BadBluetooth: Breaking Android Security Mechanisms via Malicious Bluetooth Peripherals

Fenghao Xu¹, Wenrui Diao²,³, Zhou Li⁴, Jiongyi Chen¹, Kehuan Zhang¹

The Chinese University of Hong Kong¹, Shandong University², Jinan University³, University of California, Irvine⁴
Motivation

• Bluetooth Everywhere

• Bluetooth device type: mouse/keyboard, headset ...

• Rich functionalities
Motivation

• Attractive attack interface
  ▪ More than 200 CVEs... implementation vulnerabilities (e.g., driver’s bugs)
  ▪ Privacy leakage on Bluetooth device.
  ▪ Mis-bonding. App can access any paired device. [Naveed et al. NDSS’14]

• Motivated Observation: How do users know the device type?
  ▪ Appearance?
  ▪ Displayed name/icons?
  ▪ Pairing process? (e.g., input PIN)
  ▪ Potential attacks!
Our Work

Study current Bluetooth design, focus on Android phone

• Identify several design weaknesses
  ▪ Bluetooth device: Profile Authentication
  ▪ Android app: Coarse-grained permission

• New Attack with 3 showcases

• Defense solution and evaluation
Outline

• Background

• Design weaknesses

• New attack with 3 showcases

• Defense solution and evaluation
Background: Bluetooth

- Bluetooth Profile – A general functionality (e.g., Headset Profile)

- Bluetooth Connection
  - Discovering
  - Pairing – *link key*
  - Profile connection
    - Multiple profiles at the same time

### Android Supported Profiles

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>HID</td>
<td>Human Interface Device</td>
<td>Keyboard</td>
</tr>
<tr>
<td>PAN</td>
<td>Personal Area Networking</td>
<td>Network Hotspot</td>
</tr>
<tr>
<td>HFP/HSP</td>
<td>Hands-Free/Headset</td>
<td>Wireless Headset</td>
</tr>
<tr>
<td>SAP</td>
<td>SIM Access</td>
<td>Car Kit</td>
</tr>
<tr>
<td>MAP</td>
<td>Message Access</td>
<td>Car Kit</td>
</tr>
<tr>
<td>PBAP</td>
<td>Phone Book Access</td>
<td>Car Kit</td>
</tr>
<tr>
<td>OPP</td>
<td>Object Push</td>
<td>File Transfer</td>
</tr>
<tr>
<td>A2DP</td>
<td>Advanced Audio Distribution</td>
<td>Wireless Speaker</td>
</tr>
<tr>
<td>AVRCP</td>
<td>Audio/Video Remote Control</td>
<td>Remote Media Controller</td>
</tr>
<tr>
<td>DIP</td>
<td>Device ID</td>
<td>Extra Device Information</td>
</tr>
<tr>
<td>HDP</td>
<td>Health Device</td>
<td>Blood Pressure Kit</td>
</tr>
<tr>
<td>SPP</td>
<td>Serial Port</td>
<td>App-specific</td>
</tr>
</tbody>
</table>
Background: Bluetooth

- Bluetooth Profile – A general functionality (e.g., Headset Profile)

- Bluetooth Connection
  - Discovering
  - Pairing – *link key*
  - Profile connection
    - Multiple profiles at same time
Outline

• Background

• Design weaknesses

• New attack with 3 showcases

• Defense solution and evaluation
Weakness – Profile Authentication

- Inconsistent Authentication Process on Profile
  - Device-level authentication
  - No profiles indication on pairing
  - Show a list in details menu if paired
  - Device can change profile dynamically!
Weakness – Coarse-grained Permissions

• Android Bluetooth Permissions
  ▪ `BLUETOOTH`, `BLUETOOTH_ADMIN` – normal level (implicit granted)
  ▪ For device discovery: `ACCESS_COARSE_LOCATION`
    (not required with known MAC address)

• App can access any paired device [Naveed et al. NDSS’14]

• Mis-aligned with profiles
Weakness – Coarse-grained Permissions

- Mis-aligned with profiles
  - normal-permission app (Bluetooth permissions)
  - initiate “system-level” Bluetooth connection (on behalf of the phone)
  - “Hidden” APIs - Java reflection or replace SDK
  - App privilege escalation through external device

1: app-device “direct” connection (e.g., Serial Port Profile)
2: system-device connection (e.g., Human Interface Device)
Weakness – More

• Silent Pairing
  ▪ Pairing is supposed to involve user interaction. (e.g., numerical comparison, input PIN)
  ▪ Device - neither display nor input ability
  ▪ Pair with the device stealthily

• Deceivable and vague UI
  ▪ Device name and icon – easy to cheating
  ▪ Class Device Number (CoD)

<table>
<thead>
<tr>
<th>Icon</th>
<th>CoD</th>
<th>Class Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🖥️</td>
<td>0x100</td>
<td>Computer</td>
</tr>
<tr>
<td>📞</td>
<td>0x200</td>
<td>Phone</td>
</tr>
<tr>
<td>🎧</td>
<td>0x404</td>
<td>Audio/Video-Wearable Headset</td>
</tr>
<tr>
<td>🎧</td>
<td>0x418</td>
<td>Audio/Video-Headphones</td>
</tr>
<tr>
<td>🌡️</td>
<td>0x500</td>
<td>Peripheral</td>
</tr>
<tr>
<td>☛</td>
<td>0x540</td>
<td>Peripheral-Keyboard</td>
</tr>
<tr>
<td>🖱️</td>
<td>0x580</td>
<td>Peripheral-Pointing device</td>
</tr>
<tr>
<td>⏰</td>
<td>0x600</td>
<td>Imaging</td>
</tr>
<tr>
<td>📞</td>
<td>0x000</td>
<td>General Bluetooth</td>
</tr>
</tbody>
</table>
Outline

