Combining the Security Risks of Native and Web Development: Hybrid Apps

Achim D. Brucker and Michael Herzberg
{a.brucker, msherzberg1}@sheffield.ac.uk

Software Assurance & Security Research
Department of Computer Science, The University of Sheffield, Sheffield, UK
https://logicalhacking.com/

May 12, 2017

OWASP AppSec EU Belfast

8th to 12th of May 2017

Waterfront Conference Center

{* LogicalHacking *}.com

The University Of Sheffield.

Combining the Security Risks of Native and Web Development: Hybrid Apps

Abstract

Cross-platform frameworks, such as Apache Cordova, are becoming increasingly popular. They promote the development of hybrid apps that combine native, i.e., system specific, code and system independent code, e.g., HTML5/JavaScript. Combining native with platform independent code opens Pandora's box: all the the security risks for native development are multiplied with the security risk of web applications.
In the first half of our talk, we start our talk with short introduction into hybrid app development, present specific attacks followed by a report on how Android developers are using Apache Cordova. In the second half of the talk, we will focus on developing secure hybrid apps: both with hands-on guidelines for defensive programming as well as recommendations for hybrid app specific security testing strategies.

Outline

Outline

Hybrid mobile apps

Native apps
Java \ Swift \ C#

- Developed for a specific platform
- All features available

Web apps
HTML5 and JS

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

Platform-specific

Platform-independent

Hybrid mobile apps

Native apps
Java \ Swift \ C#

- Developed for a specific platform
- All features available

Hybrid apps
HTML5, JS, and native

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

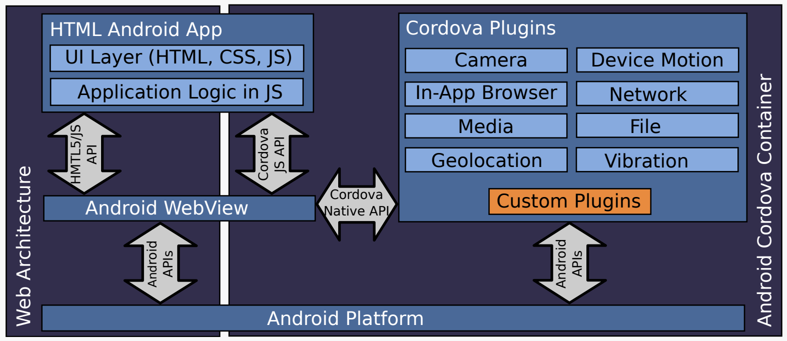
Web apps
HTML5 and JS

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

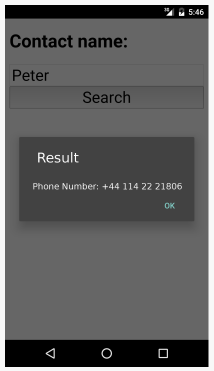
Platform-specific

Platform-independent

The architecture of Apache Cordova



Example app



Contact name:

Peter

Search

Result

Phone Number: +44 114 22 21806

OK

Example: get phone number

```
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);
    }
}
```

Example: get phone number

```
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);
    }
}
```

Example: get phone number

```
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);
    }
}
```

Example: get phone number

```
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);
    }
}
```

Example: get phone number

```
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);
    }
}
```

Example: get phone number

```
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);
    }
}
```

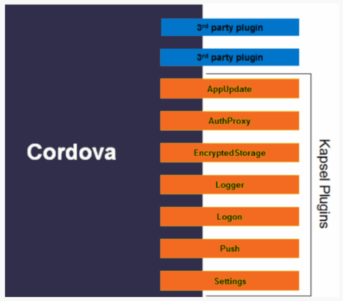
Example: get phone number

```
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);
    }
}
```

One framework, many names

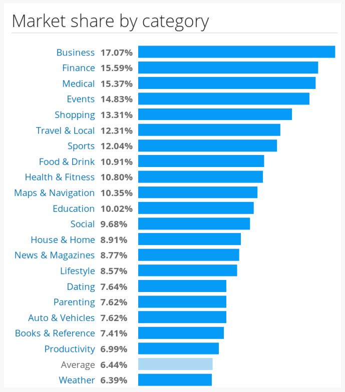
- Many frameworks extending Cordova
  - Adobe PhoneGap
  - SAP Kapsel
  - Onsen
  - ...
- These frameworks provide
  - additional plug-ins (access to native components)
  - additional HTML5/JavaScript libraries or interfaces



Outline

Cordova in the real world

- How many apps are using Cordova?
- How is Cordova used by app developers?
- Are cross-language calls common or not?



