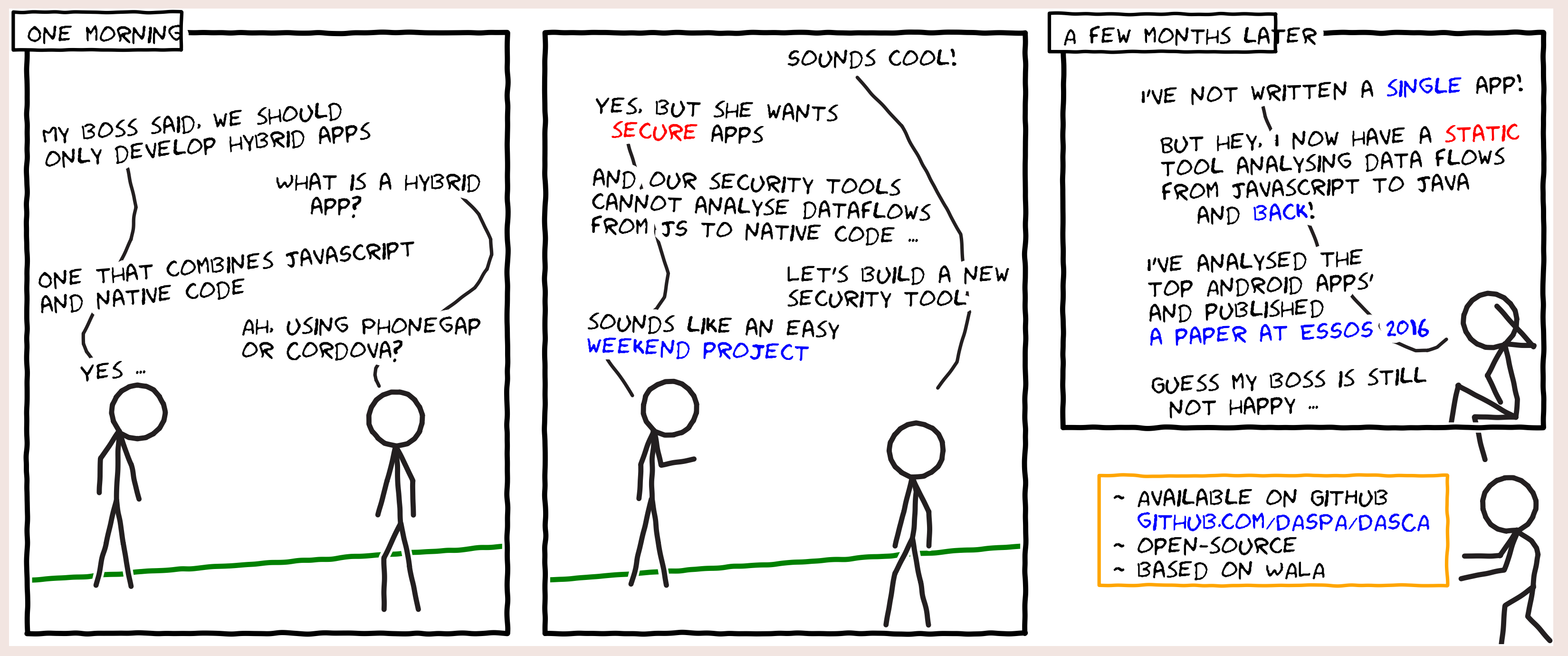
On the Static Analysis of Hybrid Mobile Apps