• Background

• Design weaknesses

• New attack with 3 showcases

• Defense solution and evaluation
BadBluetooth Attack

• Adversary Model
  ▪ Device: various ways
e.g., seller, previous owner hacked;
  exploit device vulnerabilities...
  ▪ App: with two normal-level permissions

• Google Pixel 2 with Android 8.1
• Raspberry Pi 2 running Linux with Bluetooth USB Adapter (CSR8510)
Attack Case 1: Human Interface Device (HID)

- Full functional keyboard and mouse supporting on Android
- Construct global input sequences equivalent to any user actions
Attack Case 1: Human Interface Device (HID)

- Attack Strategy
  - Adaptive – using phone brands and Android version information
  - Input Capability – mouse, keyboard event (including functional keys)
  - Output Capability – KEY_SYSRQ (capture any view on the phone)

<table>
<thead>
<tr>
<th>Linux Key</th>
<th>Code Name</th>
<th>Description (effect on Android)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY_ENTER</td>
<td></td>
<td>Enter Key (click)</td>
</tr>
<tr>
<td>KEY_TAB</td>
<td></td>
<td>Tab Key (select item)</td>
</tr>
<tr>
<td>KEY_SYSRQ</td>
<td></td>
<td>Screenshot</td>
</tr>
<tr>
<td>KEY_COMPOSE</td>
<td></td>
<td>Menu Key (open menu for current app)</td>
</tr>
<tr>
<td>KEY_POWER</td>
<td></td>
<td>Power Key (open/close screen)</td>
</tr>
<tr>
<td>KEY_WWW</td>
<td></td>
<td>Explorer (launch browser app)</td>
</tr>
<tr>
<td>KEY_PHONE</td>
<td></td>
<td>Call (launch phone app)</td>
</tr>
<tr>
<td>KEY_MAIL</td>
<td></td>
<td>Envelope (launch mail app)</td>
</tr>
<tr>
<td>KEY_ADDRESSBOOK</td>
<td></td>
<td>Contacts (launch phone book app)</td>
</tr>
<tr>
<td>KEY_HOMEPAGE</td>
<td></td>
<td>Home Key</td>
</tr>
<tr>
<td>KEY_BACK</td>
<td></td>
<td>Back Key</td>
</tr>
</tbody>
</table>
Attack Case 1: Human Interface Device (HID)

- **Attack: Information Stealing**
  - Screenshot then read

- **Attack: App and System Controlling**
  - Cross-app injection
  - System setting modification
  - Acquire dangerous permissions
  - Restart/shutdown phone...

- **Attack: Beyond the phone**
  - Steal tokens (e.g., website login, SMS code)
  - Open camera...
Attack Case 2: Personal Area Networking (PAN)

- Attack: Network Sniffing and Spoofing
  - Device as NAP
  - Force traffic to go through the device
  - MITM attack – sniffing, spoofing

- Attack: Network Consumption
  - Device as PANU
  - App opens the Bluetooth Tethering (global)
  - Share the phone’s network
Attack Case 3: Hands-Free (HFP)

- **Attack: Telephony Control**
  - Answer/reject incoming calls
  - Initiate outgoing calls (arbitrary number)

- **Attack: Voice Command Injection**
  - Google Voice Assistant
  - Trigger and inject voice command
Demo
Outline

• Background

• Design weaknesses

• New attack with 3 showcases

• Defense solution and evaluation
Defense: Profile Binding

- Fine-grained control and better visibility to user
- Bind the device with a permitted profile list, prohibit connection with other profiles

Pair Request → Profile Selection → Binding Policy → Policy Checking → Mediate Connection → Connect → Reject
Defense: Implementation

• Modified Android Open Source Project (AOSP) Android 8.1

• Modules – in the Bluetooth Process
  ▪ Pairing Monitor
  ▪ Binding Policy DB
    ➢ Settings.Secure storage
  ▪ Connection Controller

The white blocks and blue lines represent the defense framework
Defense: Evaluation

• Effectiveness
  ▪ All pairing process is monitored and prompted to users
  ▪ Only explicitly granted profiles can be connected

• Performance
  ▪ `connect()` time delay
  ▪ less than 12% with total time

<table>
<thead>
<tr>
<th>ProfileService Class</th>
<th>Original (μs)</th>
<th>Defense (μs)</th>
<th>Delays (μs)</th>
<th>Total* (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HidService</td>
<td>494.9/63.0</td>
<td>605.5/49.0</td>
<td>110.6</td>
<td>2546.0/589.4</td>
</tr>
<tr>
<td>PanService</td>
<td>235.8/45.8</td>
<td>460.4/43.1</td>
<td>224.6</td>
<td>1890.5/420.5</td>
</tr>
<tr>
<td>HeadsetService</td>
<td>473.5/62.4</td>
<td>522.2/66.5</td>
<td>48.7</td>
<td>2359.3/326.1</td>
</tr>
</tbody>
</table>

*: From upper-layer API call to connection completion (original Android OS).
Summary

• Several weaknesses on Bluetooth design, especially on Android

• We presented the BadBluetooth attack
  ▪ Device: abuse the Bluetooth profile abilities to attack phone
  ▪ App: break Android security mechanisms through a peripheral

• Three concrete attack cases – HID, PAN, HFP

• Defense solution: profile binding
  ▪ fine-grained control
Thank you!