Source: https://www.appbrain.com/

What we have done

Selection of apps

- all apps that ship Cordova from Google's Top 1000:
  - 100 apps ship Cordova plugins
  - only 50 actually use Cordova (5%)
- three selected apps from SAP (using SAP Kapsel)

Development of a static analysis tool

- analysing Android apps (`*.apk` files)
- specialised in data-flows from Java to JavaScript and vice versa
- based on WALA
- in addition: list used plugins

Manual analysis of 8 apps (including one from SAP)

- to understand the use of Cordova
- to assess the quality of our automated analysis

What we have learned: plugin use

Plugins are used for

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

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

What we have learned: app size and cross-language calls

App size:

- mobile apps are not always small
- SAP apps seem to be larger than the average

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
⋮	⋮	⋮	⋮	⋮	⋮

What we have learned: exceptional behaviours

Cordova use:

- no HTML/JS in the app
- no use of Cordova

Plugin use:

- often callbacks are not used (missing error handling)
- plugins are modified
- plugins might use JNI

Outline

Why is it hard to the the security of hybrid apps

Web technologies (i.e., JavaScript)

- lack of typing, higher order functions, asynchronous programming models
- highly dynamic (e.g., eval(…), dynamic loading)
- …

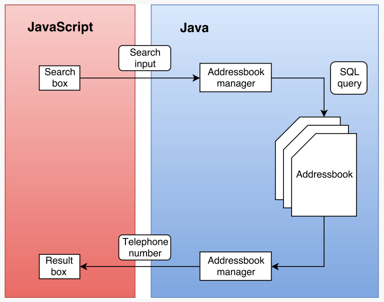
Large Libraries and Modules

- large ($\approx$ 100kLOC) third party (FOSS, proprietary) libraries
- both native (Java) and JavaScript
- complex core framework
- …

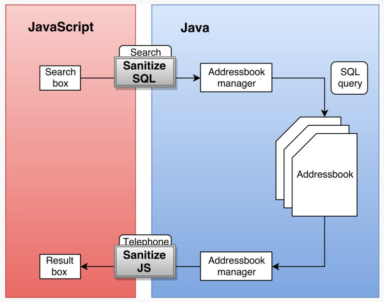
Cross-Language-Analysis

- many data-flows across language boundaries
- datatype conversion
- not only for accessing sensors (e,g, session plugin requires $>$ 10 language switches)
- …

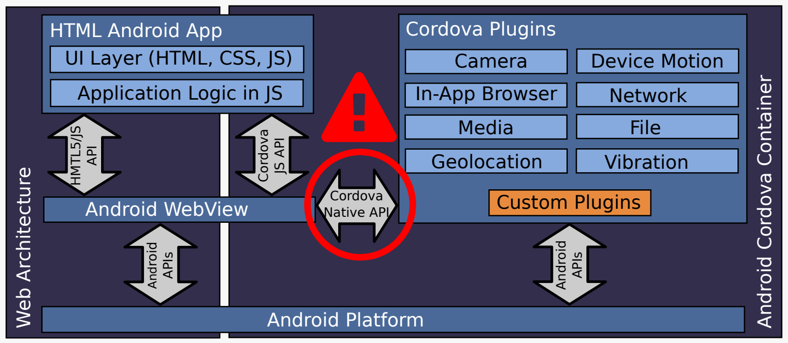
Example: Get Phone Number



Example: Get Phone Number



Weak spot: JS <-> Java bridge



Exploiting the JavaScript to Java bridge (CVE-2013-4710)

- We can expose Java methods in JavaScript

```
        foo.addJavascriptInterface(new FileUtils(), "FUtil");
```

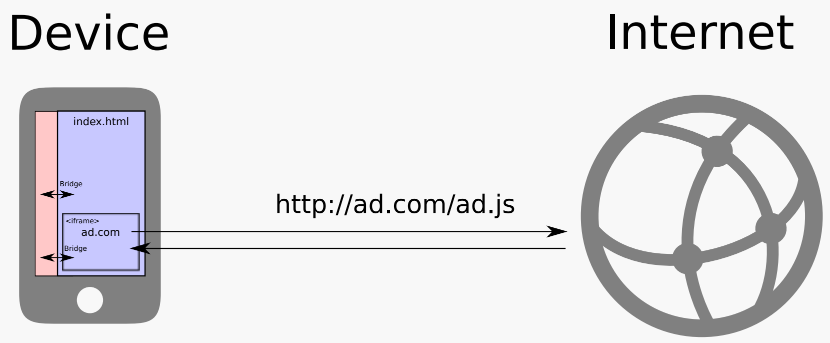
- And use them in JavaScript easily

```
<script type="text/javascript">// <![CDATA[
filename = '/data/data/com.livingsocial.www/' + id +'_cache.txt';
FUtil.write(filename, data, false);
// ]]></script>
```

