CS 6431

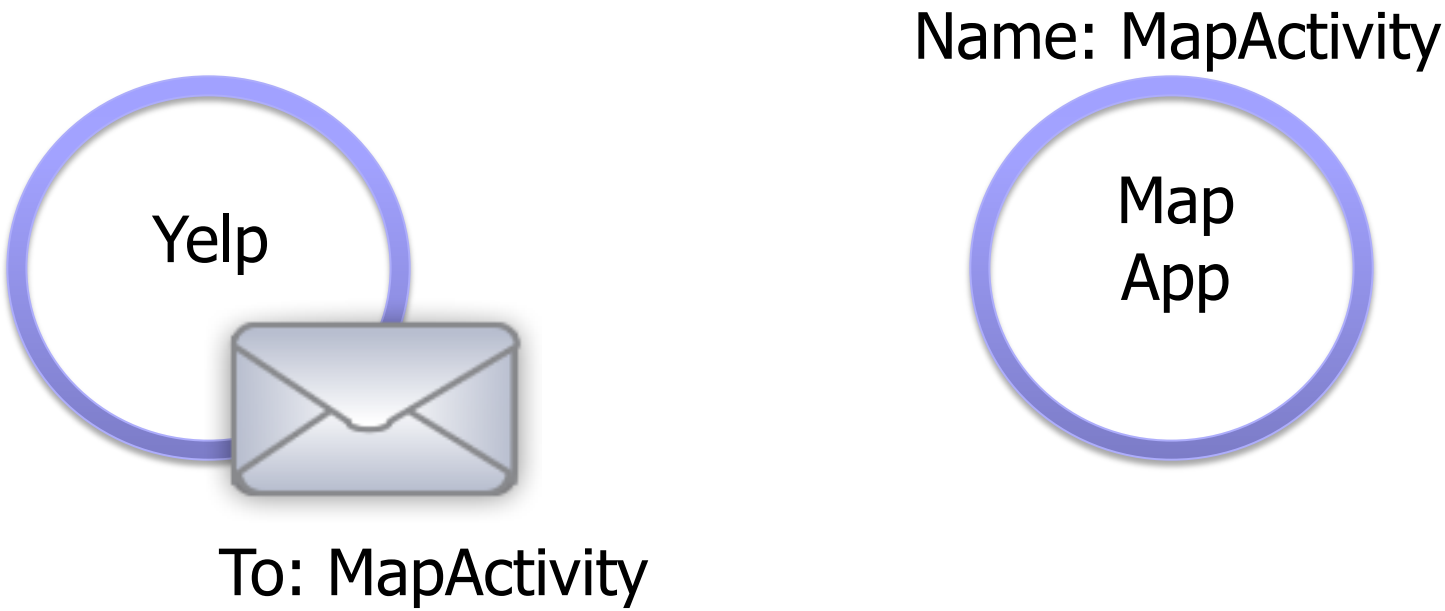
Security of Mobile Applications

Vitaly Shmatikov

Structure of Android Applications

- This is a very brief and incomplete summary
  - See Enck et al. "Understanding Android Security"
- Applications include multiple components
  - Activities: user interface
  - Services: background processing
  - Content providers: data storage
  - Broadcast receivers for messages from other apps
- Intent: primary messaging mechanism for interaction between components

Explicit Intents



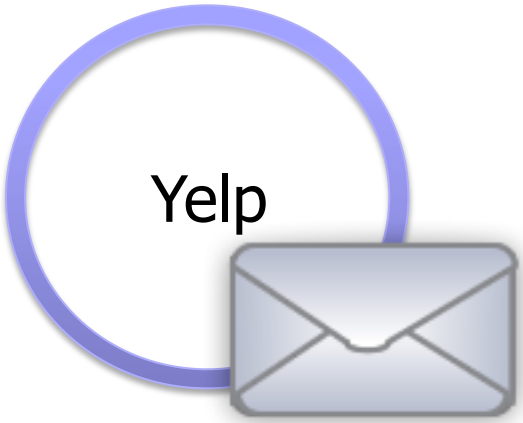
Only the specified destination receives this message

Implicit Intents

Handles Action: VIEW

Map App

Yelp



Implicit Intent
Action: VIEW

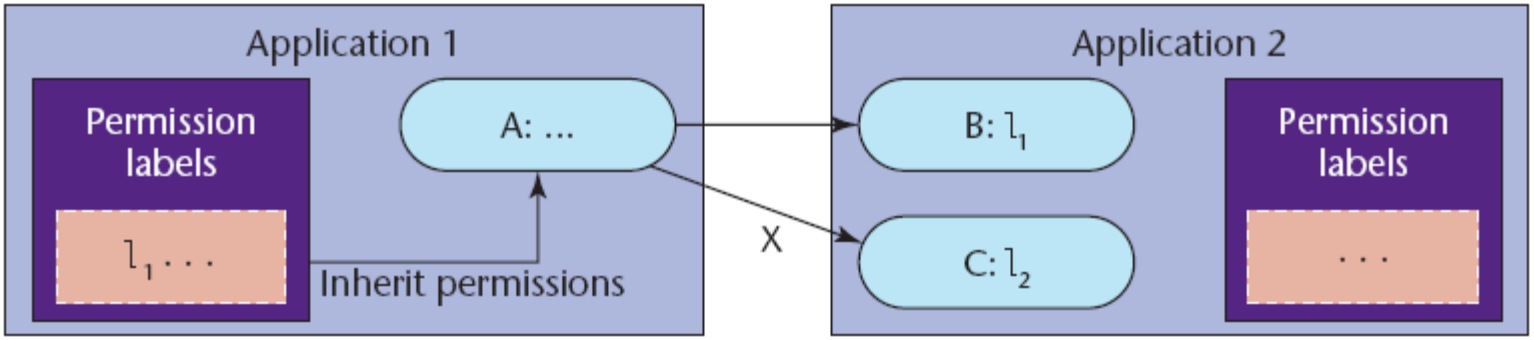
Handles Action: VIEW

Browser App

Android Security Model

Access permitted if labels assigned to the invoked component are in the collection of invoking component