Achim D. Brucker and Michael Herzberg

{a.brucker,msherzberg1}@sheffield.ac.uk

Department of Computer Science, The University of Sheffield, Sheffield, UK

The Problem


ONE MORNING
MY BOSS SAID, WE SHOULD ONLY DEVELOP HYBRID APPS
WHAT IS A HYBRID APP?
ONE THAT COMBINES JAVASCRIPT AND NATIVE CODE
AH, USING PHONEGAP OR CORDOVA?
YES ...
SOUNDS COOL!
YES, BUT SHE WANTS SECURE APPS
AND, OUR SECURITY TOOLS CANNOT ANALYSE DATAFLOWS FROM JS TO NATIVE CODE ...
LET'S BUILD A NEW SECURITY TOOL!
SOUNDS LIKE AN EASY WEEKEND PROJECT
A FEW MONTHS LATER
I'VE NOT WRITTEN A SINGLE APP!
BUT HEY, I NOW HAVE A STATIC TOOL ANALYSING DATA FLOWS FROM JAVASCRIPT TO JAVA AND BACK!
I'VE ANALYSED THE TOP ANDROID APPS' AND PUBLISHED A PAPER AT ESSOS'2016
GUESS MY BOSS IS STILL NOT HAPPY ...
~ AVAILABLE ON GITHUB GITHUB.COM/DASPA/DASCA
~ OPEN-SOURCE
~ BASED ON WALA

What is a Hybrid App?

Native apps	**Hybrid apps**	**Web apps**
Java \ Swift \ C#	HTML5, JS, native	HTML5, JS

Native apps — Java \ Swift \ C#

- Developed for a specific platform
- All features available

Hybrid apps — HTML5, JS, native

- Build once, run everywhere
- Access to device features through plugins

Web apps — HTML5, JS

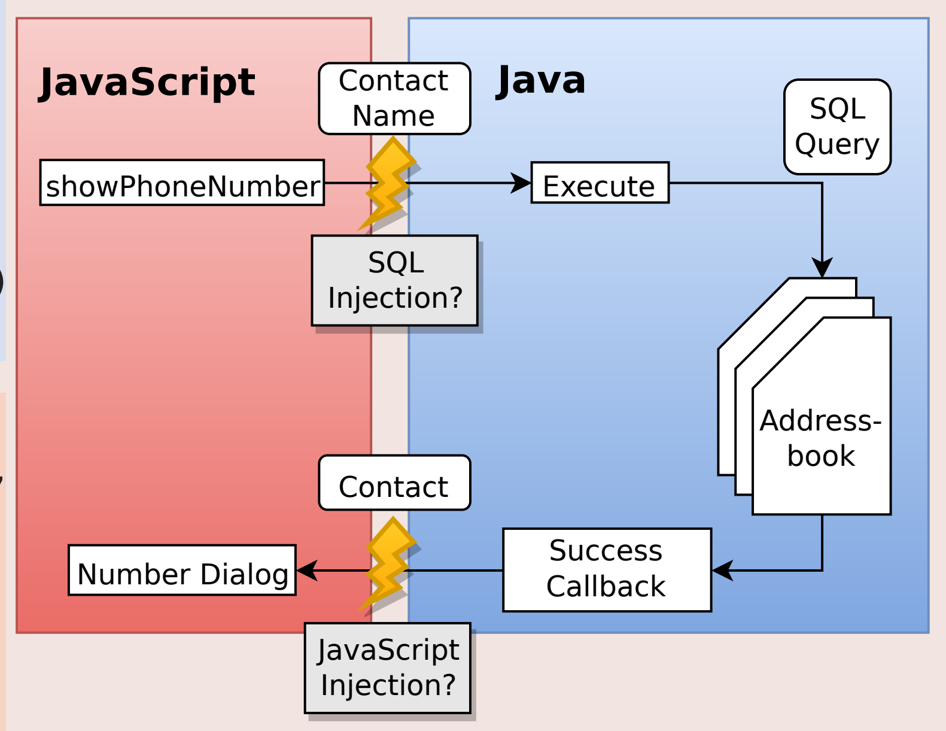
- Hosted on server, all platforms
- No access to device features

Platform-specific — Platform-independent

Dangerous Data Flows Across Borders

```
function showPhoneNumber(name) {
  var successCallback = function(contact) {
    alert("Phone␣number:␣" + contacts.phone);
  }
  var failureCallback = ...
  cordova.exec(successCallback, failureCallback,
      "ContactsPlugin", "find", [{"name" : name}])
}
```

```
class ContactsPlugin extends CordovaPlugin {
  boolean execute(String action, CordovaArgs args,
      CallbackContext callbackContext) {
    if ("find".equals(action)) {
      String name = args.get(0).name;
      find(name, callbackContext);
    } else if ("create".equals(action)) ...
  }
  void find(String name, CallbackContext callbackContext){
    Contact contact = query("SELECT␣...␣where␣name=" + name);
    callbackContext.success(contact);
  }
}
```



