Privacy and Smartphones
Jagdish Prasad Achara, Claude Castelluccia, James-Douglass Lefruit, Vincent Roca

▲ To cite this version:
| Jagdish Prasad Achara, Claude Castelluccia, James-Douglass Lefruit, Vincent Roca. Privacy and Smartphones. CAPPRIS project reunion, Sep 2013, Lyon, France. hal-00915771

HAL Id: hal-00915771
https://inria.hal.science/hal-00915771
Submitted on 9 Dec 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Mobilitics
Inria-CNIL project: privacy and smartphones

Privatics team (Jagdish Achara, Claude Castelluccia, James Lefruit, Vincent Roca) – Inria Rhone-Alpes

CAPPRIS Reunion, Lyon
Sept 10th, 2013
Outline

● Motivations

● “Private Data Leak Detection” methodology and results

● Conclusions
Motivations (1): The revolutionary arrival of AppStore model of App distribution

- A large number of actors present on the device
  - No more the presence of merely smartphone provider
  - Both first (App server itself) & third-party (trackers, A&A etc.)

Both first & third-party (very dangerous)
Motivations (2) : The arrival of App stores

- More opportunities for personal information leakage to various parties
  - Not only limited to web browsers as is the case in desktops/laptops
  - Apps for dedicated services (FB, LeMonde, SNCF etc.)
Motivations (2)

- Difficult to trust all these parties
  - various scandals in the past
    - For example, Twitter and Path uploading users all contacts to their servers [1] [2]
  - WSJ: What they know – Mobile [3]

[2] http://www.theregister.co.uk/2012/02/15/twitterStores_address_books/
Motivations (3)

● Smartphones are well suited to marketers/trackers
  ○ contain a lot of info on user **interests** and **behaviors**
    ○ much more than on desktop/laptop
    ○ because **various sensors** (GPS, Camera etc) and **comm technologies** (WiFi, GSM etc.) generate PI
    ○ because smartphones are at the **center of our cyber activities**, and **very personal** (it’s not shared usually)
    ○ because smartphones have almost **all-time Internet connectivity**
    ○ Because they’re **barely turned off**

→ leads to accurate and detailed user profiling
Motivations (4)

- A direct consequence is a large presence of online advertisers/trackers and many others...

→→→ This requires scrutinizing smartphones for privacy risks

“tracking the trackers”


**Mobilitics project and its goals**

- **started in January 2012**

- focuses on Android and iOS
  - the leading mobile OS

- **Goal**: investigate smartphone Apps and OS for potential privacy risks
Outline

• Motivations

• “Private Data Leak Detection” methodology and results

• Conclusions
General approach (iOS & Android)

1. Run Apps on instrumented versions of Android and iOS

2. Collect data in a local sqlite database

3. Analyze the data offline for potential private data leakage
iOS (1) : Some background

- Closed source and only code signed from Apple can be executed
  - enforced by secure boot chain

- Instrumenting iOS requires “Jailbreaking”
  - essentially a way to bypass Apple’s secure boot chain

- Also, no App source available → only binary rewriting is possible
iOS (2) : Some Background

- iOS Apps are written in
  - Objective-C, C, C++

- Private data can only be accessed by Apple defined frameworks written in Objective-C/C/C++

- Enforcement of user privacy by iOS
  1. Apple vetting process when Apps are submitted to AppStore
  2. Users are asked before iOS gives access to PI to an App
**iOS (3) : General Idea**

- **Idea:** change the implementation of the APIs responsible for private data access in order to:
  - detect the App accessing the private data
  - collect the data so that it can be searched later if it’s transmitted to the network

![Diagram showing the flow of data between Apps, iOS Frameworks, and a SQLite database](diagram.png)
iOS (4) : But how to do it?

- As source code is not available, binary patching?
  - It’s a nightmare, I think!

- Dynamically, at runtime?
  - Fortunately, yes!
    - Use Objective-C runtime method “method_setImplementation”
    - Replace the C/C++ functions at assembly level.

*NB: we use a third-party framework (MobileSubstrate) which makes it lot simpler… [http://iphonedevwiki.net/index.php/MobileSubstrate](http://iphonedevwiki.net/index.php/MobileSubstrate)
iOS (5) : But how to do it?

