Mobility meets Web

Al Johri & David Elutilo

Origin-Based Access Control in Hybrid Application Frameworks

Outline

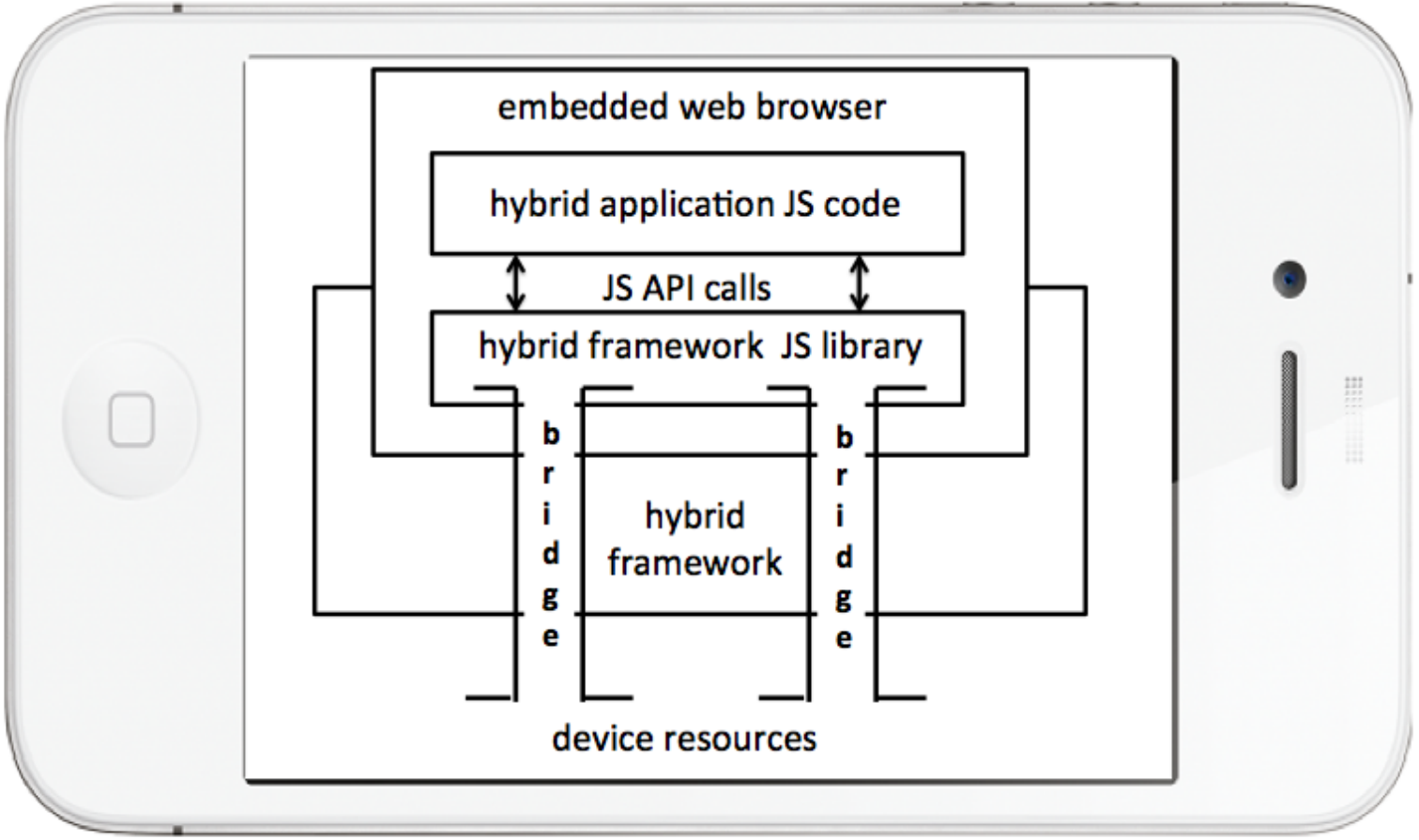
1. Introduction
   - Hybrid Apps & Frameworks
2. Security Models
3. Bridges
4. Fracking
5. Existing Defenses
6. NoFRAK
7. Related Work

Hybrid App Frameworks

http://html5please.com/

What are hybrid apps?

