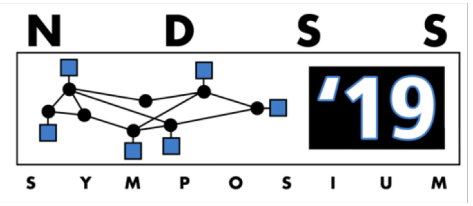


BadBluetooth: Breaking Android Security Mechanisms via Malicious Bluetooth Peripherals

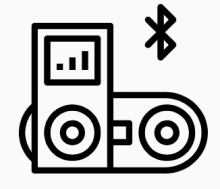
Fenghao Xu[1], Wenrui Diao[2,3], Zhou Li[4], Jiongyi Chen[1], Kehuan Zhang[1]

The Chinese University of Hong Kong[1], Shandong University[2], Jinan University[3],

University of California, Irvine[4]

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

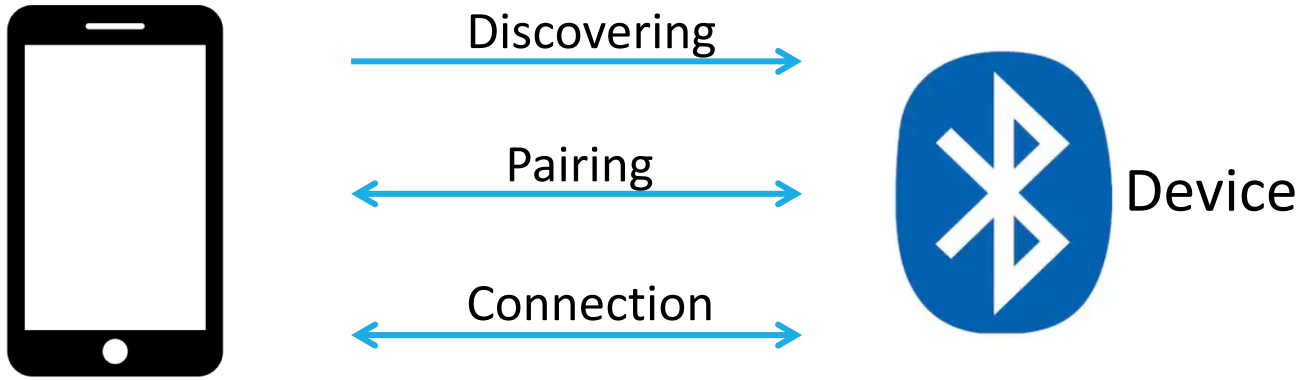
- Bluetooth Profile – A general functionality (e.g., Headset Profile)
- Bluetooth Connection
  - Discovering
  - Pairing – *link key*
  - Profile connection
    - Multiple profiles at same time

Android Supported Profiles

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

Background: Bluetooth

- Bluetooth Profile – A general functionality (e.g., Headset Profile)
- Bluetooth Connection
  - Discovering
  - Pairing – *link key*
  - Profile connection
    - Multiple profiles at same time

