Inside Job

Diagnosing Bluetooth Lower Layers Using Off-the-Shelf Devices

Jiska Classen and Matthias Hollick
Technische Universität Darmstadt
Secure Mobile Networking Lab - SEEMOO
Motivation
Use special (but very cheap) hardware, such as microbit/btlejack or Bluefruit LE.

Successfully follow the **hopping** pattern and then **overhear the initial pairing** procedure to extract secret keys (**LE Legacy** pairing in Bluetooth 4.0 and 4.1).

Maybe also active MITM (Bluetooth >= 4.2) required to get encryption keys...

...finally find out what these devices do on the lower layers!
Standard Bluetooth Sniffing Setup (Classic Bluetooth)

- Use special (> $10k) hardware, i.e. Ellisys.
- Open source solutions such as Ubertooth do not support encrypted traffic...
- Successfully follow the **hopping** pattern and then **modify the initial pairing** procedure to extract secret keys, active MITM required (Bluetooth >=4.0).
- If a stronger mode than “Just Works” is used, user needs to ignore the wrong numeric comparison.
- ...finally find out what these devices do on the lower layers!
Bluetooth Lower Layers: Security Perspective

- Bluetooth lower layers are not well-tested.

- If you know the **MAC address**, you can **connect** to a device and get more information, i.e. which LMP version it is running (often equals the **firmware version**).

  "Hi there, I’m a Broadcom Bluetooth 4.1 chip running an attackable LMP minor version of 0x2203…"
Making Bluetooth Lower Layers Accessible

- In the shown Bluetooth **sniffing setup**, initial pairing must be overheard, attacker needs to be in proximity during this pairing that only takes place once.
  → Very artificial setup, typically both **parties are aware of sniffing**, at least if secure pairing modes are used.

- Sniffing does not require MITM, **access on one of the devices** within a connection is sufficient to get all contents of a session despite hopping.
- Lower layer traffic is not embedded within HCI (Host Controller Interface / layer 3) information.
  → Bluetooth **layer 1+2 cannot be observed** out of the box.
- **Modify firmware** of existing chipsets to monitor lower layer traffic.

- SEEMOO already did this for monitor mode on Broadcom Wi-Fi chips.
InternalBlue Based on Binary Patching

Vendor specific HCI (local)

Modify firmware

LMP monitor & injection

Bluetooth

InternalBlue Based on Binary Patching

Vendor specific

HCI (local)

Modify firmware

- Nexus 5 (BCM4339 firmware)
- Only LMP, no LCP
Host Controller Interface

- Message types between host and controller
  0x01 HCI command (connect)
  0x02 ACL data
  0x03 SCO data
  0x04 HCI event (connection established / page timeout)

- Vendor-specific HCI commands (Broadcom+Cypress)
  0xfc4c Write RAM
  0xfc4e Launch RAM

- Syntax for reading, writing and executing RAM is documented for BCM4339.
- Find link manager (LMP/LCP) handlers and patch them to forward information as HCI events.
  (Actually not that easy ... use opcode checks to identify the correct handlers within 11k functions and write assembly. Patches depend on exactly the BCM4339/Nexus 5 chip.)
Platform Independence
Does it work on the newest device?

- We ported InternalBlue from **Nexus 5** to **Raspberry Pi 3/3+** and **Nexus 6P**.

- Tested on CYW20735 Bluetooth 5.0-compliant BT/BLE wireless MCU, it still has READ_RAM, WRITE_RAM, LAUNCH_RAM HCI commands.
  - Firmware version **January 18 2018**
- Reading out the whole firmware and applying temporarily patches without any checks in 2018, thank you BroadcomCypress!

- Reversing could have been faster: `patch.elf` shipped with development software contains **symbol table** for almost every firmware function...
Host Controller Interface With Symbols

- Message types between host and controller
  0x01 HCI command
  0x02 ACL data
  0x03 SCO data
  0x04 HCI event
  0x07 Broadcom diagnostics (not processed by Android, requires modifications)

- Vendor-specific HCI commands
  0xfc4c Write RAM
  0xfc4e Launch RAM
  0xfc58 Send LMP (but packet lengths are checked)

- The macOS Packet Logger supports passive LMP sniffing on some devices. Reversing this binary revealed additional information on the Broadcom diagnostic protocol. Thank you Apple :)

Found with the new symbols.
Broadcom Diagnostics Protocol

- **HCI reversing:**
  - **HCI command to send LMP packets** already included, but packets are checked for validity.
  - **fuzzlmp** patch available for *Nexus 5*.

- **Diagnostics protocol:**
  - **Patch Android driver** to forward diagnostic messages of type 0x07.
  - **Wireshark plugin** to interpret the messages...
  - **LMP and LCP logging** on all **Broadcom chips** (at least 2008-2018).

- **Diagnostics include** even more features, such as special memory access, ACL and SCO statistics, ...
Finding Bugs in Lower Layers
Uninitialized Encryption

- The attacker initiates SSP (Secure Simple Pairing) with the victim. Only the MAC address must be known for this, the device is not required to be discoverable. The victim is not required to take any action.
- Instead of completing the pairing, the attacker sends an LMP_start_encryption_req.
- Bluetooth crashes within the bignum_xormod on Nexus 5, your mileage might vary on other platforms.

CVE-2019-6994
Handler Escalation Over the Air: HCI via LMP

- Missing parameter check in a vendor specific LMP handler...
- **Crashes are the best case!**
- More reversing allows to **execute meaningful code**, but for each firmware version memory contents are different.
  (So far we did not find arbitrary code execution on **Nexus 5**.)
- On **Nexus 5** we are able to **execute test mode**, which normally needs to be enabled locally on the host.
- Many more vulnerable devices, such as **iPhone 5...6, Macbook 2012...2017, Raspberry Pi 3**.
- **CVE-2018-19860 / BT-B-g0ne**  

LMP input: \texttt{00 95 ...}

\textbf{LMP BPCS handler table}

\begin{itemize}
  \item \texttt{00 00} Features request
  \item \texttt{01 01} Features response
  \item \ldots
  \item \texttt{05 05} BFC accept
\end{itemize}

Next (unknown) handler table

\begin{itemize}
  \item \texttt{06 00} \ldots
  \item \ldots
\end{itemize}

HCI link control handler table

HCI link policy handler table

HCI host controller handler table

HCI info parameter handler table

HCI status parameter handler table

HCI test handler table

\rightarrow \texttt{95 03} Enable device under test mode

\rightarrow HCI vendor specific handler table

\rightarrow \texttt{BD 4E} Launch RAM (wrong parameters)
Demo

- Demo session this afternoon — visit me!
- See LMP and LCP monitor in action.
- See demos of the vulnerabilities.

https://github.com/seemoo-lab/internalblue
Do I have a device with a Broadcom chip?

- Platforms:
  - Android 6 and 7, Lineage OS 14.1
  - Linux/BlueZ (partially, some parts are work in progress)
  - (macOS in progress)

- Devices tested so far:
  - Nexus 5, Xperia Z3 Compact, Samsung Galaxy Note 3 (*BCM4339*, best support)
  - Nexus 6P, Samsung Galaxy S6, Samsung Galaxy S6 edge (also good support)
  - Macbook Pro 2011+2016 (with Ubuntu)
  - Raspberry Pi 3/3+
  - Thinkpad T420, T430
  - Asus USB Dongle
  - Any Linux PC with the *CYW20735* evaluation board

Devices with Broadcom chips supported by *InternalBlue*. [https://github.com/seemoo-lab/internalblue/tree/master/internalblue/fw/](https://github.com/seemoo-lab/internalblue/tree/master/internalblue/fw/)