- Combine features of Web and “native” apps
- Portable & platform-independent
  - implemented in languages like HTML & Javascript
- Access to local device resources
  - Storage
  - Camera
  - Geolocation
  - Contacts


embedded web browser
hybrid application JS code
JS API calls
hybrid framework JS library
bridge
hybrid framework
bridge
device resources

PhoneGap

- Used by 400,000 devs
- Latest versions allow whitelists
- Acquired by Adobe in 2011

MoSync

MoSync

- Implements PhoneGap’s Javascript API
- Does not provide any whitelist capabilites

Web Marmalade

- Android and iOS
- Used by 50,000+ devs
- No whitelist attribute

appMobi.

- Free framework acquired by Intel
- Implements PhoneGap Javascript API
- whitelist provided through Javascript call

Blackberry WebWorks

- Provides same functions as other frameworks strictly for Blackberry platforms
- Fine-grained, domain-specific access control

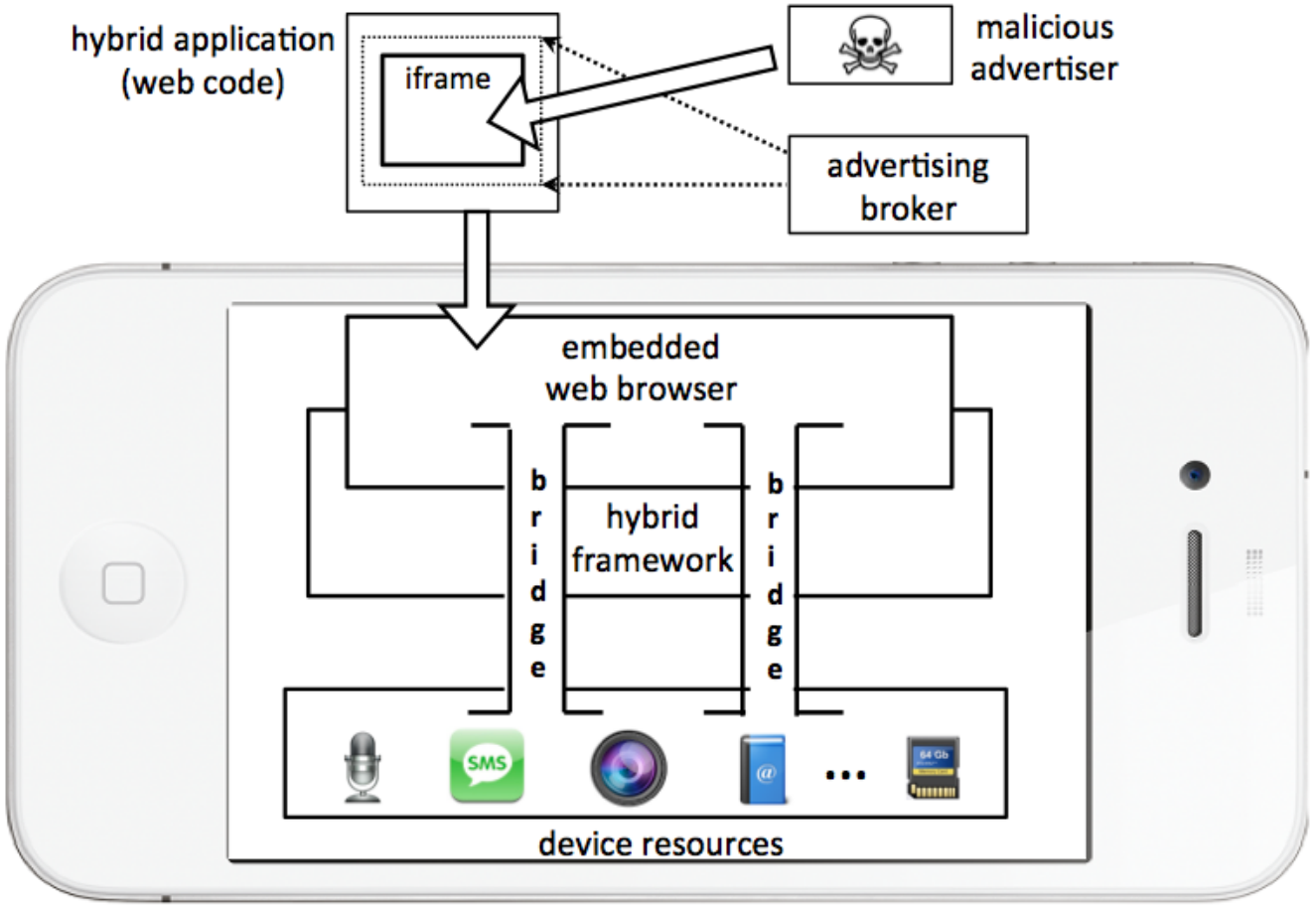


Fig. 1: Hybrid software stack

Security Models

- **Web Security**
  - Same Origin Policy (SOP)
    - defined by protocol, domain & port
  - SOP is enforced by all embedded browsers
  - iFrame Ad example

Security Models

- Local Security
  - OS takes care of security
  - Android & Windows use static permissions
  - Blackberry allows users to allow subset of requested permissions
  - iOS uses dynamic run-time access control
    - fewer permissions than than Android

Security Models

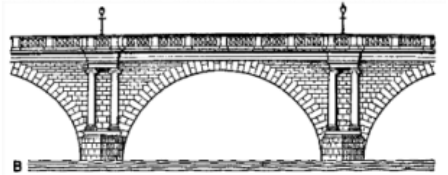
- Hybrid Security
  - Need to make sure that untrusted web content can’t access local resources
  - **NoBridge**
  - **NoLoad**
    - Conflicts with free-app business model