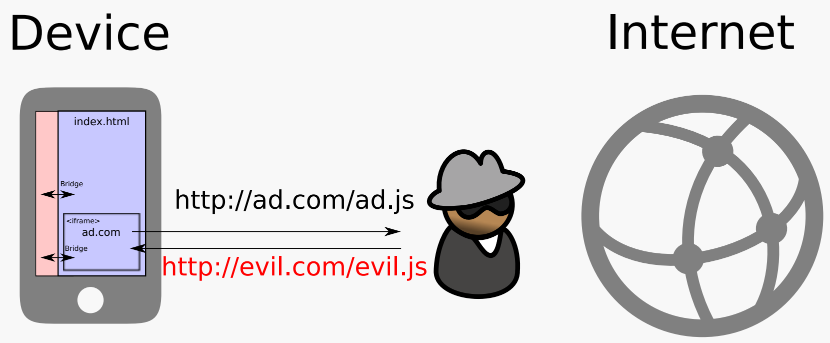
- Which might expose much more than expected

```
  function execute(cmd){
     return
     window._cordovaNative.getClass().forName('java.lang.Runtime').
                    getMethod('getRuntime',null).invoke(null,null).exec(cmd);
  }
```

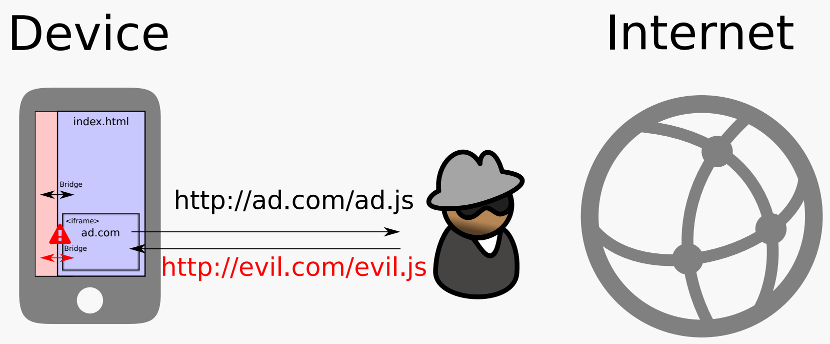
Never use http without SSL, or even iframes!



Never use http without SSL, or even iframes!



Never use http without SSL, or even iframes!



Outline

Recommendations: the (hopefully) obvious parts

Cordova apps are **Web applications**:

- do secure JavaScript programming
- content security policy, same origin policy
- …

Warning: the WebView sandbox is not as strong as on desktop Web browsers

Cordova apps are **native/Java apps**:

- do secure Java/Objective-C/…programming
- do not trust validations done in the JavaScript part of the plugin
- …

Recommendations: we should not forget

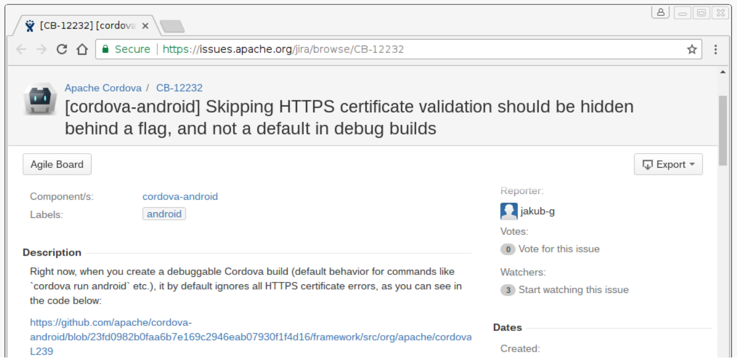
Cordova apps are **cordova applications**:

- plugin whitelisting
  https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-whitelist/
- read the Cordova security guide:
  https://cordova.apache.org/docs/en/latest/guide/appdev/security/

Cordova apps are **mobile applications**:

- permissions
- which OS versions do you need to support?
- check ssl certificates (debug mode!)
- …

Did you know


[CB-12232] [cordova

https://issues.apache.org/jira/browse/CB-12232

Apache Cordova / CB-12232

[cordova-android] Skipping HTTPS certificate validation should be hidden behind a flag, and not a default in debug builds

Component/s: cordova-android
Labels: android

Reporter: jakub-g
Votes: 0 Vote for this issue
Watchers: 3 Start watching this issue