- Based on permission labels assigned to applications and components



- Every app runs as a separate user
  - Underlying Unix OS provides system-level isolation
- Reference monitor in Android middleware mediates inter-component communication

Mandatory Access Control

- Permission labels are set (via manifest) when app is installed and cannot be changed
- Permission labels only restrict access to components, they do not control information flow – means what?
- Apps may contain "private" components that should never be accessed by another app (example?)
- If a public component doesn't have explicit permissions listed, it can be accessed by any app

System API Access

- System functionality (eg, camera, networking) is accessed via Android API, not system components
- App must declare the corresponding permission label in its manifest + user must approve at the time of app installation
- Signature permissions are used to restrict access only to certain developers
  - Ex: Only Google apps can directly use telephony API

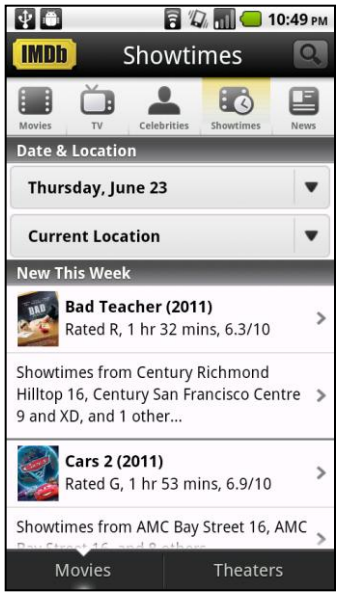
Refinements

- Permission labels on broadcast intents
  - Prevents unauthorized apps from receiving these intents – why is this important?
- Pending intents
  - Instead of directly performing an action via intent, create an object that can be passed to another app, thus enabling it to execute the action
  - Invocation involves RPC to the original app
  - Introduces delegation into Android's MAC system

Unique Action Strings

Common developer pattern

IMDb App



Handles Actions:
willUpdateShowtimes,
showtimesNoLocationError

Implicit Intent
Action: willUpdateShowtimes

Eavesdropping

[Felt et al. "Analyzing Inter-Application Communication in Android". Mobisys 2011]

IMDb App

Showtime Search

Implicit Intent
Action: willUpdateShowtimes

Eavesdropping App

Handles Action:
willUpdateShowtimes,
showtimesNoLocationError

Malicious Receiver

Intent Spoofing

[Felt et al.]

Malicious Injection App

Malicious Component

Action:
showtimesNoLocationError

IMDb App

Handles Action:
willUpdateShowtimes,
showtimesNoLocationError

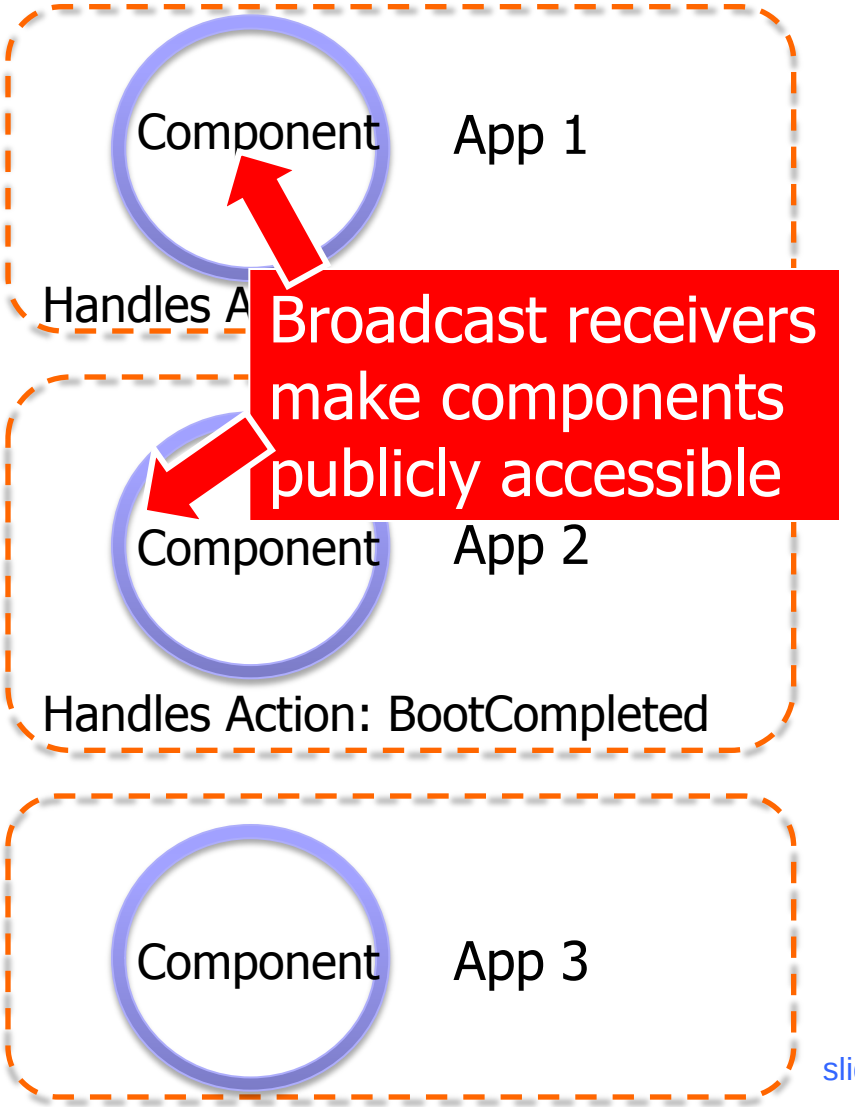
Results UI

Also man-in-the-middle

System Broadcast

[Felt et al.]