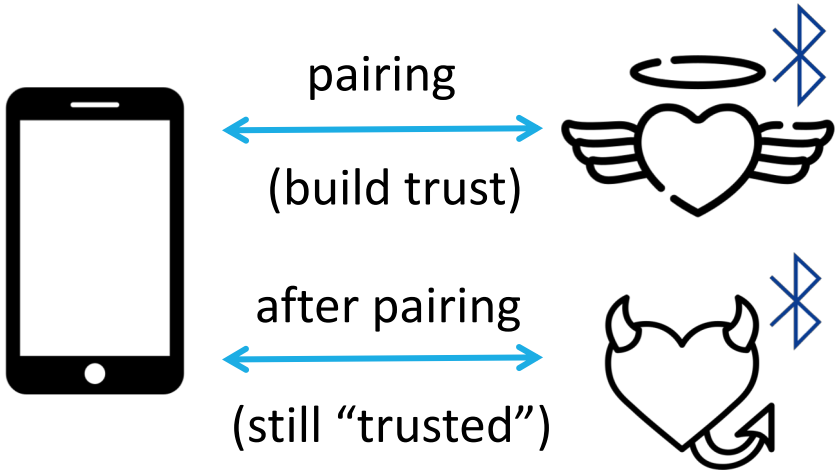


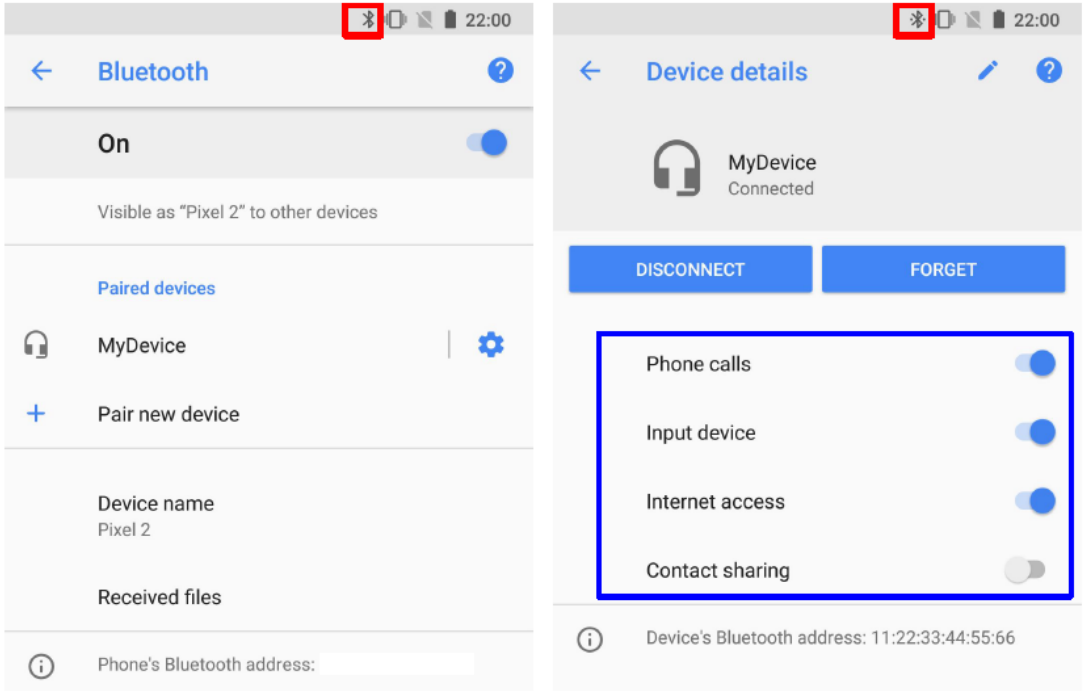
Outline

- 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!





Android Bluetooth Menu

Weakness – Coarse-grained Permissions

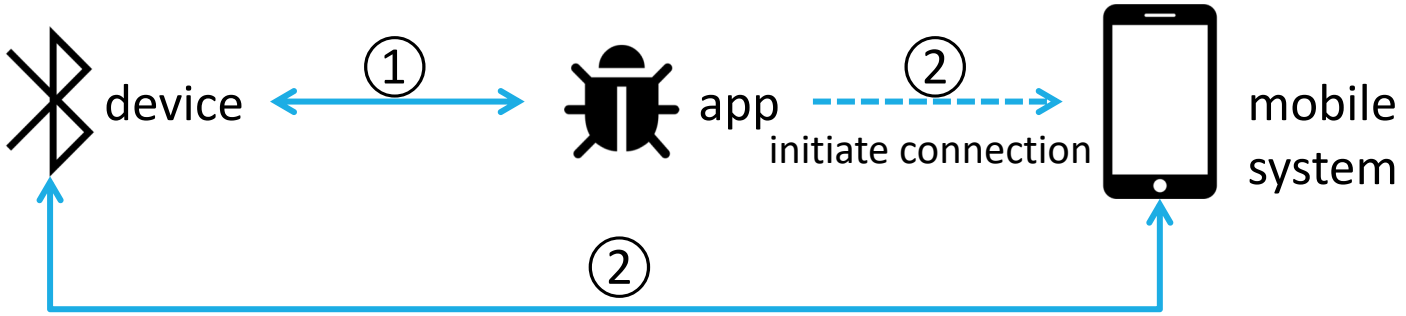
- Android Bluetooth Permissions
  - BLUETOOTH, BLUETOOTH_ADMIN – normal level (implicit granted)
  - For device discovery: ACCESS_COARSE_LOCATION

    (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)

Icon	CoD	Class Description
	0x100	Computer
	0x200	Phone
	0x404	Audio/Video-Wearable Headset
	0x418	Audio/Video-Headphones
	0x500	Peripheral
	0x540	Peripheral-Keyboard
	0x580	Peripheral-Pointing device
	0x600	Imaging
	0x000	General Bluetooth

Outline

- Background
- Design weaknesses
- New attack with 3 showcases
- Defense solution and evaluation