Bridges

addJavascriptInterface

prompt / alert / confirm

loadUrl / executeScript / InvokeScript



cookies

Home-made

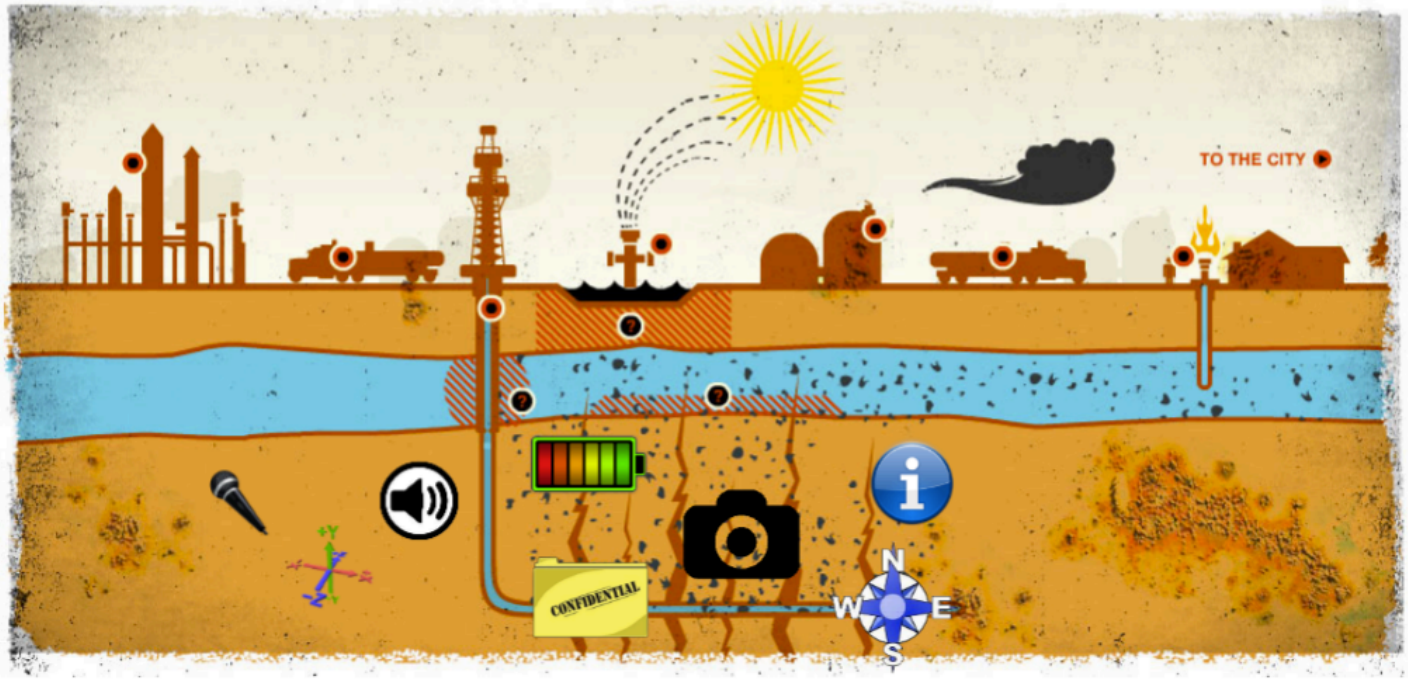
Web-to-local bridges

- Used to access local device resources
- Local code exposed to Javascript running in embedded browsers
  - Android - ‘addJavascriptInterface’
  - MacOS - ‘windowScriptObject’
- Insecure to malicious Javascript
- Local code can also override event handlers in browsers

Local-to-web bridges

- Used by local half to return data to Javascript libary on web half
- Local code triggers events notifying Javascript that data is ready to be retrieved
- “Frame Confusion”

Fracking



What is Fracking?

“Any attack that allows malicious foreign-origin Javascript to ‘drill’ through the defense layers and gain unauthorized access to device resources.”

