Spartan Jester

End-to-end information flow control for hybrid Android applications

Julian Sexton
MITRE

Andrey Chudnov
Galois, Inc

David Naumann
Stevens Institute of Technology

Copyright © 2017 Galois, Inc

Partially supported by NSF award CNS-1228930
Example: personal assistant app

- Android permissions: INTERNET, RECORD_AUDIO, FILE_SYSTEM.
- Not enough to stop undesirable flows while allowing intended ones
Information flow control

$X$ flows to $Y \equiv$ variation in $X$ causes variation $Y$

### Explicit

$Y = X$;

### Implicit

$Y = 0$;

```java
if (X) {
    Y = 1;
}
```

- **Common abstraction**: labels on sources of data, propagated to memory locations.
- **Could be implemented as a type system, and types as sets of labels.**
- **$X$ having a label $\{A,B\}$ means it's value could have been influenced by sources with labels $\{A\}$ and $\{B\}$**

### Static

- Types/labels are propagated and checked at compile time
- Best fit for statically typed languages, like Java

### Dynamic

- Memory locations are tagged with types/labels at run-time, and updated during execution.
- Best fit for dynamically-typed languages, like JavaScript
Static Information Flow Control for Android

- TaintDroid [Enck, TOCS 2014]. Dynamic taint tracking in Dalvik VM.


- WALA [Tripp, FASE 2013]. Multi-language static analysis framework including taint analysis. Includes good Android support.

- SPARTA [Ernst, CCS’14]. Static data flow tracking, policy-oriented taint tracking. Fits neatly with Java annotations.

- Joana [Hammer, IJIS’09] [Mohr, CEUR’15]. PDG-based sound information flow analysis for Java. Ongoing work on Android support.
Hybrid mobile apps

Android App

Native component
Java

WebView: embedded browser

Web component
HTML + JavaScript

Phone APIs

72 F / 60 F      Cloudy
Hybrid mobile apps are portable

Android App
IOS App

Native component
Java
Objective-C

UIWebView: embedded browser
Web component
HTML + JavaScript

72 F / 60 F      Cloudy

Phone APIs
Hybrid mobile apps are portable

- Generic native component
  - PhoneGap
- Embedded browser
  - Web component
  - HTML + JavaScript

Phone APIs

Temperature: 72 F / 60 F
Weather: Cloudy
Hybrid mobile apps: IFC challenges

Native component
Java

WebView: embedded browser
Web component
HTML + JavaScript

?
Goal: strong end-to-end IFC for hybrid apps

Native component
Java

WebView: embedded browser
Web component
HTML + JavaScript
Previous work on IFC for hybrid apps

- Code injection attacks in HTML5 mobile apps
  - [Jin, CCS’14]
- Cordova/PhoneGap only
- Accurate modeling of APIs
  - Focused on detecting Cross-Site Scripting attacks
  - Taint analysis (no implicit flow)

Use Actarus [Guarnieri, ISSTA’11] for JavaScript analysis:
- Limitations in handling DOM, with and eval
## Previous work on IFC for JavaScript

<table>
<thead>
<tr>
<th></th>
<th>Label tracking</th>
<th>Secure Multi-Execution</th>
<th>Inlined</th>
<th>Browser modification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Compiler</td>
<td>Compiler</td>
</tr>
<tr>
<td>IFlowSigs</td>
<td></td>
<td></td>
<td>Compiler</td>
<td>Compiler</td>
</tr>
<tr>
<td>[Santos 2014]</td>
<td></td>
<td></td>
<td>[Chudnov 2015]</td>
<td>[Chudnov 2015]</td>
</tr>
<tr>
<td>JEST</td>
<td></td>
<td></td>
<td>[Hedin 2014]</td>
<td>[Hedin 2014]</td>
</tr>
<tr>
<td>JSFlow</td>
<td></td>
<td></td>
<td>[Bichawat 2014]</td>
<td>[Bichawat 2014]</td>
</tr>
<tr>
<td>WebKitIFC</td>
<td></td>
<td></td>
<td>[Rajani 2015]</td>
<td>[Rajani 2015]</td>
</tr>
<tr>
<td>ZaphodFacets</td>
<td></td>
<td></td>
<td>[Austin 2012]</td>
<td>[Austin 2012]</td>
</tr>
<tr>
<td>FlowFox</td>
<td></td>
<td></td>
<td>[DeGroef 2012]</td>
<td>[DeGroef 2012]</td>
</tr>
</tbody>
</table>

