How Reliable is My Wearable: A Fuzz Testing-based Study

Edgardo Barsallo Yi, Amiya Maji, Saurabh Bagchi

Purdue University

Outline

- Motivation and Background
  - Android and Android Wear Overview
  - Approach to Evaluate Reliability: Qui-Gon Jinn (QGJ)
  - Evaluation
  - Conclusion and Lessons Learned
Motivation

- Reliability of Android is well explored but wearables come with a new set of challenges
- Wearable devices are sensor rich
- Devices have limited resources
  - Display area, computing power, volatile and non-volatile memory, battery size
- More background work (services) than foreground work (activities)
- Communication pattern where many apps are controlled by a mobile counterpart
- A large use-case is monitoring, accumulation and dissemination of health and fitness data

Android Wear (AW)

- The most popular wearable OS (released in 2014): We use 2017 release
- User Interface (UI) is designed to require minimal human interaction (micro transactions)
- Applications are more services driven, in contrast to Android applications, which usually have rich GUI
- AW makes heavy use of sensors
  - Common use case of fitness and health monitoring
Android IPC: Background

- Android programming model is based on passing **Intent** messages among the components.
- An **Intent** describes an operation to be performed and data to perform it
  - The basic information includes: Action, Category, Data, Component, and Extras
  - Types: Explicit and Implicit

Outline

- Motivation and Background
  - Approach to Evaluate Reliability: *Qui-Gon Jinn (QGJ)*
- Evaluation
- Conclusion and Lessons Learned
Approach to Evaluate Reliability

- Evaluate robustness of Android Wear apps by injecting fuzzed Intents
- Discover vulnerabilities through random and mutated Intents
- Propose recommendations for improving the robustness of Android Wear apps

Qui-Gon Jinn (QGJ)

- QGJ is a user-level app and does not require any system-level permission
- QGJ-Master
  - Generates intents based on Fuzz Injection Campaigns (FIC)
  - Supports fuzz injection of Activities and Services components
- QGJ-UI
  - Mutates semi-valid and random UI events
  - Injects UI events using Android Debug Bridge (adb)
SB1  Is this true? No permission or no system-level permission?
   Saurabh Bagchi, 6/24/2018

EBY1  QGJ does not require any additional permission to fuzz applications. Just the basic permission needed by an app
   Edgardo Barsallo Yi, 6/25/2018
### Fuzz Injection Campaigns

- **Semi-valid Action and Data**
  - `{act=ACTION_EDIT, dat=file://sdcard/file.txt, cmp=some.component.name}`
  - `{act=ACTION_DIAL, data=file://sdcard/file.txt, cmp=some.component.name}`

- **Blank Action or Data**
  - `{act=ACTION_VIEW, dat=https://youtu.be/j5dMnAP242Z, cmp=some.component.name}`

- **Random Action or Data**
  - `{act=ACTION_EDIT, dat=content://contacts/people/1, cmp=com.android.contacts}`
  - `{act=ACTION_EDIT, dat=q1w2e3Q!W@E#, cmp=com.android.contacts}`

- **Random Extras**
  - `{act=ACTION_INSERT, dat=content://contacts/people/1, cmp=com.android.contacts (has extras)}`
  - `{act=ACTION_INSERT, data=content://contacts/people/1, cmp=com.android.contacts (has random extras)}`

---

### Outline

- Motivation and Background
- Approach to Evaluate Reliability: *Qui-Gon Jinn* (QGJ)
- Evaluation
- Conclusion and Lessons Learned
SB2  Explain each type of FIC by giving the correct and the fuzzed versions.
Saurabh Bagchi, 6/24/2018

SB3  Which is random here? And what is "Data"?
Saurabh Bagchi, 6/24/2018

EBY6  Either the action or the data can be random, but not both at the same time.
Edgardo Barsallo Yi, 6/26/2018

EBY7  Data: An URI that represents the data item to be operated
Edgardo Barsallo Yi, 6/26/2018
Target Applications