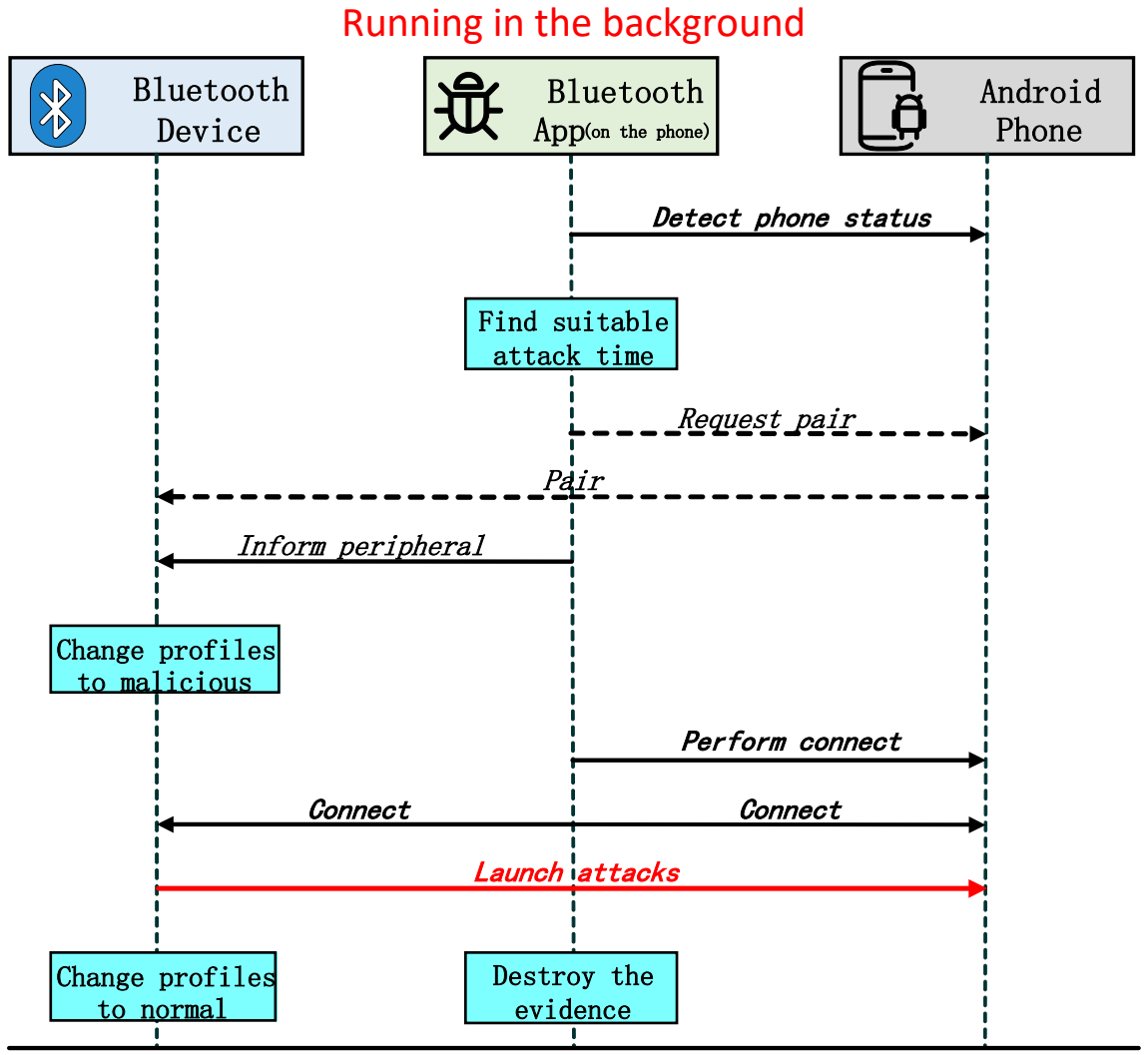
BadBluetooth Attack

- Adversary Model
  - Device: various ways

    e.g., seller, previous owner hacked;