Event notifications broadcast by the system (can't be spoofed)



Exploiting Broadcast Receivers

[Felt et al.]

Malicious App

Malicious Component

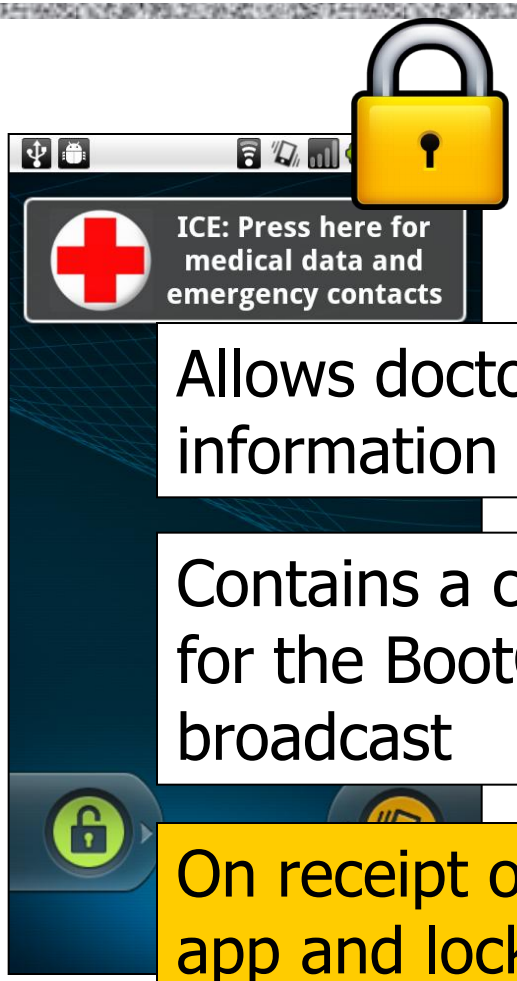
To:
App1.Component

App 1
Handles Action:
BootCompleted

Component

Real World Example: ICE

[Felt et al.]

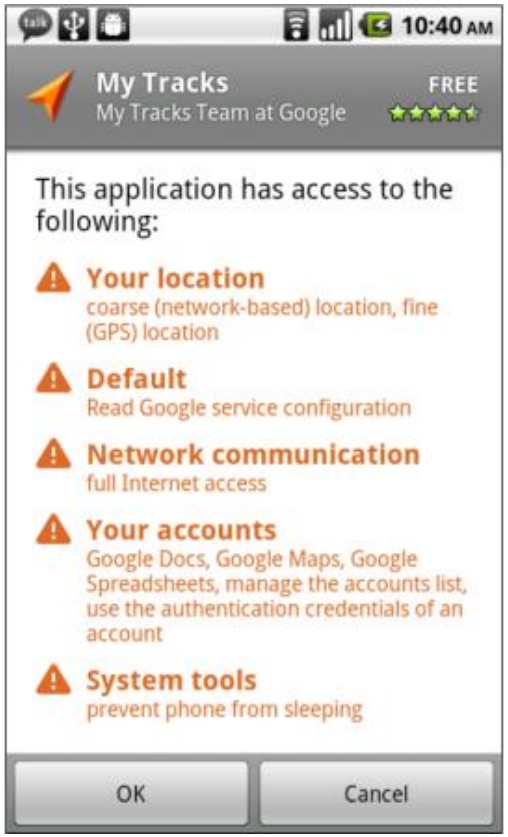


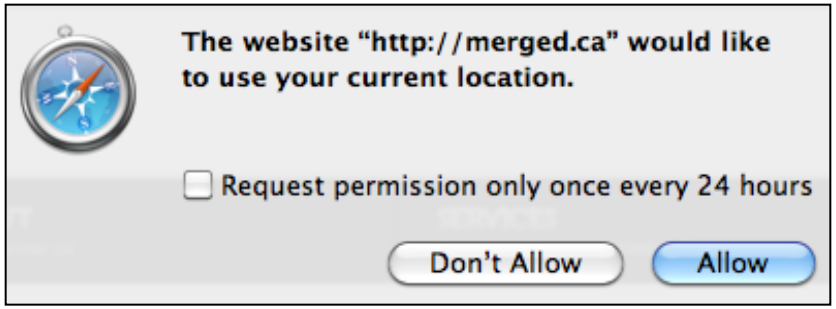
Allows doctors access to medical information on phones

