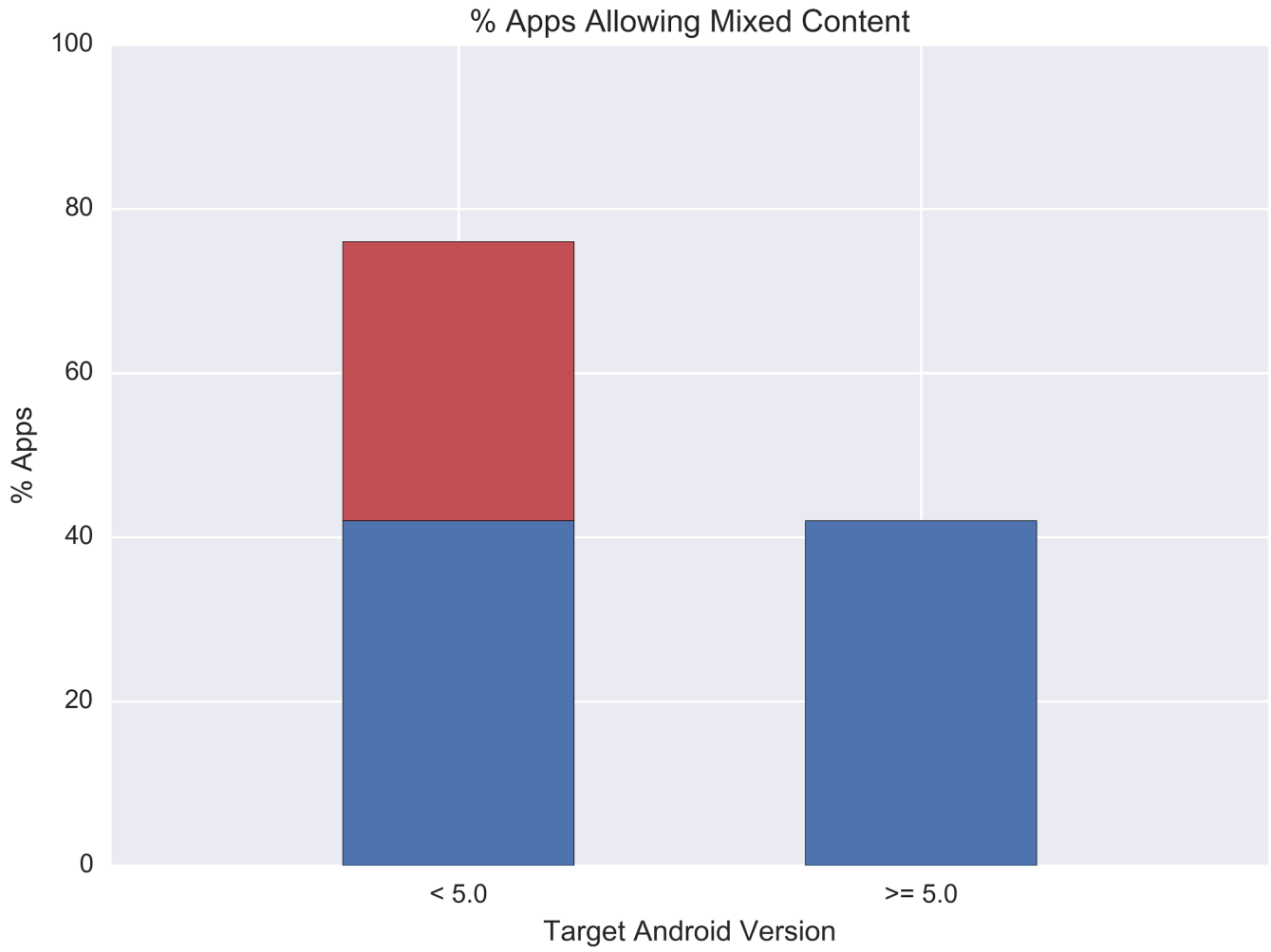
Mixed Content in WebView

Major web browsers block Mixed Content

In Android 5.0, WebViews block Mixed Content by default