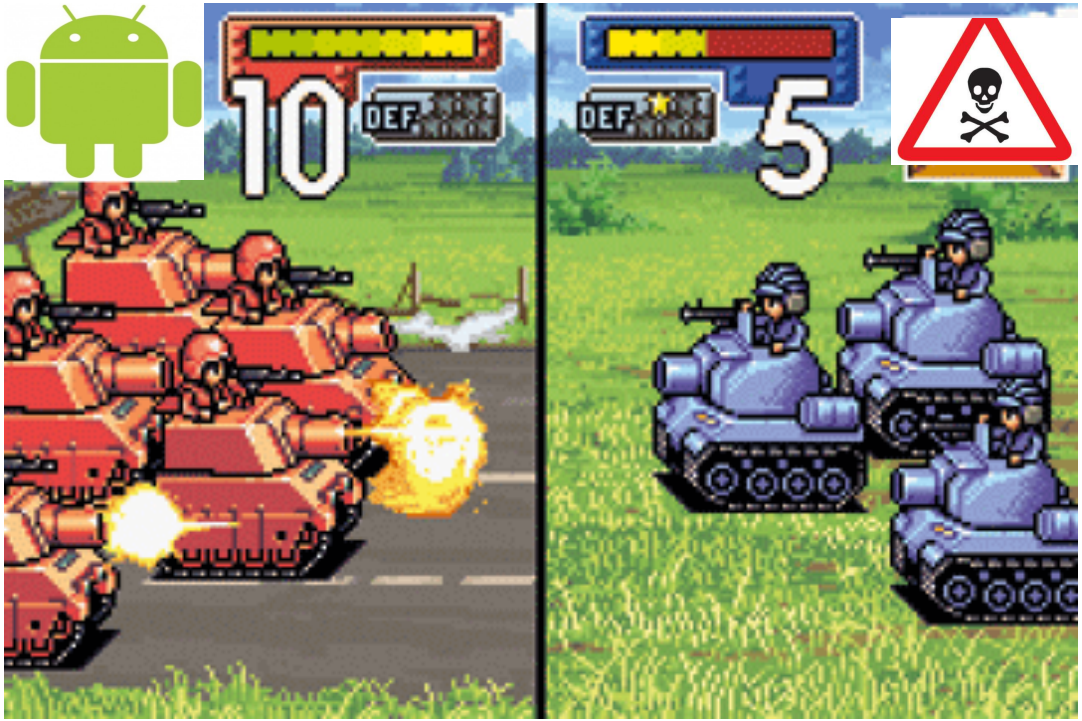
Fracking

- Hybrid frameworks rely on 4 defense layers:
  - SOP
  - Bridges
  - Origin checks
  - Access Control

Fracking - Types of Attacks

- Chosen-bridge attacks
- Interface-bridge exploits
- Event-bridge exploits
- URL-bridge exploits

Existing Defenses



NoLoad

- Any data from non-whitelisted origins is not loaded
- 2 error types:
  - incorrect URL interception
  - incorrect URL matching

NoLoad - Android

- Before v2.6, PhoneGap had no defense against unauthorized content
- Flawed implementation after
  - stricter than SOP
- URL matching exploit
  - Regex only looked at beginning of string
- HTTPS/HTTP ignored
- iOS shared the same vulnerabilites

NoLoad

- Any data from non-whitelisted origins is not loaded
- 2 error types:
  - incorrect URL interception
  - incorrect URL matching
- Doesn’t account for content only known at runtime
- Incompatible with free apps
  - Get money or get fracked

NoBridge

- Foreign content can load but only whitelisted domains can access the bridge.
- Relies on local half to correctly determine origin
  - unsupported by many embedded browsers
- Our NoFRAK attempts to solve this issue

NoFRAK

NoFRAK

- Enforces *NoBridge* property
- Key idea is web access to bridges need to authentication by unforgeable capability tokens
-