Contains a component that listens for the BootCompleted system broadcast

On receipt of this intent, exits the app and locks the screen

Permissions: Not Just Android



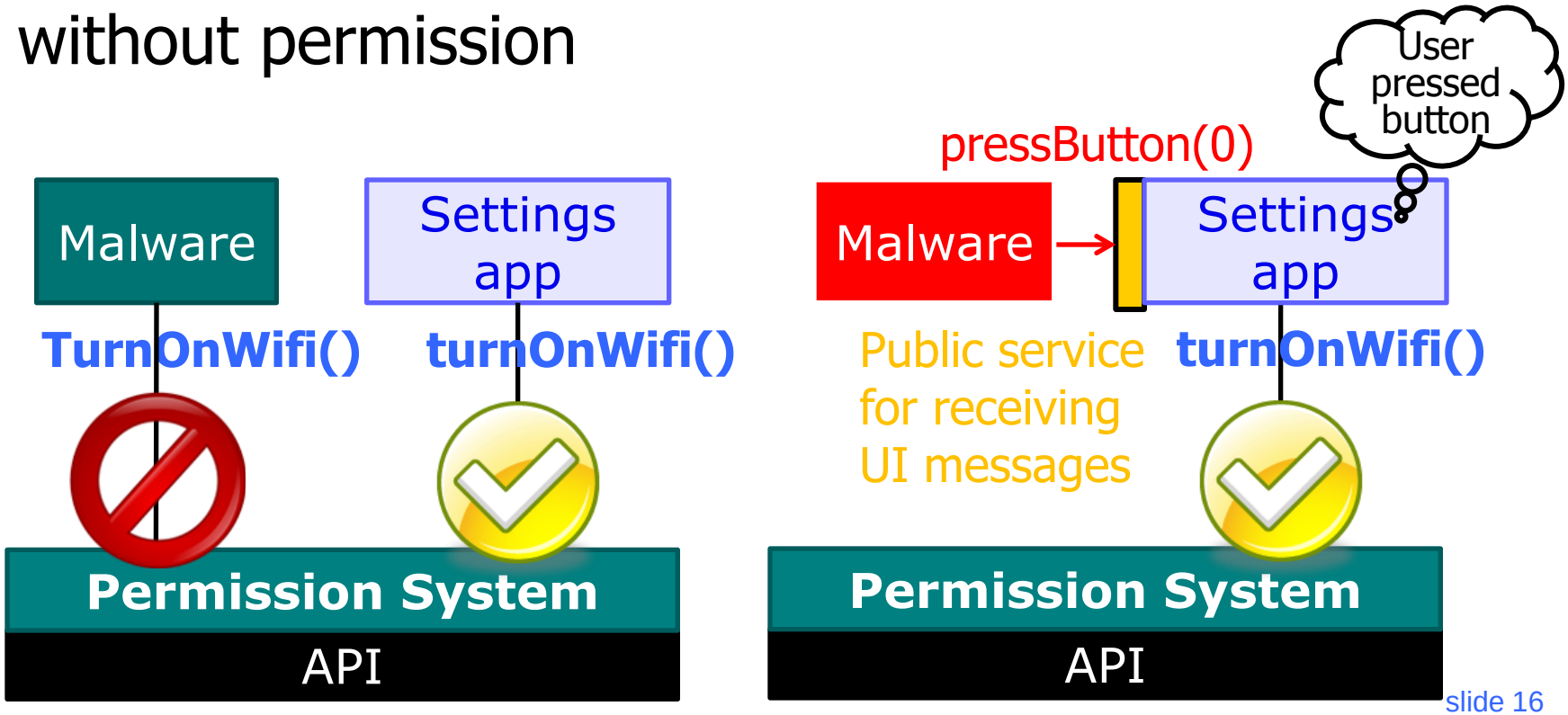


All mobile OSes, HTML5 apps, browser extensions…

Permission Re-Delegation

[Felt et al. "Permission Re-Delegation: Attacks and Defenses". USENIX Security 2011]

- An application with a permission performs a privileged task on behalf of an application without permission