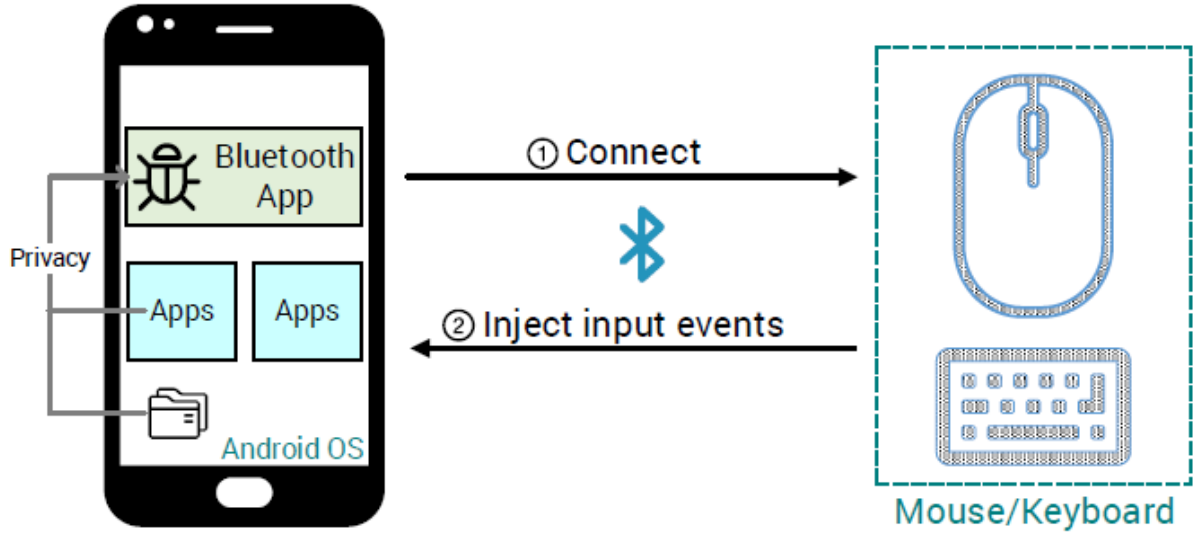
    exploit device vulnerabilities…
  - App: with two normal-level permissions

- 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



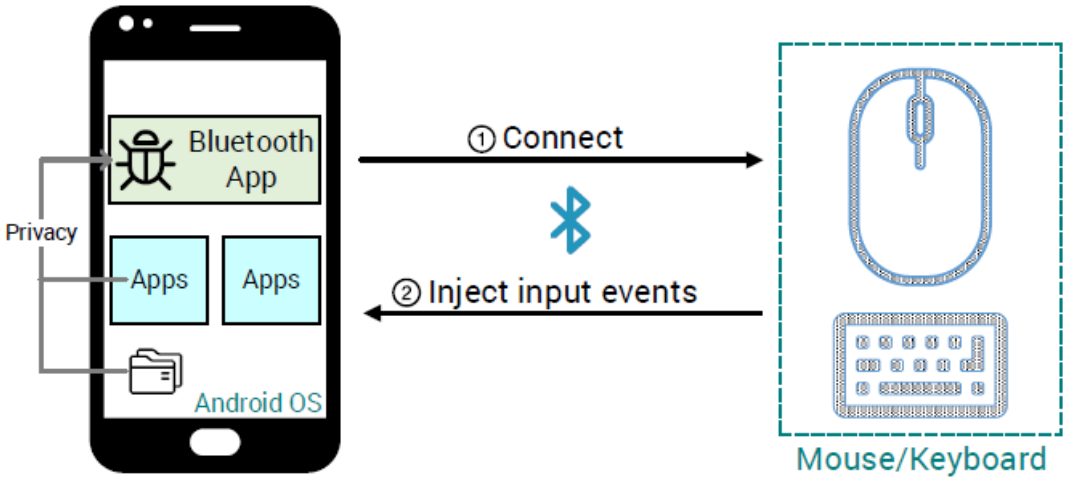
Attack Case 1: Human Interface Device (HID)

- 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)

Linux Key Code Name	Description (effect on Android)
KEY_ENTER	Enter Key (click)
KEY_TAB	Tab Key (select item)
KEY_SYSRQ	Screenshot
KEY_COMPOSE	Menu Key (open menu for current app)
KEY_POWER	Power Key (open/close screen)
KEY_WWW	Explorer (launch browser app)
KEY_PHONE	Call (launch phone app)
KEY_MAIL	Envelope (launch mail app)
KEY_ADDRESSBOOK	Contacts (launch phone book app)
KEY_HOMEPAGE	Home Key
KEY_BACK	Back Key

