A Study of the Feasibility of Co-located App Attacks against BLE

and a

Large-Scale Analysis of the Current Application-Layer Security Landscape

Pallavi Sivakumaran, Jorge Blasco
Background: Bluetooth Low Energy Data Access and Pairing
<table>
<thead>
<tr>
<th>Handle</th>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00AB</td>
<td>Primary Service</td>
<td>Heart Rate Service</td>
</tr>
<tr>
<td>0x00AC</td>
<td>Characteristic</td>
<td>Heart Rate Measurement</td>
</tr>
<tr>
<td>0x00AD</td>
<td>Heart Rate Measurement</td>
<td>80bpm</td>
</tr>
<tr>
<td>0x00AE</td>
<td>Characteristic</td>
<td>Heart Rate Control Point</td>
</tr>
<tr>
<td>0x00AF</td>
<td>Heart Rate Control Point</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Handle</td>
<td>Attribute Type</td>
<td>Attribute Value</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0x00AB</td>
<td>Primary Service</td>
<td>Heart Rate Service</td>
</tr>
<tr>
<td>0x00AC</td>
<td>Characteristic</td>
<td>Heart Rate Measurement</td>
</tr>
<tr>
<td>0x00AD</td>
<td>Heart Rate Measurement</td>
<td>80bpm</td>
</tr>
<tr>
<td>0x00AE</td>
<td>Characteristic</td>
<td>Heart Rate Control Point</td>
</tr>
<tr>
<td>0x00AF</td>
<td>Heart Rate Control Point</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Read Request for Handle 0x00AD
(“Heart Rate Measurement”)

Read Response for Handle
0x00AD = 80bpm
- Permissions
  - Access
  - Authentication (pairing)
  - Authorization
Q1: Can an Unauthorised App Access Protected Data?
Co-located App
Data Access Scenario #1
Scan and Connect

Connect GATT, Read Request for Handle 0x00AD
Connect GATT, Read Request for Handle 0x00AD

ERROR: Insufficient Authentication
Scan and Connect

Connect GATT, Read Request for Handle 0x00AD

ERROR: Insufficient Authentication

Pairing Exchange
Connect GATT, Read Request for Handle 0x00AD
Pairing Exchange

ERROR: Insufficient Authentication

Scan and Connect

Bluetooth pairing request
Device
TestBLE
Pairing code
147632
CANCEL     PAIR

Bluetooth pairing request
Device
TestBLE
Pairing code
147632
Pairing grants access to every aspect of your life.
CANCEL     PAIR
Scan and Connect

Connect GATT, Read Request for Handle 0x00AD

ERROR: Insufficient Authentication

Pairing Exchange

Encrypted link

Read/Write Request for Handle 0x00AD

Response for Handle 0x00AD
Scan and Connect

Link Encryption Using Stored Credentials

Encrypted link

Connect GATT, Read/Write Request for Handle 0x00AD

Response for Handle 0x00AD
User is not aware

Link Encryption Using Stored Credentials

Encrypted link

Connect GATT, Read/Write Request for Handle 0x00AD

Response for Handle 0x00AD
Inside Job: Understanding and Mitigating the Threat of External Device Mis-Bonding on Android

Muhammad Naveed1, Xiaoyong Zhou2, Soteris Demetriou1, Xingfeng Wang3, Carl A Gunter4
1Department of Computer Science, University of Illinois at Urbana-Champaign
2School of Informatics and Computing, Indiana University at Bloomington
{naveed2, sdm2012, cgunter}@illinois.edu, {zhou, xfw7}@indiana.edu

Abstract—Today’s smartphones can be armed with many types of external devices, such as medical devices and credit card readers, that enrich their functionality and enable them to be used in application domains such as healthcare and retail. This new development comes with new security and privacy challenges. Existing phone-based operating systems, Android in particular, are not ready for protecting authorized use of these external and navigation but also for such critical activities as personal financial management and healthcare. These new applications often rely on the hardware not already built into the smartphone and therefore need an external device to work together with the phone through Bluetooth, Near-Field Communication (NFC) and other channels. A prominent example is smartphone-enabled healthcare devices with or without mobile workers.
Co-located App
Data Access Scenario #2
Scan and Connect

... Pairing Exchange

Encrypted link
Get Connected Devices
Get Connected Devices

Encrypted link

Connect to GATT Server

Read/Write Request for Handle 0x00AD

Response for Handle 0x00AD

Opportunistic data access
(not possible in Classic Bluetooth)
Get Connected Devices

- Encrypted link
- Connect to GATT Server
- Read/Write Request for Handle 0x00AD
- Response for Handle 0x00AD

No scanning required
GoodApp
needs access to

- Bluetooth  
- Bluetooth Admin  
- Location  
- Internet  

Google Play  ACCEPT

Allow GoodApp to access your location?

DENY  ALLOW
**GoodApp**

needs access to

- Bluetooth
- Bluetooth Admin
- Location
- Internet

Google Play: **ACCEPT**

**EvilApp**

needs access to

- Bluetooth
- Internet
- NFC
- Other

Google Play: **ACCEPT**

**Allow GoodApp** to access your location?

DENY    ALLOW
Summary of unauthorised data access scenarios:

- **Scenario #1**
  - Malicious app can access data at any time (as long as Bluetooth is on and BLE device is nearby, of course!).
  - Malicious app requires BLUETOOTH, BLUETOOTH_ADMIN, LOCATION permissions (user may view the app as being intrusive).