- **Categories**: Health/Fitness and Not Health/Fitness
  - Based on the fact that health/fitness tracking apps are prominent in AW ecosystem (use of hardware and software sensors)
- **Classification**: Built-in and Third-party apps
  - Built-in apps are pre-installed on the wearable device, while third-party apps are installed by the user
- **Maturity Level**: Third-party apps with at least 1M downloads from the Google Play Store

- **Comparison between Android and AW Ecosystem**:
  - We conducted similar experiments on Android using QGJ-Main
  - We focused on Android built-in apps which are often used by third-party application for implementing common functionalities (com.android)

---

Experimental Configuration

- **QGJ-Master**
  - Android 7.1.1 (released Dec 2016)
  - Android Wear 2.0 (released Feb 2017)
- **QGJ-UI**
  - Android Wear 2.0 (Emulator)
- **Applications**

<table>
<thead>
<tr>
<th>OS</th>
<th>Classification</th>
<th>#</th>
<th># Activities</th>
<th># Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>AW</td>
<td>Health/Fitness</td>
<td>2</td>
<td>81</td>
<td>34</td>
</tr>
<tr>
<td>AW</td>
<td>Health/Fitness</td>
<td>11</td>
<td>80</td>
<td>59</td>
</tr>
<tr>
<td>AW</td>
<td>No Health/Fitness</td>
<td>9</td>
<td>168</td>
<td>188</td>
</tr>
<tr>
<td>AW</td>
<td>No Health/Fitness</td>
<td>24</td>
<td>185</td>
<td>117</td>
</tr>
<tr>
<td><strong>AW Total</strong></td>
<td></td>
<td><strong>46</strong></td>
<td><strong>514</strong></td>
<td><strong>398</strong></td>
</tr>
<tr>
<td>A</td>
<td>com.android</td>
<td>63</td>
<td>595</td>
<td>218</td>
</tr>
</tbody>
</table>
**Error Manifestations**

- **System Reboot**
  - The OS reaches an unrecoverable state and the device reboots
- **Crash**
  - Application crashes due inability to handle malformed intents
- **Hang or unresponsive**
  - The application experiences temporary unresponsiveness or freezes permanently
- **No effect**
  - No effect or failure manifestation due to the malformed injection

**Distribution of Behaviors Among Fuzz Intent Campaigns**

<table>
<thead>
<tr>
<th></th>
<th>Reboot</th>
<th>Crash</th>
<th>Hang</th>
<th>No Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Health</td>
<td>No Health</td>
<td>Health</td>
<td>No Health</td>
</tr>
<tr>
<td>Semi-valid</td>
<td>8%</td>
<td>0%</td>
<td>23%</td>
<td>30%</td>
</tr>
<tr>
<td>Blank Action or Data</td>
<td>0%</td>
<td>0%</td>
<td>31%</td>
<td>24%</td>
</tr>
<tr>
<td>Random Action or Data</td>
<td>0%</td>
<td>0%</td>
<td>31%</td>
<td>33%</td>
</tr>
<tr>
<td>Random Extras</td>
<td>0%</td>
<td>3%</td>
<td>15%</td>
<td>30%</td>
</tr>
</tbody>
</table>

- System Reboots occurred on both categories
- Injection has no effect at roughly the same rate (~70%) on both categories
SB4  This is not clear. Please reword.
Saurabh Bagchi, 6/24/2018

EBY5  Reworded.
Edgardo Barsallo Yi, 6/25/2018
Reliability per Category

No significant difference between **Health/Fitness** apps and other apps

**Built-in** applications showed more failures compared to **Third Party** apps

How Reliable is the Wearable Software Stack?