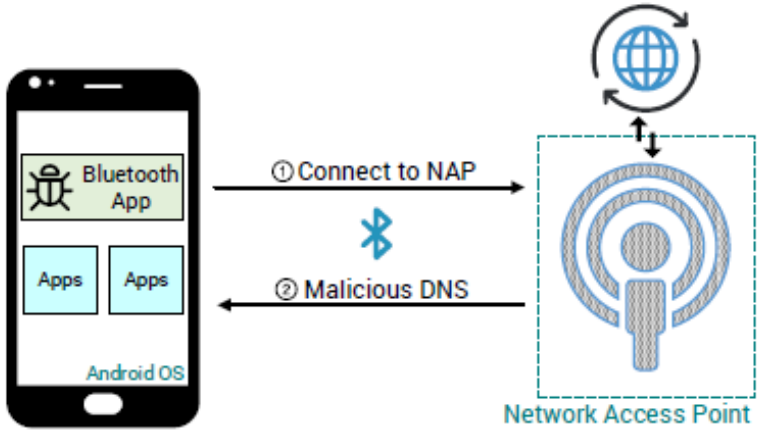
Attack Case 1: Human Interface Device (HID)

- 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…

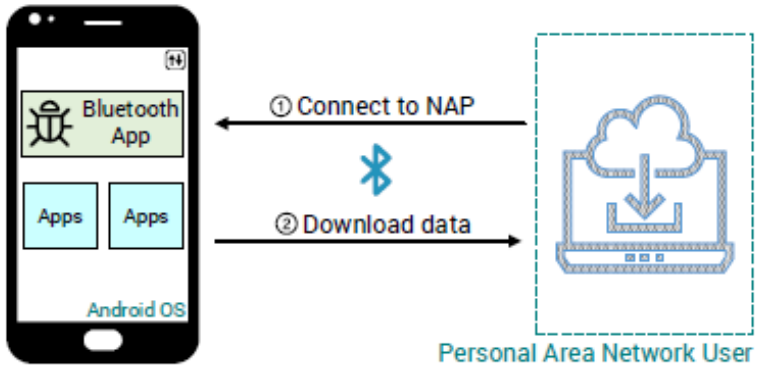


Attack Case 2: Personal Area Networking (PAN)

