Dissecting Broadcom Bluetooth

Dennis Mantz
Technische Universität Darmstadt
Security Analyst @ ERNW GmbH (Heidelberg)

Jiska Classen
Technische Universität Darmstadt
Secure Mobile Networking Lab - SEEMOO
Motivation
Reverse engineering Bluetooth firmware - why?!

- Dissecting firmware gives interesting insights on a **security** perspective.

- Modifying firmware allows to have a **full-featured working Bluetooth** implementation and then **adding your features**...

- Attach open source to a “**closed” source** project.

- Requires background in security, code analysis, wireless signals... Not many people can do it, but many require the results.

- We like reverse engineering and already had great experiences with similar projects (e.g.: **nexmon**).
## Terminology

<table>
<thead>
<tr>
<th>Host</th>
<th>Remote Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Controller Interface (HCI)</td>
<td></td>
</tr>
<tr>
<td>Device Mgr</td>
<td></td>
</tr>
<tr>
<td>Link Manager</td>
<td></td>
</tr>
<tr>
<td>Baseband Resource Manager</td>
<td></td>
</tr>
<tr>
<td>Link Controller</td>
<td></td>
</tr>
<tr>
<td>Bluetooth PHY</td>
<td></td>
</tr>
<tr>
<td>RFCOMM</td>
<td>SDP</td>
</tr>
<tr>
<td>L2CAP</td>
<td></td>
</tr>
</tbody>
</table>
Features
InternalBlue

Vendor specific HCI (local)

Modify firmware

LMP monitor & injection

Crash other Broadcom firmwares (CVE-2018-19860)

Fixed coordinate invalid curve attack test (CVE-2018-5383)

https://github.com/seemoo-lab/internalblue

Patching firmware

- Broadcom offers vendor specific HCI commands READ_RAM, WRITE_RAM, LAUNCH_RAM.
- .hcd-files shipped with the driver also use these commands to apply patches to RAM and ROM.
- ROM-patching is limited to a few slots, but that’s sufficient for branches into RAM.
- Neither .hcd-files nor vendor specific HCI commands require signatures, authentication, etc. Just insert your code :)

- Currently only assembly code, but we’re working on C support with NexMon.

NexMon. [https://github.com/seemoo-lab/nexmon](https://github.com/seemoo-lab/nexmon)
Reversing ...

- Okay... maybe not that simple. Where can we patch? What are we patching? Which functions are interesting?
- Almost no strings, no function names, no documentation except 2822 pages of Bluetooth 5.0 standard.
- Byte sequences in the standard help locating some functions.
- Many similarities between different firmware versions :)

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 Broadcom/Cypress!

- Reversing could have been faster: patch.elf shipped with development software contains **symbol table** for almost every firmware function...
LMP monitoring and injection

- **LMP:** Link Manager Protocol
- Located below HCI, cannot easily be sniffed as handling happens within firmware.

- Created assembly hooks to forward LMP via HCI. HCI is then forwarded by the recompiled Android Bluetooth stack (debugging features) via TCP. We automatically start a Wireshark **monitor**, which needs an LMP dissector plugin.

- Another hook allows to **inject** LMP frames if a connection exists, i.e. controlling test mode on a device under test if it was locally enabled:
  
  - sendlmp 57 54557575555555555555255

- Working assembly snippets currently only on Nexus 5 and partially on Nexus 6P.
We ❤ Bluetooth

Broken by design...
Discoverability

- If Bluetooth is on, **anyone can connect to a device** - no matter if it is discoverable.
- MAC addresses can be derived by sniffing with a software-defined radio.
- [Demo opening connections via known Bluetooth addresses]
Bluetooth 5.0 still offers “Just Works” pairing if a device claims to have **no input and no output**. IO capabilities are not authenticated.

“Just Works” pairing is not secure against MITM.

MITM can simply fake Niño and then attack “Just Works”.

Smartphones only show a **yes/no-question** instead of warning the user: *This might be insecure pairing!*

[Demo of other devices not showing a pin]
Testing other devices for known bugs


- [PoC zeroed y-coordinate in elliptic curve crypto]
  
  https://media.ccc.de/v/2018-154-internalblue-a-deep-dive-into-bluetooth-controller-firmware#t=1690

Details on this attack: http://www.cs.technion.ac.il/~biham/BT/
Try this at home! https://github.com/seemoo-lab/internalblue/blob/master/examples/CVE_2018_5383_Invalid_Curve_Attack_PoC.py
Finding Bugs

here it is →
Our own little bug...

- Just a missing “if” somewhere. They **silently patched** it in firmware version ~**summer 2014** but never shipped .hcd-patches for older firmwares. Long development cycles mean those devices are still around.

- Incomplete list of vulnerable devices:
  - Nexus 5
  - iPhone 5, 5s, 6
  - Xperia Z3, Z5
  - Raspberry Pi 3
  - Samsung Galaxy Note 3

- CVE-2018-19860 / **BT-B-g0ne**
  [Demo of remote crash]

  “**does not exist**”

  “**not standard compliant**”

  “**does not affect WiFi performance**”
...little bugs grow up so fast!

- Missing parameter check...
- **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.

- CVE-2018-19860 / **BT-B-g0ne**
  [Demo of remote device under test / jamming]
Test mode execution

- Master (attacker) and remote device exchange test packets.
- Master can **disable adaptive frequency hopping** (AFH) on target device but not change its own...
- No matter if AFH is disabled or not, one can see both devices hopping on all channels during test mode.
- Works on **Nexus 5 and Xperia Z3** (BCM4339).
Bug finding toolchain

- Adding **tracepoints** with InternalBlue - only execute once, dump registers, stack and heap, example here is for LMP dispatcher in Nexus 5:
  \[ \text{tp add 0x3f3f4} \]

- **Emulation** with Unicorn/radare2 which generates **function call sequences** and **memory diffs**. Currently only running for one function call.
- Emulation with qemu/gdb for sequences of incoming frames (work in progress).

- Whatever, it generates tons of hexadecimal stuff on that you can stare for hours.

Unicorn/radare2 emulation is a modified setup from Hugo (got it after Fitbit talk at 34C3) and Matthias Hanreich (who extended the emulator to a Fitbit fuzzer).
Fixing Bugs

It's dead, Jim!
Bluetooth firewall

- **Actual fix**: Fix vulnerable handler. We have a .hcd-patch ready for Nexus 5. Releasing that fix would tell you which handler is vulnerable. Patch size is **14 bytes**...

- **Generic fix**: Apply generic **filters**, because invisible devices will reply to pings, connection establishments, etc.

  We wanted to release these filters for 35C3, but they crash Bluetooth of some connecting devices. More recent devices. Ooops...
How long will the old bug be around?

- **Vendor fix**: vendors need to provide updated .hcd-files with their operating system updates.
- Some devices are **too old** to get vendor updates...
- Vendor updates will **leak the vulnerability**.

**Turn off Bluetooth if your device has a Broadcom chipset and was introduced to the market before 2017.**

- If you have a very old chip you are not vulnerable: iPhone 4, 4s, Thinkpad T420, iMac 2009...
Q&A

https://github.com/seemoo-lab/internalblue