|                  |                |                         | Browser | Interpreter         |
| Compiler         |                |                         |         | Interpreter         |
|                  |                |                         |         | Browser             |
| ES5              |                |                         | ES5     | ES5                 |
| ES5\strict      |                |                         | ES5\strict | ES5\strict |
| Partial DOM Support |            |                         | Significant DOM support | Significant DOM support |
| No UCF           |                |                         | Coarse UCF | Full UCF |
| UCF = Unstructured Control Flow
## Previous work on IFC for JavaScript

<table>
<thead>
<tr>
<th>IFlowSigs</th>
<th>JEST</th>
<th>JSFlow</th>
<th>WebKitIFC</th>
<th>ZaphodFacets</th>
<th>FlowFox</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Santos 2014]</td>
<td>[Chudnov 2015]</td>
<td>[Hedin 2014]</td>
<td>[Bichawat 2014][Rajani 2015]</td>
<td>[Austin 2012]</td>
<td>[DeGroef 2012]</td>
</tr>
<tr>
<td>Compiler</td>
<td>Compiler</td>
<td>Interpreter</td>
<td>Browser</td>
<td>Interpreter</td>
<td>Browser</td>
</tr>
<tr>
<td>⇐ES3</td>
<td>ES5\strict</td>
<td>ES5\strict</td>
<td>ES5</td>
<td>⇐ES3</td>
<td>ES5</td>
</tr>
<tr>
<td>Partial DOM Support</td>
<td>Significant DOM support</td>
<td>Bare-bones DOM support</td>
<td>Significant DOM support</td>
<td>No DOM support</td>
<td>Black box DOM</td>
</tr>
<tr>
<td>No UCF</td>
<td>Full UCF</td>
<td>Coarse UCF</td>
<td>Full UCF</td>
<td>No UCF</td>
<td>N/A</td>
</tr>
</tbody>
</table>

UCF = Unstructured Control Flow
Previous work on IFC for JavaScript

<table>
<thead>
<tr>
<th>IFlowSigs</th>
<th>JEST</th>
<th>JSFlow</th>
<th>WebKitIFC</th>
<th>ZaphodFacets</th>
<th>FlowFox</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Santos 2014]</td>
<td>[Chudnov 2015]</td>
<td>[Hedin 2014]</td>
<td>[Bichawat 2014]</td>
<td>[Austin 2012]</td>
<td>[DeGroef 2012]</td>
</tr>
<tr>
<td>Compiler</td>
<td>Compiler</td>
<td>Interpreter</td>
<td>Browser</td>
<td>Interpreter</td>
<td>Browser</td>
</tr>
<tr>
<td>⊂ ES3</td>
<td>ES5\strict</td>
<td>ES5\strict</td>
<td>ES5</td>
<td>⊂ ES3</td>
<td>ES5</td>
</tr>
<tr>
<td>Partial DOM</td>
<td>Significant</td>
<td>Bare-bones</td>
<td>Significant</td>
<td>No DOM</td>
<td>Black box</td>
</tr>
<tr>
<td>Support</td>
<td>DOM support</td>
<td>DOM support</td>
<td>DOM support</td>
<td>support</td>
<td>DOM</td>
</tr>
<tr>
<td>No UCF</td>
<td>Full UCF</td>
<td>Coarse UCF</td>
<td>Full UCF</td>
<td>No UCF</td>
<td>N/A</td>
</tr>
</tbody>
</table>