- 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



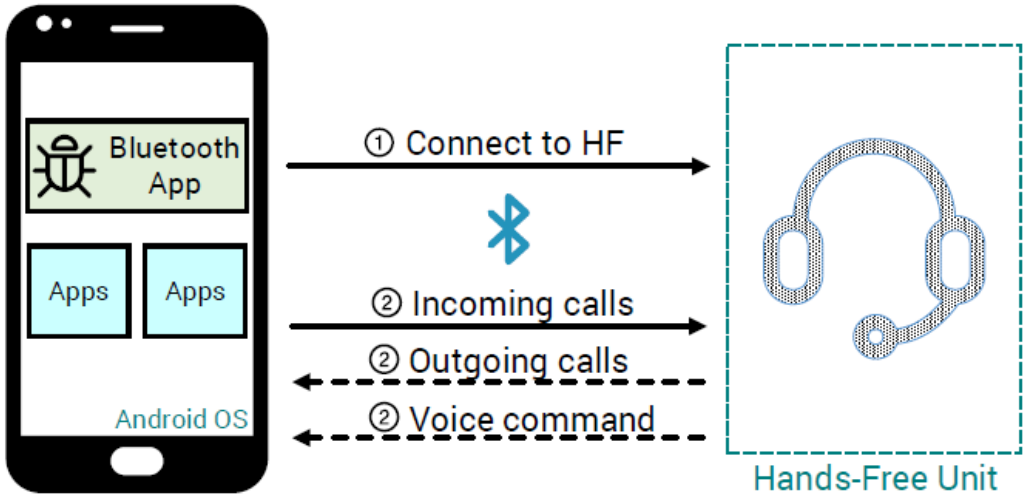
(a) Device as NAP