- Distribution of error manifestation among the application components
  - 13% of components reported failures
  - Crashes are more common than unresponsive manifestation (9X)
  - System reboots affected less than 1% of components
### Distribution of Crashes

<table>
<thead>
<tr>
<th>Exception</th>
<th>#Crashes</th>
<th>%</th>
<th>Exception</th>
<th>#Crashes</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>NullPointerException</td>
<td>54</td>
<td>30.9%</td>
<td>NullPointerException</td>
<td>42</td>
<td>53.2%</td>
</tr>
<tr>
<td>ClassNotFoundException</td>
<td>46</td>
<td>26.3%</td>
<td>IllegalStateException</td>
<td>10</td>
<td>12.7%</td>
</tr>
<tr>
<td>IllegalArgumentException</td>
<td>31</td>
<td>17.7%</td>
<td>IllegalArgumentException</td>
<td>9</td>
<td>11.4%</td>
</tr>
<tr>
<td>IllegalStateException</td>
<td>10</td>
<td>5.7%</td>
<td>ActivityNotFoundException</td>
<td>4</td>
<td>5.1%</td>
</tr>
<tr>
<td>RuntimeException</td>
<td>9</td>
<td>5.1%</td>
<td>Exception</td>
<td>4</td>
<td>5.1%</td>
</tr>
<tr>
<td>ActivityNotFoundException</td>
<td>7</td>
<td>4.0%</td>
<td>WindowManager$BadTokenException</td>
<td>3</td>
<td>3.8%</td>
</tr>
<tr>
<td>UnsupportedOperationException</td>
<td>6</td>
<td>3.4%</td>
<td>ClassNotFoundException</td>
<td>3</td>
<td>3.8%</td>
</tr>
<tr>
<td>Others</td>
<td>12</td>
<td>6.9%</td>
<td>Others</td>
<td>4</td>
<td>5.1%</td>
</tr>
</tbody>
</table>

- **75%** - Android
- **73%** - Android Wear

### Resilience against UI injection

- **QGJ-Master** focus on the communication between components (either starting an Activity or a Service)
  - After, an activity or service has been started, some user interaction (UI events) take places.
  - QGJ-UI emulates this interaction to test the robustness of apps
- **No system crash** during the UI injection
- **Fewer number of exceptions and crashes** than QGJ-Master
  - QGJ-UI only injects events to launcher activities
  - **adb** tools have a robust input validation
System Crashes from User-level Application

• No extra permissions at install time
• The manifestation depends on the transient state of the device
  – The reboots were not triggered by single intent, but due to error propagation across components and software aging through repeated fuzzing campaigns.
• 2 apps crashed Android Runtime
  – A health app (third party) raised a SIGABRT signal during the experiment, after experiencing some unresponsiveness.
  – A built-in app raised a SIGSEVG signal. The app crashed multiple times during the injection before triggering the reboot.

Outline

• Motivation and Background
• Approach to Evaluate Reliability: *Qui-Gon Jinn* (QGJ)
• Evaluation

Conclusion and Lessons Learned
SB5  Not clear. Please reword.
Saurabh Bagchi, 6/24/2018

EBY4  Reworded.
Edgardo Barsallo Yi, 6/25/2018
Conclusions and Insights

• **Distribution of exception** types differ between Android and Android Wear
  – Input validation (e.g. NullPointerException) is still the major cause of crashes
  – High incidence of crashes on AW are tied to the state of the application/device (e.g. IllegalStateException)

• **Software Aging**: Further research on software aging can help identify and mitigate transient system reboots that are state dependent

• **Input Validation**: Although Android’s input validation has improved compared to earlier work [Maji, DSN’12] it is still a major cause for crashes.
  – Need more awareness and tool support
This seems like a vague hand-wavy insight - instead provide some evidence behind the top level bullet.
Saurabh Bagchi, 6/24/2018

I added some notes too.
Edgardo Barsallo Yi, 6/25/2018

This has not been mentioned anywhere in the rest of the presentation. What is the context or evidence behind the claim?
Saurabh Bagchi, 6/24/2018

I included some reference to software aging on slide 19: System Crashes from User-level Application.
Edgardo Barsallo Yi, 6/25/2018