UCF = Unstructured Control Flow
Bridging SPARTA and JEST

Host App
Analysed with SPARTA against an information-flow policy

WebView

Android object
Web API

Android facade
Web Facade

HTML+JS

Web Component
Automatically rewritten using JEST against an information-flow policy
@JavascriptInterface
public String schedule(String date)
{
    ...
}
Android WebView

```java
WebView wv = WebView(this);
int arg = 3;
String invoke = "f("+arg+")";
wv.evaluateJavaScript(invoke, callb);
```
Bridging SPARTA and JEST

```java
@JavascriptInterface
class ScheduleDate {
    public void scheduleDate(@Source("USER_INPUT") String date)
        @Source({"INTERNET","FILESYSTEM"})
        @Sink({"WRITE_CALENDAR"}) String date;
    {
        ...
    }
}

i = Android.schedule(dates[i]);
...
Bridging SPARTA and JEST

@JavascriptInterface
public void scheduleDate
    @Source(\"USER_INPUT\")  @Sink(\{})  String
    (@Source(\"INTERNET\",\"FILESYSTEM\")
    @Sink(\"WRITECALENDAR\")  String  date)
{
    ...
}

... 
i =
    Android.schedule(dates[i]);
    ...

WebView

SPARTA policy file ← → JEST policy file
Bridging SPARTA and JEST

@JavascriptInterface
class HostApp {
    public void scheduleDate
        @Source({"USER_INPUT"}) @Sink({}) String
            (@Source({"INTERNET","FILESYSTEM"})
             @Sink({"WRITE_CALENDAR"}) String date)
            {
                ...
            }
}

... i = Android.schedule(dates[i]);...

WebView

SPARTA policy file \rightarrow JEST policy file
Bridging SPARTA and JEST

@JavascriptInterface
public void scheduleDate
    @Source({"USER_INPUT"}) @Sink({}) String
    (@Source({"INTERNET","FILESYSTEM"})
    @Sink({"WRITE_CALENDAR"}) String date)
{
    ...
Bridging SPARTA and JEST

WebView

Host App

@JavascriptInterface
public void scheduleDate
   @Source("USER_INPUT") @Sink({}) String
   (@Source("INTERNET","FILESYSTEM")
      @Sink("WRITE_CALENDAR") String date)
{
   ... 
}

WebViewFacade
Encode as JSON

Android object

Web API

Web Facade

HTML+JS

SPARTA policy file → JEST policy file

JSON encoded value

Label ≤ {INTERNET,FILESYSTEM}

WebViewFacade
Encode as JSON

JavaScript:

```
... 
i = Android.schedule(dates[i]);
... 
```
Bridging SPARTA and JEST

```java
@JavascriptInterface
public void scheduleDate
    @Source({"USER_INPUT"}) @Sink({}) String
    @Source({"INTERNET","FILESYSTEM"}) @Sink({"WRITE_CALENDAR"}) String date
    {
        ...
    }
```

`WebViewFacade`
Encode as JSON

`WebView`
Case study: mobile org-mode

Java
- thin layer over Android API
- capture the voice input
- input the password

HTML + JavaScript
- most app logic is here
- parsing of org files
- displaying contents
- filtering of sensitive information
Analyst workflow

1) Express the policy for Java code in terms of a SPARTA policy file and SPARTA source annotations.

2) Derive the JEST policy: principals and channel labels based on the Java policy for functions marked as @JavascriptInterface.

3) Run SPARTA to check the Java code.

4) Run JEST on the embedded page
   - at build time, or
   - via a proxy server
Experience report

- IFC still non-trivial...
- SPARTA
  - Explicit flows only
  - Type inference is intraprocedural and requires annotating every function
  - The labeling of WebView interactions is coarse-grained. Labels it with INTERNET, but our policy prevents flows there.
- JEST
  - Multiple implicit flows due to exceptions in the web component
  - This is due to the use of string operations that are methods
  - All this causes run-time failures ("No-sensitive-upgrade")
  - We convert implicit flow to explicit and use upgrade annotations
Conclusions. Q&A.

- Investigated an approach to track information flow in hybrid Android apps

- First combination of static (Java) and dynamic (JavaScript) analysis for this goal

- This is a promising direction, but more work is required on improving the usability of tools and the policy specification.

- A single policy language that can talk about the IO channels of both native and web components is necessary to avoid duplication and policy mismatch.