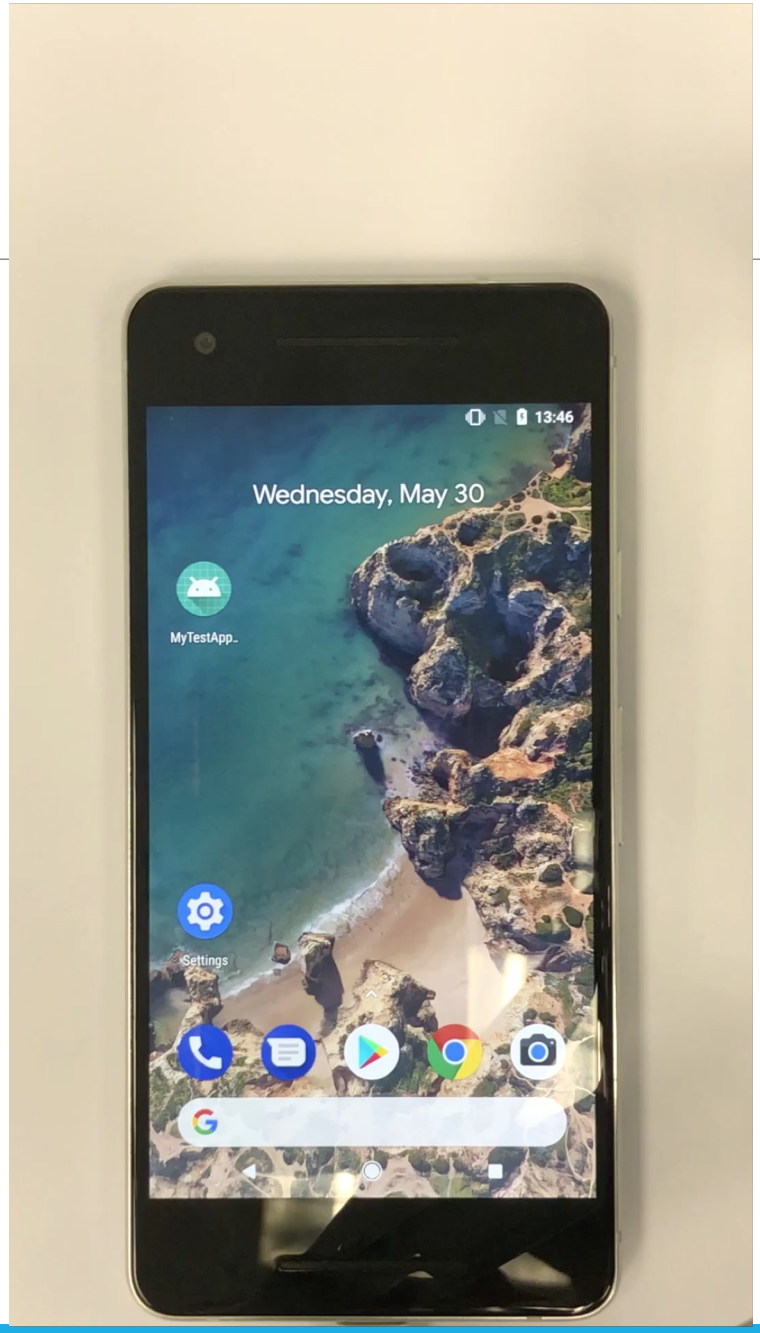
(b) Device as PANU

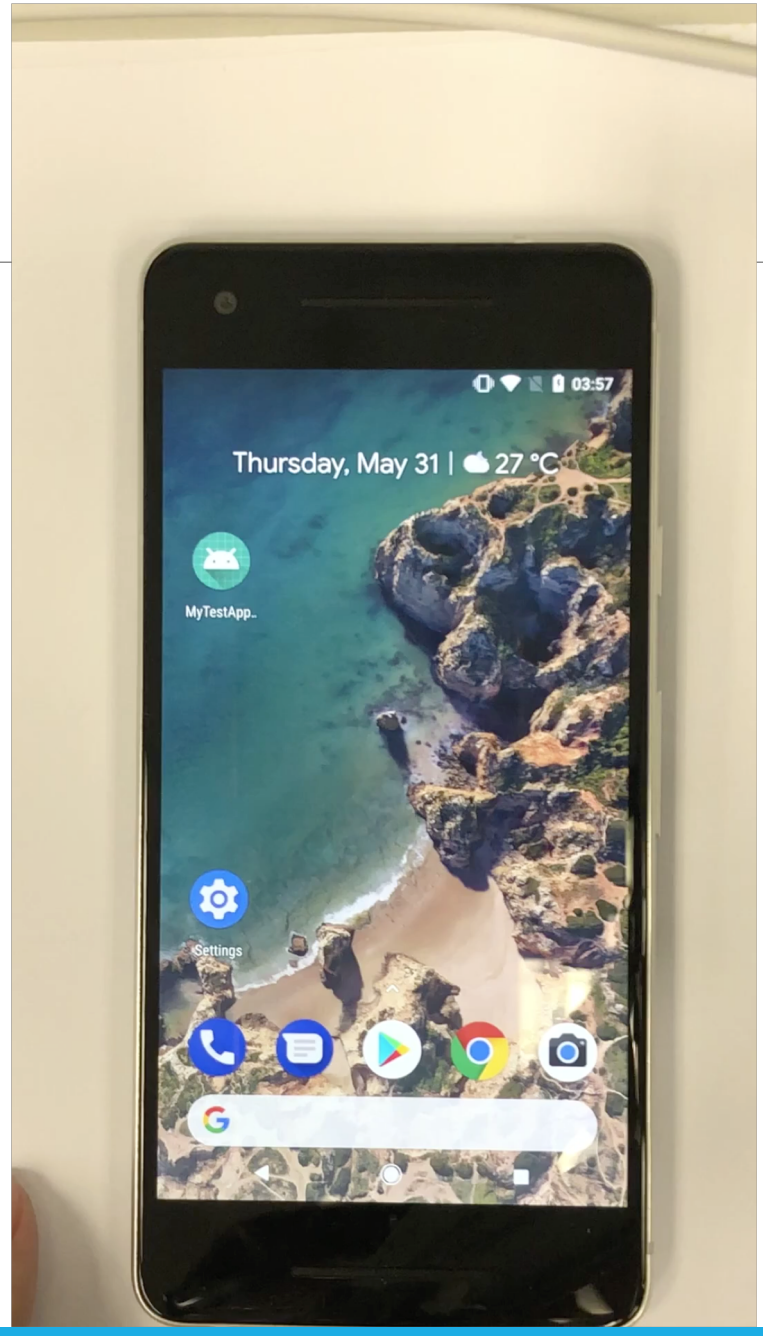
Attack Case 3: Hands-Free (HFP)