NoFRAK - Design

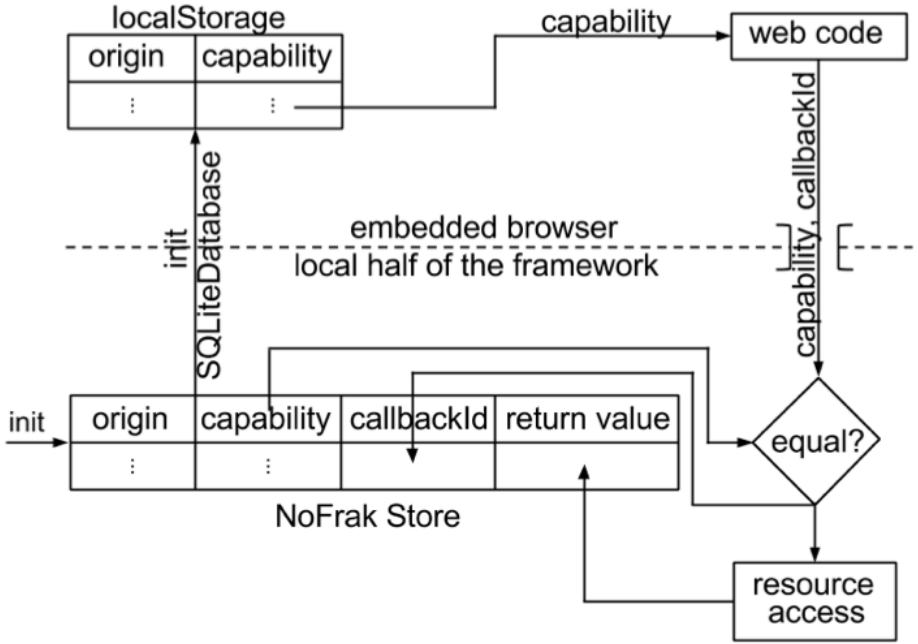


Fig. 12: NoFrak: Invoking a bridge

NoFrak - Evaluation

Device Resource	# of public methods
Accelerometer	3
Camera	3
Capture	4
Compass	3
Contacts	5
File	36
Geolocation	3
Globalization	12
InAppBrowser	6
Media	10
Notification	5
Splashscreen	2
Storage	8

TABLE I: Number of public methods for accessing different device resources in PhoneGap

	PhoneGap	NoFrak	Overhead
Sync	1.7713 ms	1.7755 ms	1.0024x
Async	0.1244 ms	0.1317 ms	1.0586x

TABLE II: Performance overhead of NoFrak

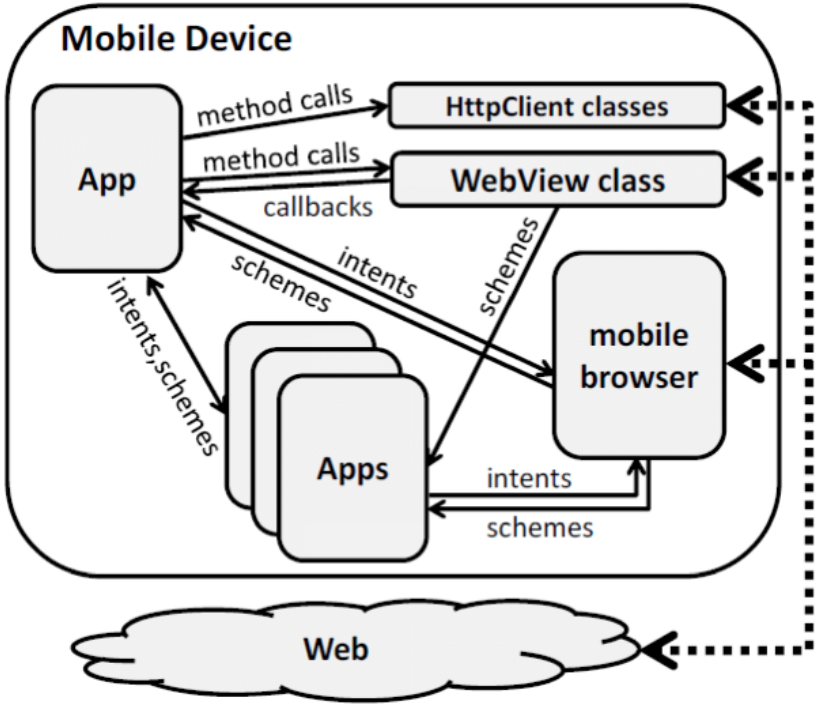
Unauthorized Origin Crossing on Mobile Platforms

The Problem

- Unlike browsers, mobile OSes don’t have mechanism to control cross-origin communications
  - app to app
  - app to web
- Leaves sensitive data open to maleficence

Mobile Channels

- Intent
  - Android-specific
  - app to app
- Scheme
  - e.g.***youtube://play?id=…***
  - app to app, web to app
- Web-accessing utility classes
  - WebView, HttpClient, etc.