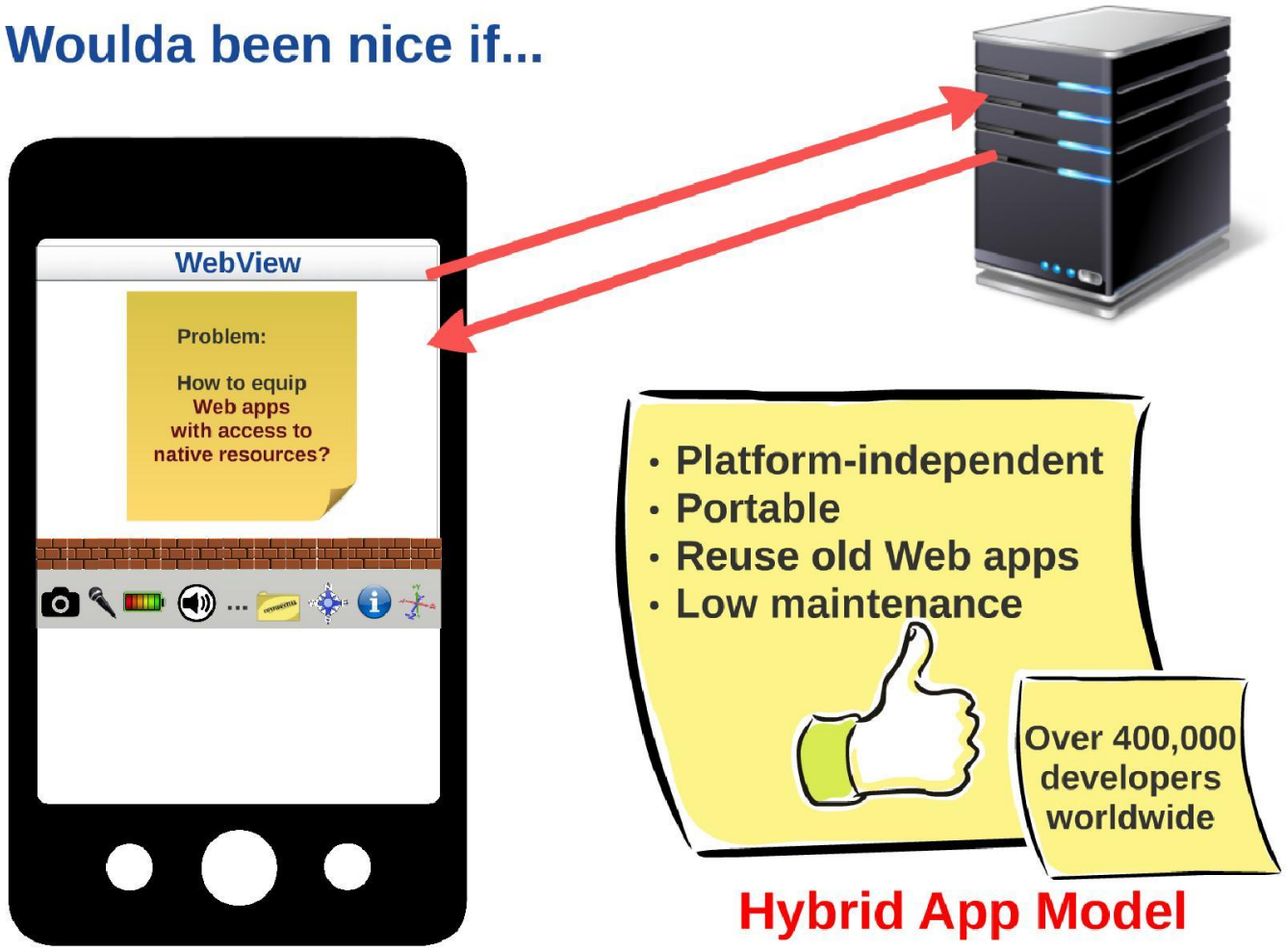


Examples of Re-Delegation

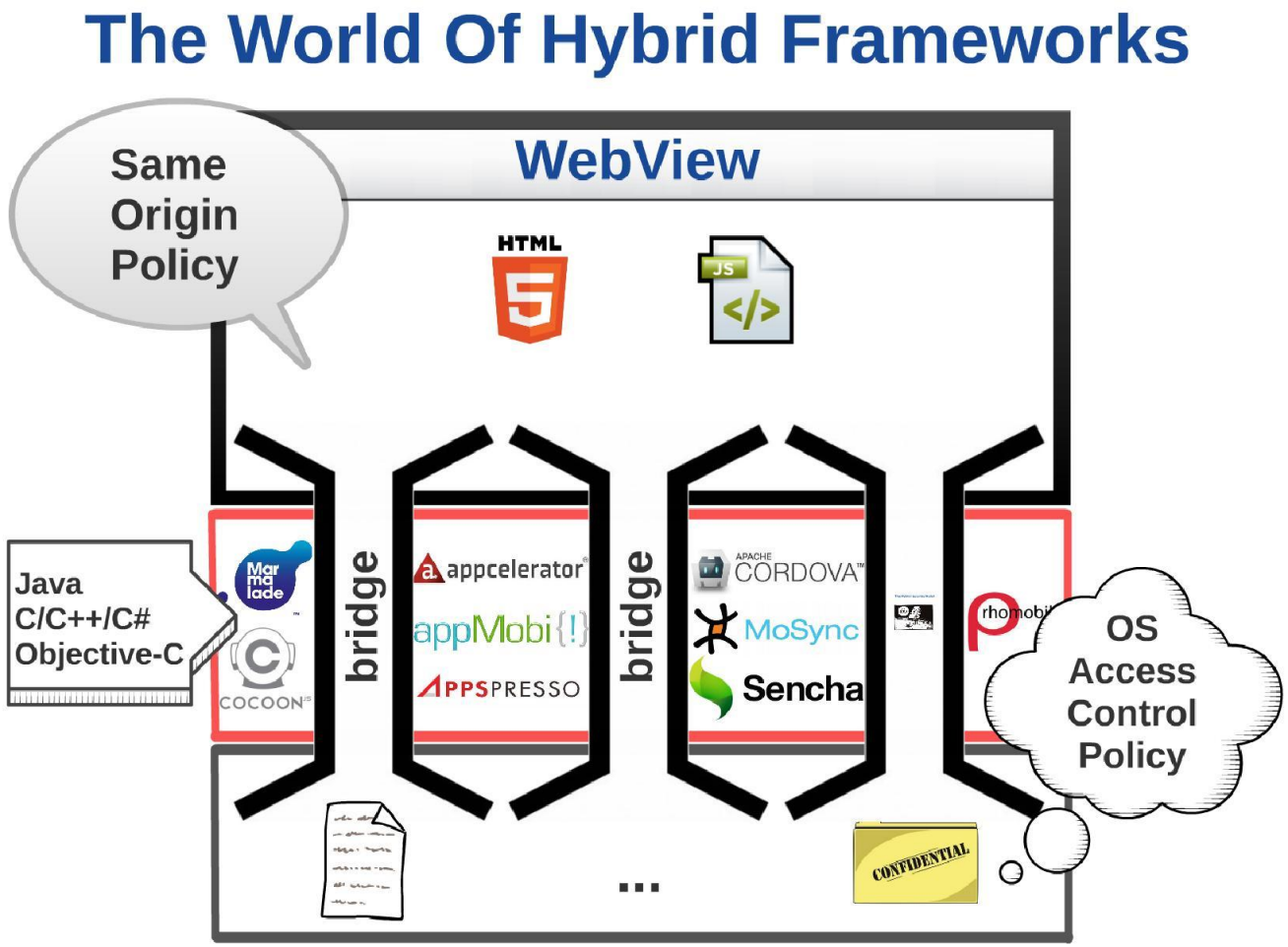
[Felt et al.]

- Permission re-delegation is an example of a "confused deputy" problem
- The "deputy" app may accidentally expose privileged functionality…
- … or intentionally expose it, but the attacker invokes it in a surprising context
  - Example: broadcast receivers in Android
- … or intentionally expose it and attempt to reduce the invoker's authority, but do it incorrectly
  - Remember postMessage origin checks?

Mobile Apps in Web Languages



Hybrid App Development



WebView

[Luo et al. "Attacks on WebView in the Android System". ACSAC 2011]

- Embedded browser in smartphone apps
- Basic same origin policy inside the browser + holes in the browser sandbox allowing Web code to invoke native functionality
  - Camera, contacts, file system, etc.
- Multiple "bridges" between Web and local code
  - JavaScript interfaces to local objects
  - Interception of browser events (eg, special URLs)
  - Other custom and ad-hoc schemes

Invoking Java from JavaScript

[Luo et al.]