- 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


13:46
Wednesday, May 30
MyTestApp..
Settings

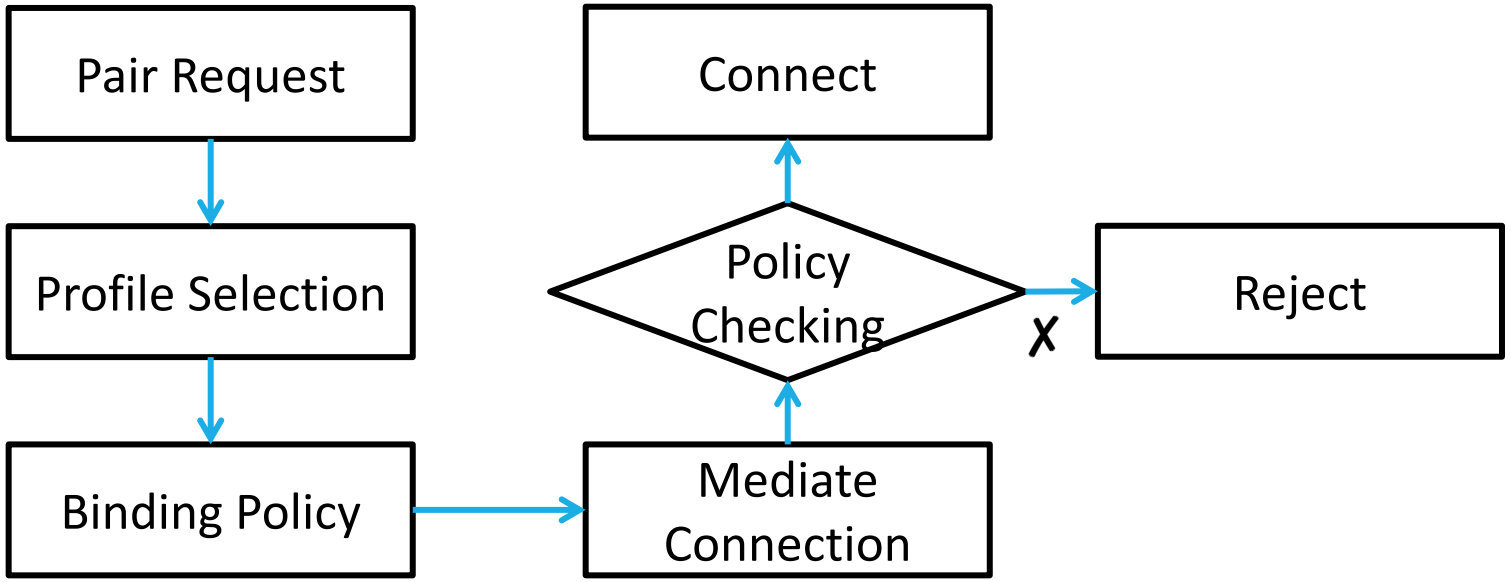

03:57
Thursday, May 31 | 27 °C
MyTestApp..
Settings

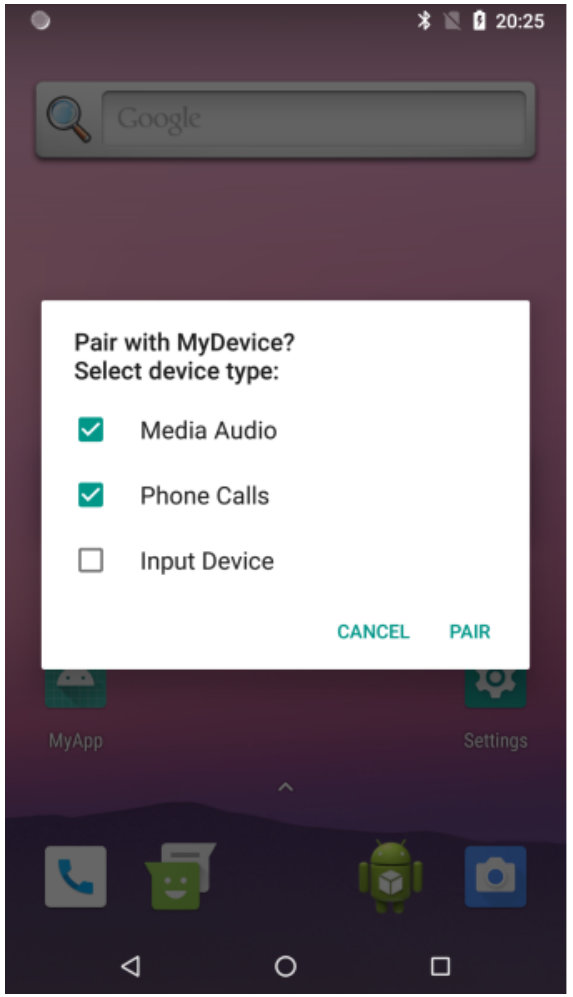
Outline

- 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

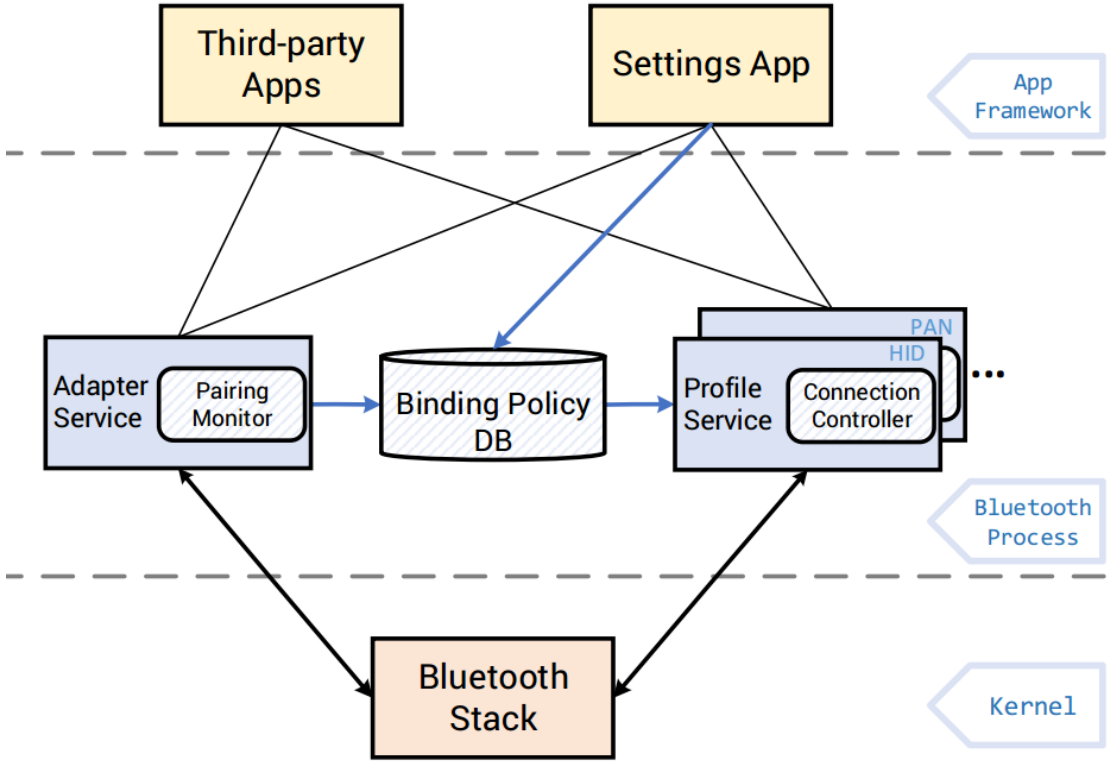




Defense: Implementation

- 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 VII: Profile connection evaluation. (mean/std)

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

*: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

Q&A

Thank you!