Description

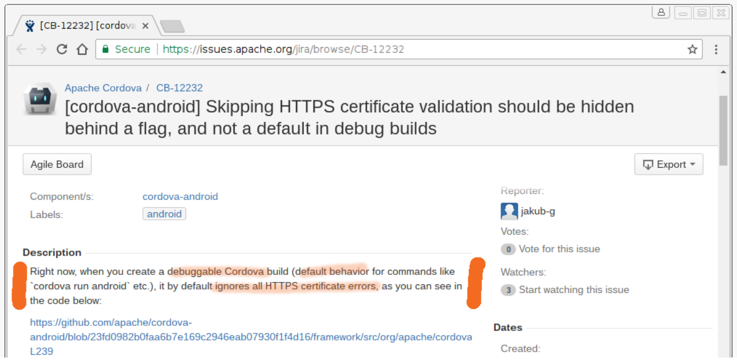
Right now, when you create a debuggable Cordova build (default behavior for commands like `cordova run android` etc.), it by default ignores all HTTPS certificate errors, as you can see in the code below:

https://github.com/apache/cordova-android/blob/23fd0982b0faa6b7e169c2946eab07930f1f4d16/framework/src/org/apache/cordova
L239

Dates

Created:

Did you know


[CB-12232] [cordova

https://issues.apache.org/jira/browse/CB-12232

Apache Cordova / CB-12232

[cordova-android] Skipping HTTPS certificate validation should be hidden behind a flag, and not a default in debug builds

Component/s: cordova-android
Labels: android

Reporter: jakub-g
Votes: 0 Vote for this issue
Watchers: 3 Start watching this issue

Description

Right now, when you create a debuggable Cordova build (default behavior for commands like `cordova run android` etc.), it by default ignores all HTTPS certificate errors, as you can see in the code below:

https://github.com/apache/cordova-android/blob/23fd0982b0faa6b7e169c2946eab07930f1f4d16/framework/src/org/apache/cordova
L239

Dates

Created:

Recommendation: use the latest framework version

Frameworks (and the underlying OS) can have vulnerabilities:

- use the latest version of Cordova
- monitor for public know vulnerabilities (e.g., CVEs)

Or, in other words: secure your software supply chain

Framework vulnerabilities can be severe:

- Java code execution via JavaScript: CVE-2013-4710
  Avoid Cordova on Android below 4.1 & use AddJavaScriptInterface annotation
- (incomplete) overview:
  https://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-27153/Apache-Cordova.html

Outline

If you are using static analysis: Considerations

Deep framework analysis

- Closest to the actual program
- But: Framework very expensive

Modelling the framework

- Models the Cordova framework
- Analyses plugins

Modelling the plugin (interfaces)

- Models both framework and plugins
- Analyses only UI and business logic part
- But: Developers can write own plugins

If you are using static analysis: Recommendations

In case you are mostly

- developing HTML5/JavaScript:
  - Use a SAST tool for JavaScript
  - Ensure that your plugin APIs are configured as sink/sources
- developing plugins
  - Use a SAST tool for Java
  - Ensure that your plugin APIs are configured as sink/sources
- developing full-featured Cordova apps
  - Method 1: scan Java and JavaScript “in isolation”
    - use a SAST tool for JavaScript (with configured sink/source for all plug-ins)
    - use a SAST tool for Java (with configured sink/source for all plug-ins)
  - Method 2: scan Java and JavaScript source code together
    - ensure that Cordova framework is modelled to allow analysis of cross-language data-flows

If you are using dynamic analysis (e.g., pen testing)

- check for attacks on the native part by an “web attacker” (e.g., SQL injection)
- check for plug-ins with removed JavaScript part (if you can inject JavaScript, you can use those plug-ins)
- and it is always good to ensure that debug-mode is disabled

Outline

Conclusion

- Hybrid mobile apps are getting more popular
  - they are recommended by enterprise vendors
  - they are used outside of the “traditional mobile devices” (e.g., web kiosk, smart TVs)
- Securing hybrid apps is a challenge and requires expertise in
  - Web application security
  - native/Java security
  - mobile security
  - Cordova security

Thank you for your attention!
Any questions or remarks?

Contact:

Dr. Achim D. Brucker and Michael Herzberg
Department of Computer Science
University of Sheffield
Regent Court
211 Portobello St.
Sheffield S1 4DP, UK

{a.brucker, msherzberg1}@sheffield.ac.uk
https://logicalhacking.com/blog/

Bibliography

Achim D. Brucker and Michael Herzberg.
On the static analysis of hybrid mobile apps: A report on the state of apache cordova nation.
In Juan Caballero and Eric Bodden, editors, *International Symposium on Engineering Secure Software and Systems (ESSoS)*, number 9639 in Lecture Notes in Computer Science, pages 72–88. Springer-Verlag, 2016.

Stanislav Dashevskyi, Achim D. Brucker, and Fabio Massacci.
On the security cost of using a free and open source component in a proprietary product.
In Juan Caballero and Eric Bodden, editors, *International Symposium on Engineering Secure Software and Systems (ESSoS)*, number 9639 in Lecture Notes in Computer Science, pages 190–206. Springer-Verlag, 2016.

Document Classification and License Information