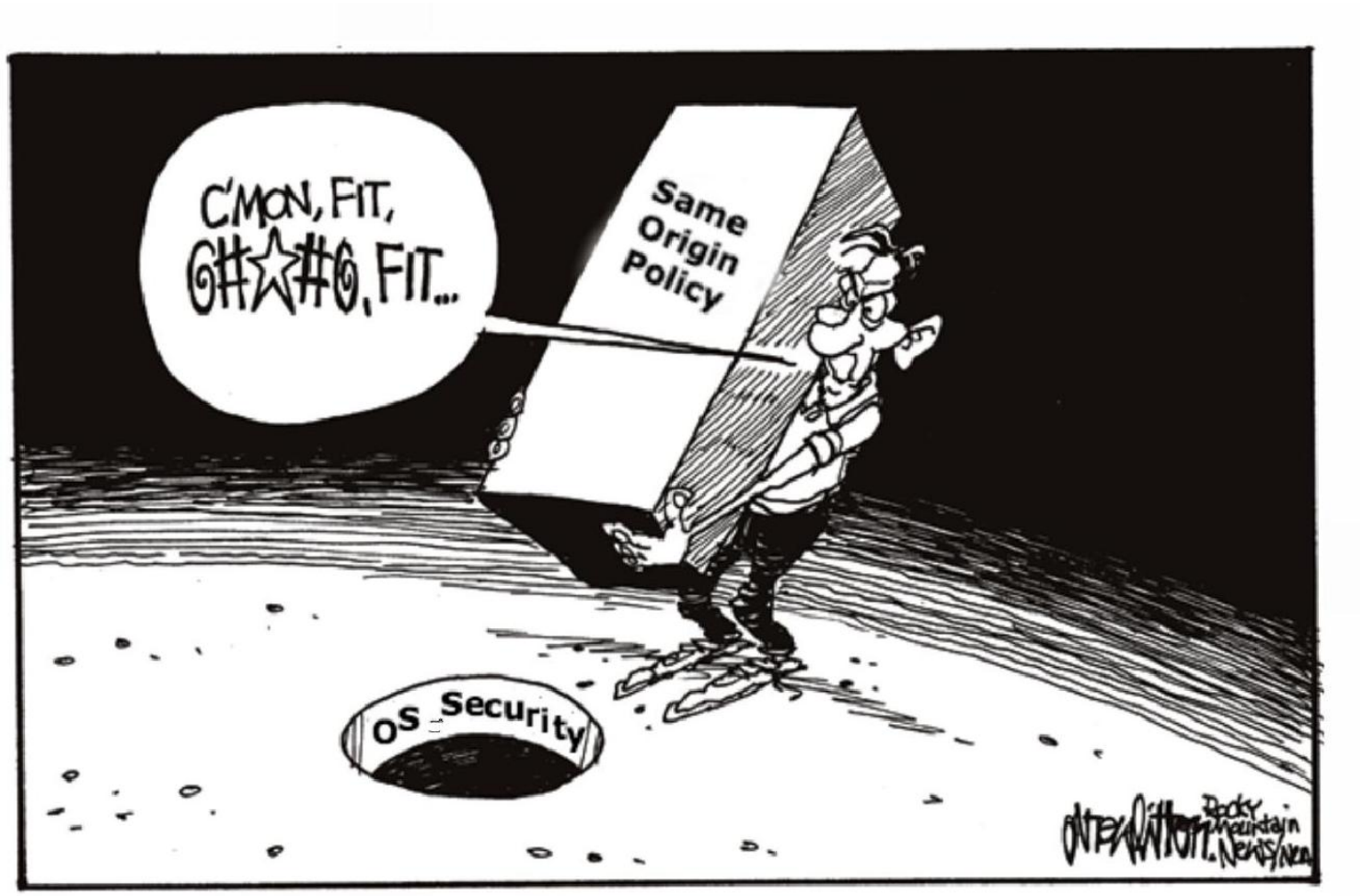
```
wv.addJavascriptInterface(new FileUtils(), "FUtil");
wv.addJavascriptInterface(new ContactManager(), "GC");
...
// The FileUtils class has the following methods:
public int write (String filename, String data, boolean append);
public String read (filename);
...
// The ContactManager class has the following methods:
public void searchPeople (String name, String number);
public ContactTriplet getContactData (String id);
...
```

Invoking JavaScript from Java

[Luo et al.]

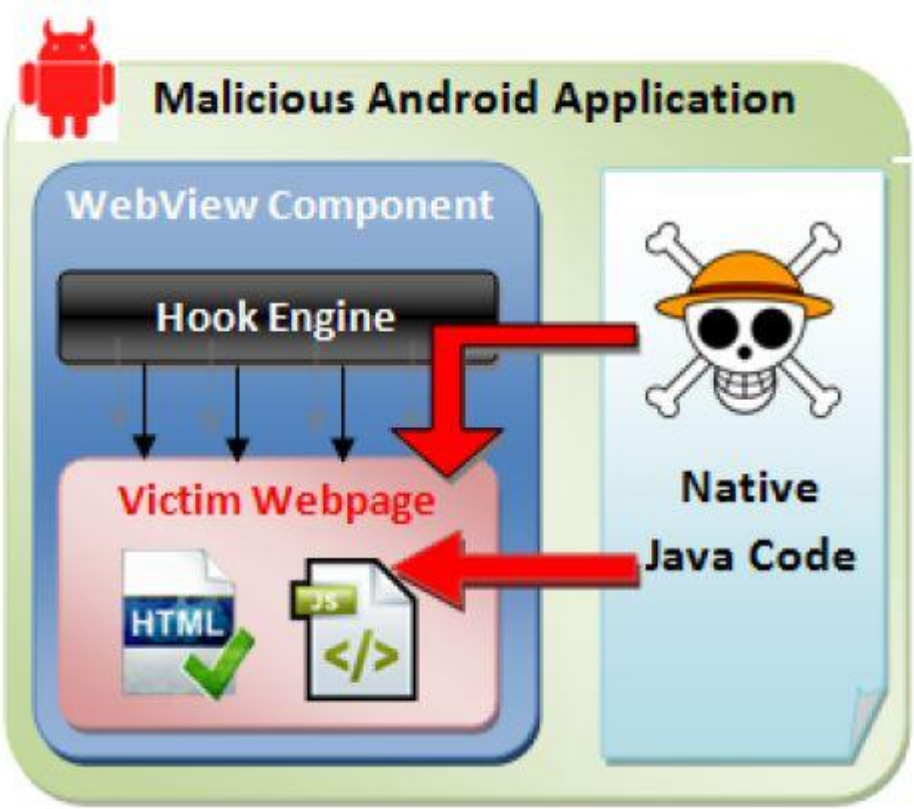
```
String str="<div><h2>Hello World</h2></div>";
webView.loadUrl("javascript:document.appendChild(str);");
webView.loadUrl("javascript:document.cookie='';");
```