The Solution

Deep framework analysis

- Closest to the actual app
- Analyses the whole application
- **But**: Analysing the whole framework is very expensive

Modelling framework

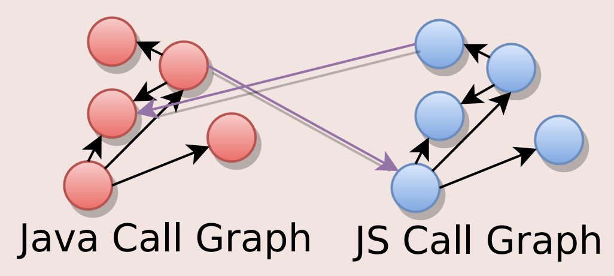
- Combines advantages
- Models Cordova framework
- Analyses (custom) Plugins, UI, and business logic

Modelling plugins

- Models framework and plugins
- Analyses only UI and business logic part
- **But**: No support for custom plugins

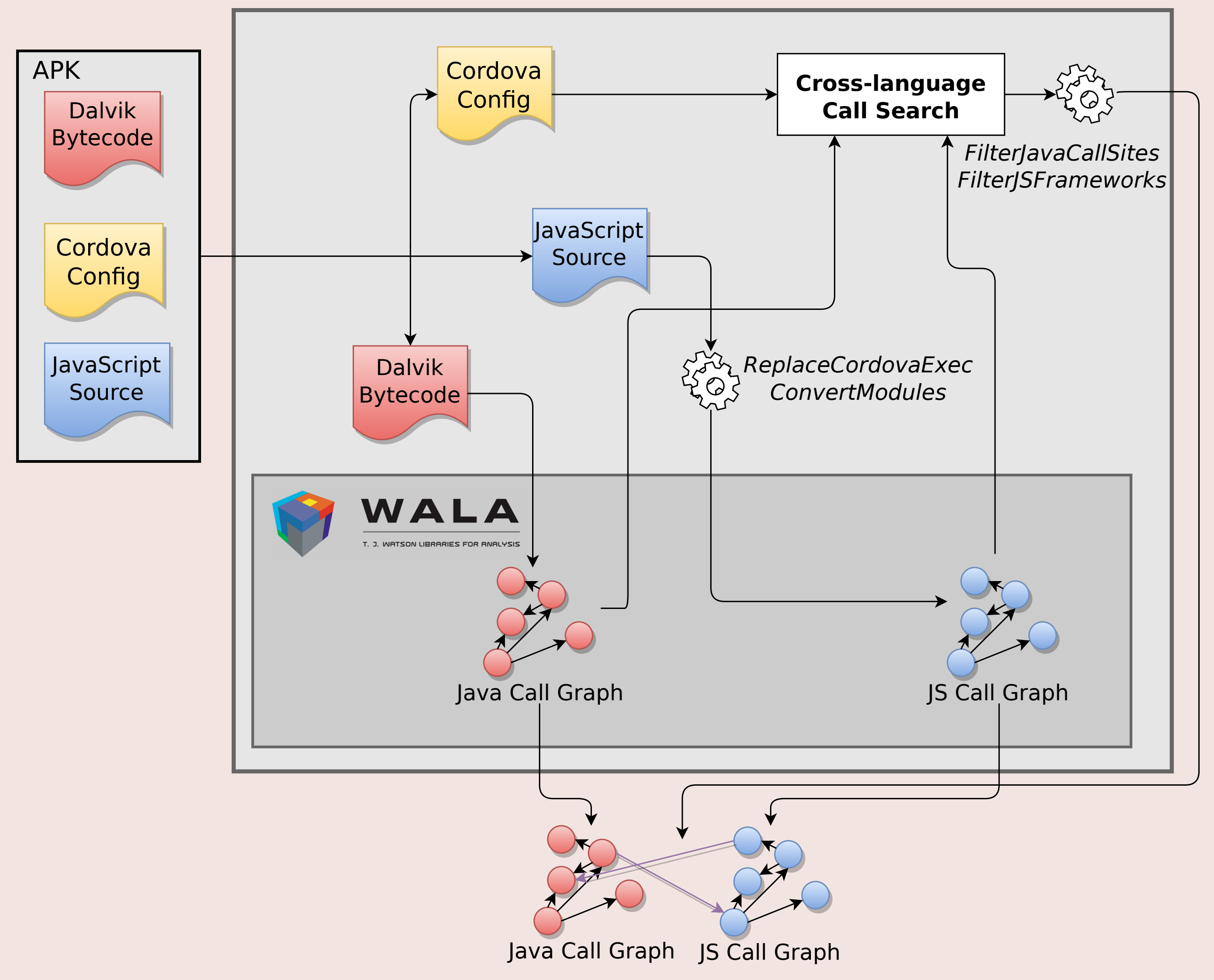
Our approach: Modelling framework

Core Idea



Call graphs are a fundamental datastructure and commonly used for static analysis purposes. We therefore developed an approach that combines the call graphs for the individual languages of the hybrid app into a **unified call graph**, which represents the whole application and can be used for further analysis. In order to evaluate our approach, we implemented a prototype of a tool for the Android versions of Apache Cordova apps on top of WALA.

Tool Architecture



The idea is to first build call graphs of Java and JavaScript separatly and then connect them using four heuristics that exploit frequent coding patterns: *ConvertModules*, *ReplaceCordovaExec*, *FilterJavaCallSites*, and *FilterJSFrameworks*. The first two preprocess the JavaScript source code before building the call graph, and the last two leverage static code analysis to improve the resulting unified call graph.

Results

- Recall: $\dfrac{\textit{Correctly reported calls}}{\textit{All reported calls}}$
- Precision: $\dfrac{\textit{Correctly reported calls}}{\textit{Calls actually present}}$

App	kLoC	kNodes	Plugins	Recall	Precision	Calls
app_{01}	43	9	5	33%	75%	17
app_{02}	27	8	4	100%	66%	13