Can override default with `setMixedContentMode()`

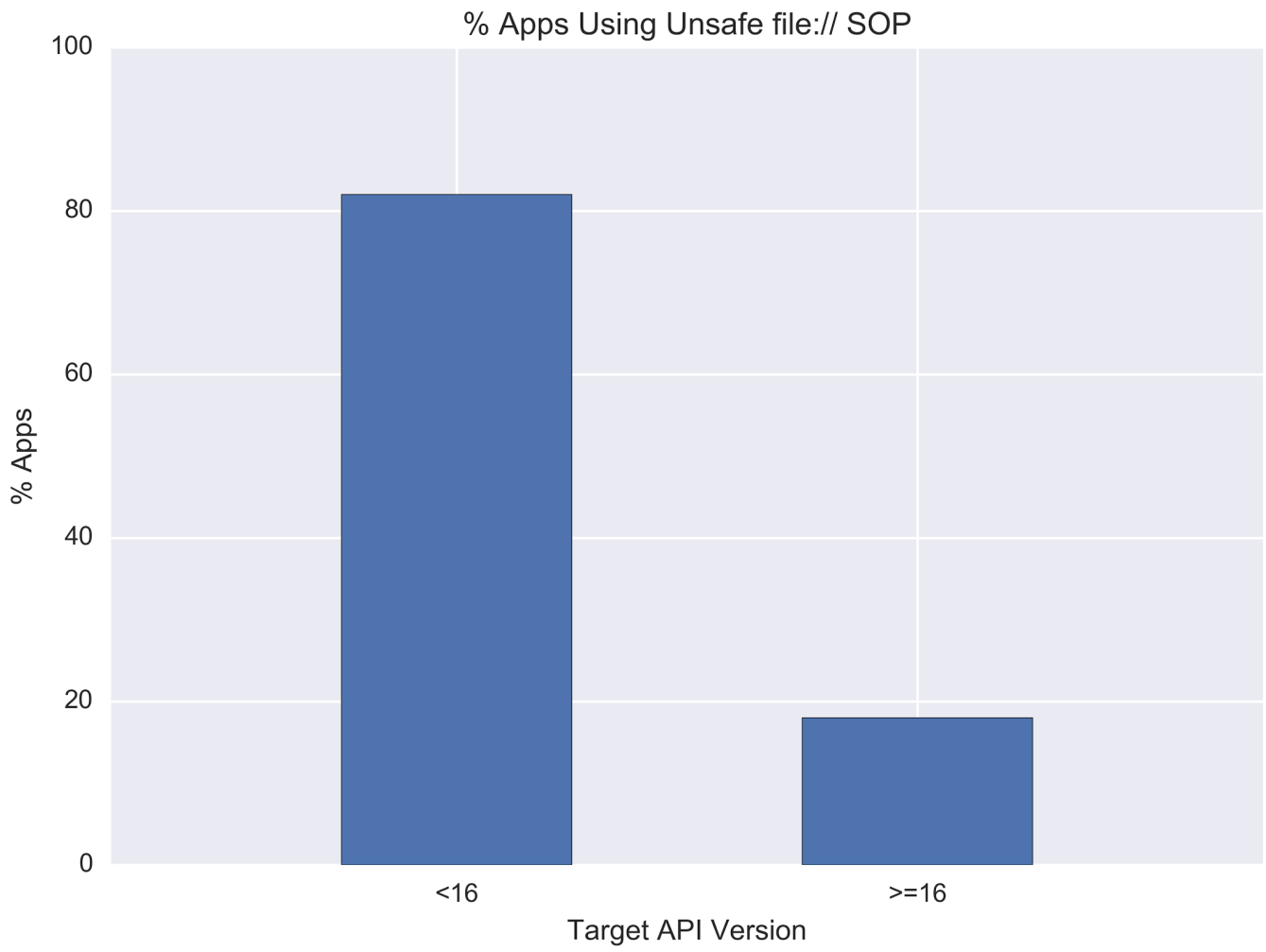

% Apps Allowing Mixed Content
100
80
60
40
20
0
% Apps
< 5.0
>= 5.0
Target Android Version