- Whole code (modified implementation of the methods) is compiled in a dylib
  - and loaded at launch time in a process of interest
iOS (6) : But how to do it?

- We capture relevant info (method args, return values) and store it in a local sqlite DB

- In order to confirm privacy leaks
  - We also need to dump whole network data
    - follows the same technique (method/function interception)
    - done at BSD Socket level to ensure no App can bypass it
iOS (7) : Post Analysis of data collected

1. Identify private data accessed by Apps

2. Search for private data in the network traffic to see if it’s sent, and where

3. Search for private data in the input to cryptographic / hash functions, and if there’s some, search the output in the network traffic

4. Find out if Apps use cross-App tracking techniques by using the “UIPasteBoard” class
iOS (8) : Limitations

- Are private data manipulations (hash, encryption etc.) done with custom functions...
  - ...rather than using standard iOS API?
  - if yes, we cannot detect it as we don’t know what to search in the network traffic 😞
  - For example, a simple XOR with a static key is sufficient

→ a fundamental limitation of our approach
  - hard to evaluate if this is current practice or not
  - But this means...results obtained using our technique would be lower-bound
iOS (9) : Tests and results

- We chose 78 representative free iOS Apps
  - Goal is to be representative of the main App categories
  - Same set of Apps would be tested on Android (chosen Apps are available on both platforms): to have a behavioral comparison

<table>
<thead>
<tr>
<th>Private Information (PI)</th>
<th>Total No. of Apps accessing PI</th>
<th>Total No. of Apps sending unmodified PI</th>
<th>Total No. of Apps modifying PI</th>
<th>Total No. of Apps sending modified PI</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDID</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Accounts</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AdIdentifier</td>
<td>36</td>
<td>26</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>Location</td>
<td>12</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>IdentifierForVendor</td>
<td>25</td>
<td>3</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>DeviceName</td>
<td>11</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AddressBook</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ProcessNames</td>
<td>NA</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Carrier Network</td>
<td>NA</td>
<td>6</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>WiFiMACAddress</td>
<td>NA</td>
<td>6</td>
<td>47</td>
<td>17</td>
</tr>
<tr>
<td>BluetoothMACAddress</td>
<td>NA</td>
<td>0</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>SerialNumber</td>
<td>NA</td>
<td>0</td>
<td>17</td>
<td>0</td>
</tr>
</tbody>
</table>
iOS (10) : Trackers disguise device tracking

- 59% Apps bypass the official iOS6 “AdvertisingID”
  - should not be the case
  - the AdID is supposed to let the end-user control tracking by resetting it as desired…
  - … it’s just an illusion 😞

- 37% Apps will still bypass the AdID with future iOS7 that bans the access to MAC address
  - this % will increase as more companies will shift to other types of permanent identifiers for tracking
Android (1) : Overview of methodology

- We benefit from the open-source nature of Android for instrumentation
  - Change the Android source code itself

- Same technics as with iOS:
  - Add all events captured in a local sqlite database
  - Dump the network data at BSD Socket level
  - Dump the encryption/hash data
  - Perform post-analysis

- Still in progress…
Outline

● Motivations

● “Private Data Leak Detection” methodology and results

● Conclusions
Conclusions and remarks

• Trackers disguise device tracking
  ❍ 59% Apps are employing techniques they are not supposed to, in order to track users
  ❍ makes iOS6 “AdvertisingID” almost useless
  ❍ little progress in future iOS7
  ❍ Apple can’t ignore this trend

• Private data is sent to various parties
  ❍ As shown in the Table before

• Live experiment to be conducted at CNIL with various users
Improvements to make (version 2)

● We still need to distinguish between first and third-party (would require manual interception to some extent?)

● Increase the number of Apps being tested (with paid Apps too this time to verify if some difference exist wrt. privacy)

● Some known glitches to be fixed (access to serial number, bluetooth MAC Address etc.)
Last but not least: Don’t believe naively what you read...

● The RATP App, v5.4.1
  ○ “We don’t collect any personal information”

● Really?
  ○ “list of active Apps, MAC address, device name, position (20m accuracy), permanent ID”
  ○ sent to Adgoji (SSL) or Sofialis (cleartext)

● See our blog: : part-1 et part-2
Questions/Remarks?

Thanks