The Hybrid Security Model



Attacks from Malicious App

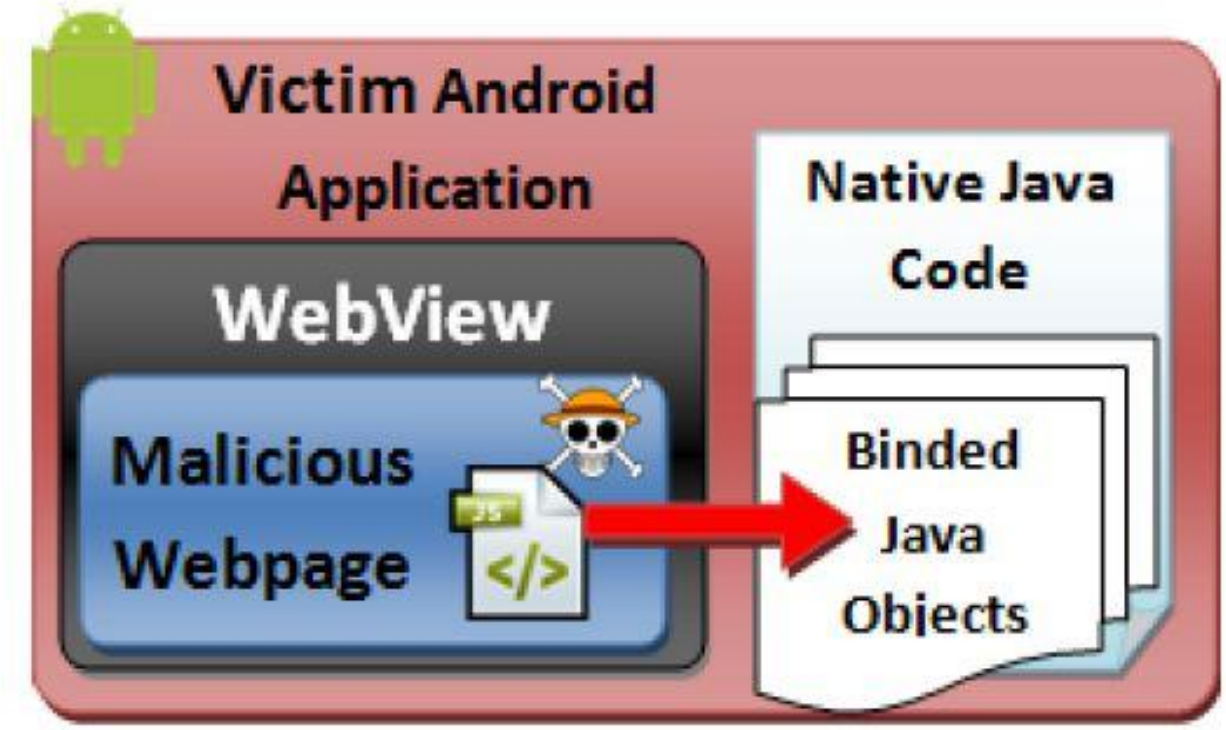
[Luo et al.]



JavaScript injection
Event sniffing and hijacking

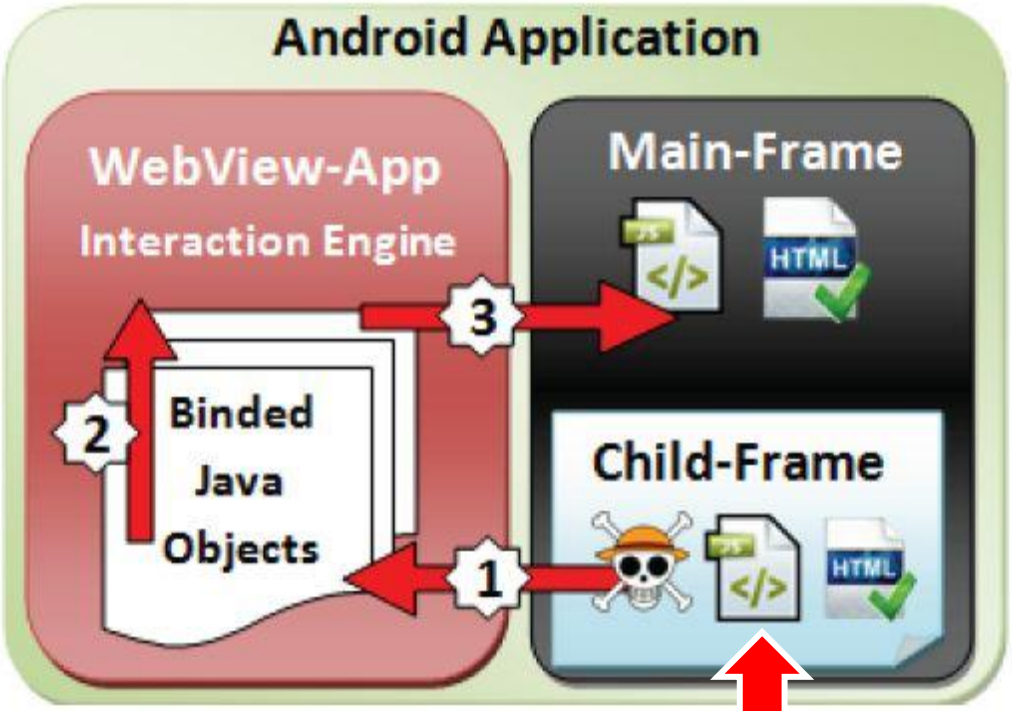
Attack from Malicious Web Content

[Luo et al.]