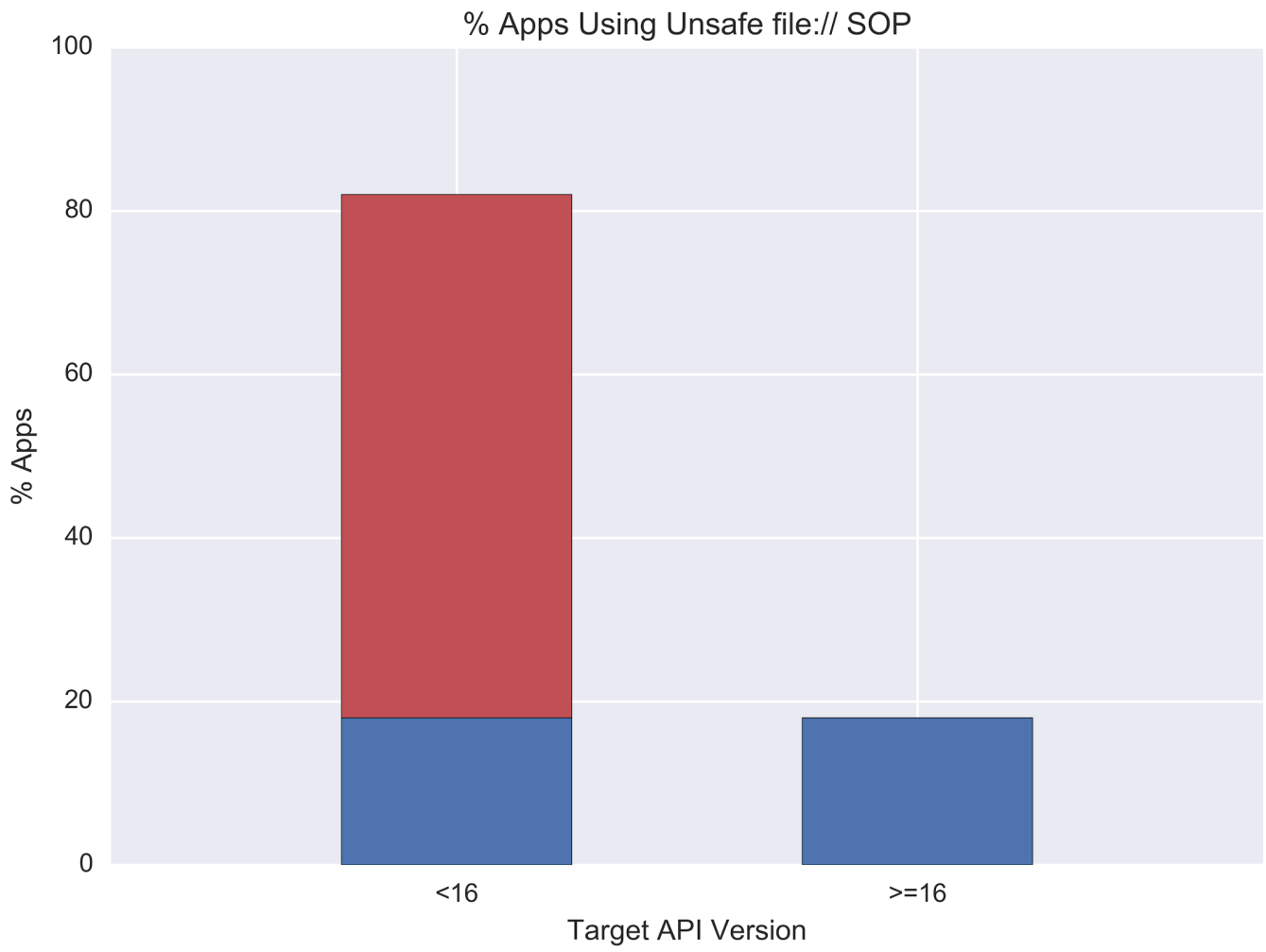

% Apps Allowing Mixed Content
100
80
60
40
20
0
% Apps
< 5.0
>= 5.0
Target Android Version

SOP for `file://` URLs in WebView

Android 4.1 separate `file://` URLs are treated as unique origins

Can override with `setAllowFileAccessFromFileURLs()`


% Apps Using Unsafe file:// SOP
100
80
60
40
20
0
% Apps
<16
>=16
Target API Version


% Apps Using Unsafe file:// SOP
100
80
60
40
20
0
% Apps
<16
>=16
Target API Version

Summary of Target Fragmentation

Android apps can run using outdated OS behavior

- The large majority of Android apps do this
- Including popular and well maintained apps

Outdated security code invisibly permeates the app ecosystem

- “Patched” security vulnerabilities still exist in the wild
- “Risky by default” behavior is widespread

Two lectures on mobile security

- Introduction: platforms and trends
- Threat categories
  - Physical, platform malware, malicious apps
- Defense against physical theft
- Malware threats
- System architecture and defenses
  - Apple iOS security features and app security model
  - Android security features and app security model
- Security app development
  - WebView – secure app and web interface dev
  - Device fragmentation

Tues

Thurs

Comparison: iOS vs Android

- App approval process
  - Android apps from open app store
  - iOS vendor-controlled store of vetted apps
- Application permissions
  - Android permission based on install-time manifest
  - All iOS apps have same set of “sandbox” privileges
- App programming language
  - Android apps written in Java; no buffer overflow…
  - iOS apps written in Objective-C

Comparison

	iOS	Android	Windows
Unix	x	x	
Windows			
Open market		x	
Closed market	x		
Vendor signed	x		
Self-signed		x	
User approval of permissions		x	
Managed code		x	
Native code	x		

Comparison

	iOS	Android	Windows
Unix	x	x	
Windows			x
Open market		x	
Closed market	x		x
Vendor signed	x		
Self-signed		x	x
User approval of permissions		x	7-> 8
Managed code		x	x
Native code	x		

Two lectures on mobile security

- Introduction: platforms and trends
- Threat categories
  - Physical, platform malware, malicious apps
- Defense against physical theft
- Malware threats
- System architecture and defenses
  - Apple iOS security features and app security model
  - Android security features and app security model
- Security app development
  - WebView – secure app and web interface dev
  - Device fragmentation

Tues

Thurs