- **Scenario #2**
  - Malicious app can only access data when good app is connected.
  - Malicious app requires only BLUETOOTH permission (activity less visible to user/app appears more benign).
Protecting BLE Data
Several stakeholders

- Android (and other OSs)
  - Don’t allow multiple apps to share a BLE connection.
  - Associate pairing credentials with the app that triggered pairing?
- Bluetooth SIG
  - Add application layer protection and modify sensitive profiles. Flexibility?
- Developers
  - Implement application-layer security 😞
  - Awareness? (We informed the Android Security Team and the Bluetooth SIG of the need for documentation regarding this issue.)
Q2: What Proportion of Devices Have End-to-End Protection for BLE Data?
BLECryptracer:

- Tool to identify the presence of cryptographically-processed BLE data.
- Analyses Android APKs:
  1. Use Androguard to obtain smali.
  2. Identify BLE data access methods.
  3. Perform “slicing” to trace through smali code, and see if we hit cryptographic libraries.
If cryptographically-processed BLE data is identified, BLECryptracer assigns the result a “confidence level”:

- **High**: If BLE-crypto link is identified via direct register value transfers and/or immediate method invocations.
- **Medium**: If BLE-crypto link is identified by considering abstract/interface methods and/or associated registers.
- **Low**: If crypto is identified in any instruction within any previously encountered method (originating from BLE data access call).
<table>
<thead>
<tr>
<th>Access</th>
<th>Tool</th>
<th>Conf. Level</th>
<th>App Set</th>
<th>Detected</th>
<th>TP</th>
<th>FP</th>
<th>TN</th>
<th>FN</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Amandroid</td>
<td>N/A</td>
<td>92</td>
<td>49</td>
<td>44</td>
<td>5</td>
<td>10</td>
<td>33</td>
<td>90%</td>
<td>57%</td>
<td>70%</td>
</tr>
<tr>
<td>Read</td>
<td>BLE Crypt-racer</td>
<td>High</td>
<td>92</td>
<td>62</td>
<td>58</td>
<td>4</td>
<td>11</td>
<td>19</td>
<td>94%</td>
<td>75%</td>
<td>83%</td>
</tr>
<tr>
<td>Write</td>
<td>BLE Crypt-racer</td>
<td>Med</td>
<td>30</td>
<td>11</td>
<td>7</td>
<td>4</td>
<td>7</td>
<td>12</td>
<td>64%</td>
<td>37%</td>
<td>47%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>19</td>
<td>12</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>67%</td>
<td>67%</td>
<td>67%</td>
</tr>
<tr>
<td></td>
<td>Amandroid</td>
<td>N/A</td>
<td>92</td>
<td>56</td>
<td>49</td>
<td>7</td>
<td>8</td>
<td>28</td>
<td>88%</td>
<td>64%</td>
<td>74%</td>
</tr>
<tr>
<td></td>
<td>BLE Crypt-racer</td>
<td>High</td>
<td>92</td>
<td>50</td>
<td>46</td>
<td>4</td>
<td>11</td>
<td>31</td>
<td>92%</td>
<td>60%</td>
<td>72%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Med</td>
<td>42</td>
<td>22</td>
<td>19</td>
<td>3</td>
<td>8</td>
<td>12</td>
<td>86%</td>
<td>61%</td>
<td>72%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>20</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>7</td>
<td>50%</td>
<td>42%</td>
<td>45%</td>
</tr>
</tbody>
</table>
Real-world APKs

- Executed against 18,929 APKs (from Androzoo) that have BLE data access calls.
BLECryptracer Results

**BLE Reads**
- 46%
- 24.5%
- 28.5%
- 1%

**BLE Writes**
- 54%
- 25%
- 19%
- 2%

Legend:
- High
- Medium
- Low
- None
Several APKs implement BLE functionality via 3rd party libraries.

- Beacon, DFU, BLE "helper"/wrappers...
- BLE writes: 63% APKs solely use libraries.
- BLE reads: 58% use only libraries.

App-specific BLE data access methods less likely to incorporate crypto.
% APKs with Cryptographically Processed BLE Data

- **Health & Fitness**
  - High: 50%
  - Medium: 30%
  - Low: 20%

- **Lifestyle**
  - High: 50%
  - Medium: 30%
  - Low: 20%

- **Business**
  - High: 50%
  - Medium: 30%
  - Low: 20%

- **Travel & Local**
  - High: 90%
  - Medium: 10%

- **Shopping**
  - High: 50%
  - Medium: 30%
  - Low: 20%

- **Maps & Navigation**
  - High: 50%
  - Medium: 30%
  - Low: 20%

- **Medical**
  - High: 10%
  - Medium: 20%
  - Low: 70%

Legend:
- **High**
- **Medium**
- **Low**
% APKs with Cryptographically Processed BLE Data

- Health & Fitness
- Lifestyle
- Business
- Travel & Local
- Shopping
- Maps & Navigation
- Medical

- High
- Medium
- Low
- Cryptographical correctness (CogniCrypt)
  - ECB or other bad mode
  - Hardcoded keys
  - Non-random IVs
  - Incomplete operations
In Summary...
Pairing-protected attributes on the BLE device can be read and written by any application on the Android device.

Regardless of pairing method.

Opportunistic data access enables malicious apps to request fewer permissions than legitimate apps.
Different stakeholders involved. Difficult to determine responsibility.

Currently, security is in the hands of developers.

Almost half of all BLE APKs don’t protect BLE reads/writes. Also, bad crypto practices in some that do.

70% of “Medical” apps don’t protect BLE data.

https://github.com/projectbtle/BLECryptracer
Thank You