app_{03}	106	18	8	1%	93%	61
app_{04}	53	14	3	100%	100%	7
app_{05}	64	10	7	33%	66%	29
app_{06}	53	8	12	35%	97%	316
sap_{01}	52	19	6	100%	66%	15
dvhma	17	7	4	100%	100%	15

Bigger Test Set Without Recall and Precision

Cross-language calls:

- calls from Java to JS: very common
- calls from JS to Java: surprisingly uncommon

App	Category	Java2JS	JS2Java	JS [kLoC]	Java [kLoC]
sap_{01}	Finance	2	12	35.5	17.0
sap_{02}	Business	20814	39	345.3	53.5
sap_{03}	Business	9531	75	572.3	135.8
app_{01}	Finance	9	13	26.3	17.8
app_{02}	Finance	2	10	11.2	16.8
app_{03}	Social	2349	31	4.6	103.7
app_{04}	Business	1	6	37.5	16.8
app_{05}	Finance	6	26	20.0	44.8
app_{06}	Finance	693	70	30.4	24.3
app_{07}	Travel & Local	3430	43	129.0	304.0
app_{08}	Entertainment	14220	67	36.7	23.0
app_{09}	Lifestyle	51553	89	36.3	44.7
app_{10}	Finance	8	36	43.7	18.4
app_{11}	Business	0	0	14.0	438.9
⋮	⋮	⋮	⋮	⋮	⋮
app_{48}	Business	22	26	9.6	106.1
app_{49}	Social	1300	66	62.4	92.1
app_{50}	Social	97338	109	192.6	391.0

Plugin Usage

Plugins are used for

- accessing device information
- showing native dialog boxes and splash screens
- accessing network information
- accessing the file storage
- accessing the camera
- . . .

Many different versions and some modified!

Plugin	
device	52%
inappbrowser	50%
dialogs	40%
splashscreen	36%
network-information	28%
file	28%
console	24%
camera	22%
statusbar	22%
PushPlugin	22%

Publications

A. D. Brucker and M. Herzberg. On the static analysis of hybrid mobile apps: A report on the state of apache cordova nation. In J. Caballero and E. Bodden, editors, *International Symposium on Engineering Secure Software and Systems (ESSoS)*, Lecture Notes in Computer Science, pages 72–88. Springer-Verlag, 2016. doi: 10.1007/978-3-319-30806-7_5.