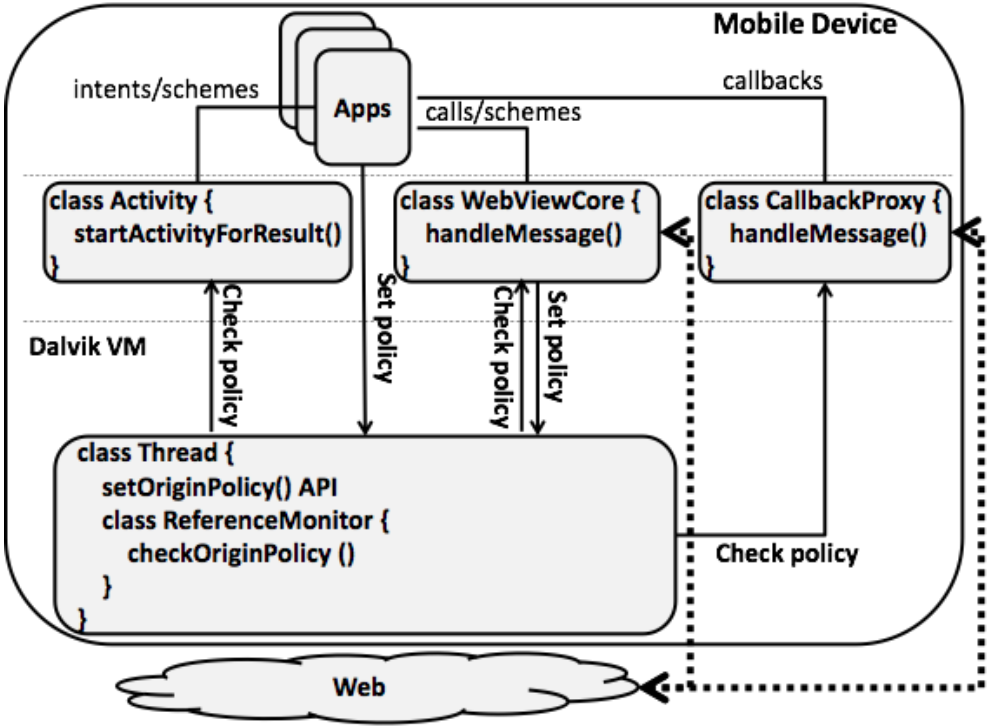


Vulnerability	Channel	Platform	Affected apps	Key points
Next-intent	Intent	Android	Dropbox app, Facebook app	An app calls startActivity(intent), its private Activities can be invoked, no matter which origin the intent is from.
fbconnect	scheme	Android	Facebook app and apps using Facebook SDK	A webpage, once redirects to a scheme URL, has no control on which app can receive it
website-invoke-app	scheme	Android & iOS	PlainText, TopNotes, Nocs, Yelp, TripAdvisor, etc.	The developer cannot know the true origin of the scheme message when receiving it.
callback	WebView	Android & iOS	pInterest app	Webpages from **arbitrary origins** can invoke code of the victim app through callback
Header-attaching	HttpClient	Android & iOS	Dropbox app	Headers will be attached to URLs from **arbitrary origins** through HTTPClient

Solution?

- **M**obile **Or**igin-**B**ased **S**ecurity (**Morbs**)
  - labels messages with origin information
  - exposes origin info to developers
  - mediates message passing based on origins

Morbs - Implementation



Morbs - Evaluation

Vulnerability	Fix without Morbs	Fix with Morbs
Next-intent	**Architecture change** of Dropbox and Facebook app	No modification
fbconnect	**Deprecation of the core feature** (affecting all apps with Facebook SDKs, which took several months)	setOriginPolicy("app://com.facebook.katana")
Website-invoke-app	Change **both Dropbox app and SDKs**	setOriginPolicy("app://com.getdropbox.dropbox")
callback	Unknown	setOriginPolicy("https://*.facebook.com")
Header-attaching	Add code to Dropbox app to check whether a URL is from dropbox.com before attaching authorization header	setOriginPolicy("http://*.dropbox.com")

Morbs - Evaluation

Channel	Type of Communication	Communication Delay (ms)	Impact of Morbs
intent	in-app	42.142	0.52%
	cross-app	46.267	0.47%
scheme	app-app	64.077	0.34%
	web-app	115.301	0.19%
utility classes	HttpClient	225.035	0.10%
	WebView	692.955	0.03%

Thank You.
Questions?