Frame Confusion

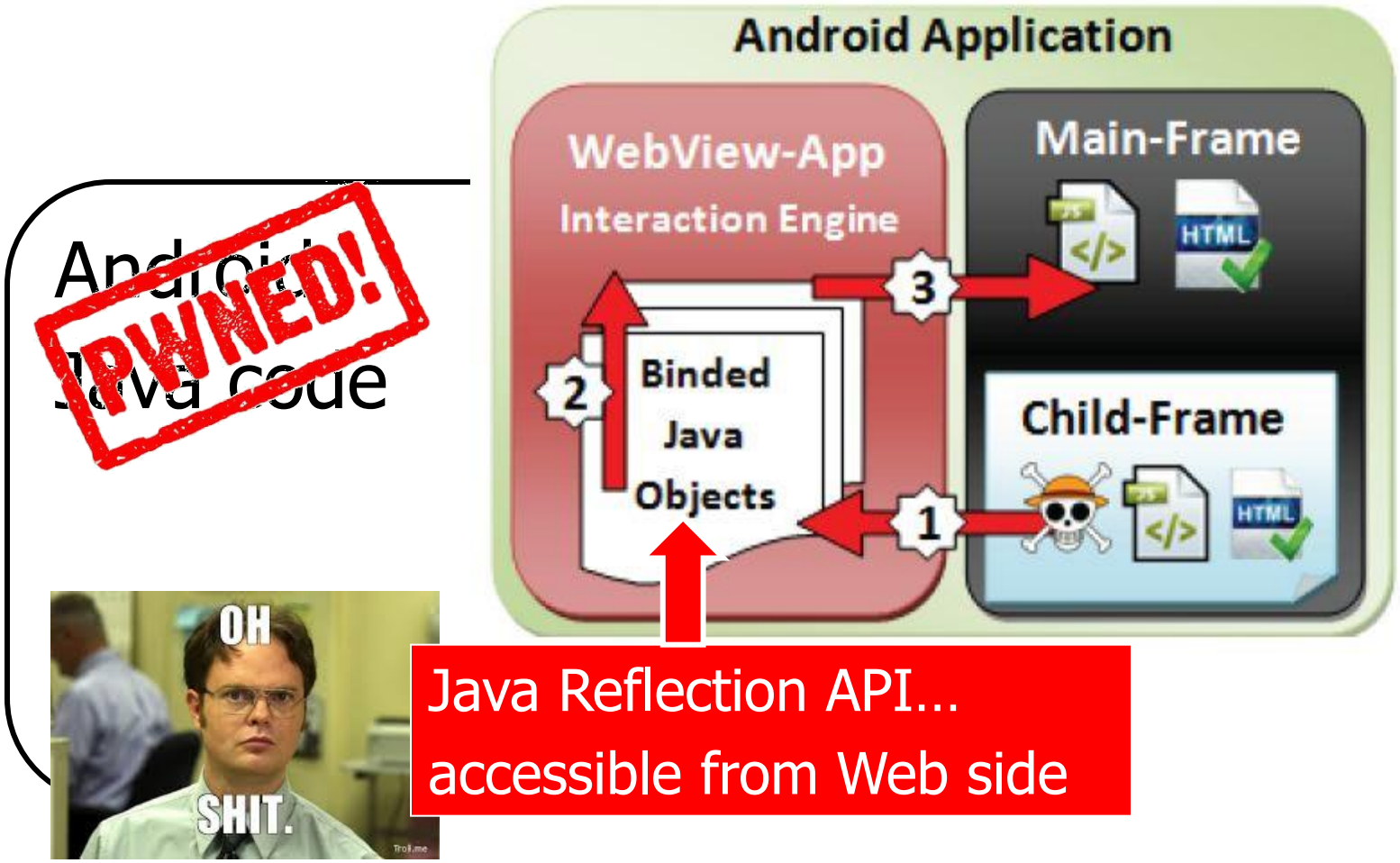
[Luo et al.]



What is the origin of this JavaScript object?

It Gets Worse

[Luo et al.]



Simple Fixes Don't Work

[Georgiev et al. "Breaking and Fixing Origin-Based Access Control in Hybrid Web/Mobile Application Frameworks". NDSS 2014]

- Most hybrid frameworks don't even attempt to verify whether access request comes from an authorized Web origin
- PhoneGap attempts to filter based on developer-provided whitelist
  - Mediation either incomplete (does not catch iframe loads) or too strict (prohibits even loading of content from other origins, breaks look-and-feel)
  - Incorrect origin checks
    - Broken regexes bite again – anchoring bugs, etc.

Showing this content is Ok, only native access should be blocked

State of the Union

- Convergence of Web and mobile programming
- Complex, poorly understood software stacks with badly fitting security policies
- New classes of vulnerabilities
  - Worst case: Web advertiser gets to inject arbitrary code into mobile apps running on your phone!%#$!
- Evolving defenses
  - Our capability-based NoFrak defense is being integrated into PhoneGap, but that's just the first step…