Extracting the painful (blue)tooth

Matteo Beccaro
• Security Consultant at Secure Network
• Technical Research Leader at Opposing Force, Physical Security division of Secure Network
• @bugardy

Matteo Collura
• Student at Politecnico di Torino (www.polito.it)
• Electronic Engineer
• Researcher in several fields
• @eagle1753
Who we are…

Matteo Beccaro
- Security Consultant at Secure Network
- Technical Research Leader at Opposing Force, Physical Security division of Secure Network
- @bughardy
Who are we…

Matteo Collura
• Student at Politecnico di Torino
• Electronic Engineer
• Researcher in different fields concerning security (NFC, bluetooth)
• Now focusing on social skills (NLP, social engineering..)
• @eagle1753
1. Index
2. What the hell is bluetooth?
3. Known and unknown risks…
   1. …<risk1>
   2. …<risk2>
   3. …<our vuln>
4. Demo Time!
5. Future works…
What the hell is Bluetooth?

- Wireless standard for exchanging data over short distances.
- Short wavelength UHF: 2.4 – 2.485 GHz
- 79 channels (usually) + Adaptive Frequency Hopping
- Name coming from Harald Bluetooth

\[\begin{align*}
\star &+ \triangleleft = \mathbb{BL}T\text{O}\thinspace \text{TH}
\end{align*}\]

- Scandinavian humor... 😊
What the hell is Bluetooth?

Layer protocol architecture

Core protocols

Cable replacement protocols

Telephony control protocols

Adopted protocols

So many different stacks!

LMP, L2CAP, SDP are mandatory!
What the hell is Bluetooth?

• So many updates!

Version 1:
• 1.0: Mandatory BD_ADDR
• 1.2: Adaptive frequency-hopping spread spectrum → resistance to interferences and eavesdropping (theoretically 😊)

Version 2:
• 2.0: EDR (optional) for faster data transfer, GFSK+PSK modulation
• 2.1: Secure Simple Pairing, Extended Inquiry Response
What the hell is Bluetooth?

- So many updates!

**Version 3:**
- 3.0: Alternative MAC/PHYs for high data transfer, Unicast Connectionless Data

**Version 4:**
- 4.0: Includes now Bluetooth Low Energy protocol (or Smart)
- 4.1: Limited discovery time, lower consumptions, LE link layer topology
- 4.2: LE Data packet extension, LE «secure» connections, Link Layer privacy (really?)
1. Index
2. What the hell is bluetooth?
3. Known and unknown risks...
   1. ...<BlueSnarf>
   2. ...<BlueBug>
   3. ...<BlueChop>
   4. ...<our vuln>
4. Demo Time!
5. Future works...
Known and unknown risks...

- BlueSnarf, by Holtmann & Laurie

  When? → Late 2003
  What? → Bluetooth implementation on mobile phones and pocket palms
  Why? → «(in)security» of OBEX protocol

  Easy GET requests to common files (calendar, contacts..) → No authentication needed
  No prompts on the user’s side → No authentication needed
Known and unknown risks...

• BlueBug, by Adam Laurie & Martin Herfurt

When? 2004 @DEFCON12

What? Bluetooth implementation on mobile phones, especially Symbian OS

Why? Security loophole

- No secure auth prior to v2.0
- Control device through plain serial connection
- Download items via OBEX protocol w/out prompts
Known and unknown risks...

- Legacy (prior to v2.0) pairing procedure

Encryption Algorithm

Reflected to Device B
Known and unknown risks...

- Legacy (prior to v2.0) authentication procedure

![Diagram showing authentication流程](https://via.placeholder.com/150)

- Previously evaluated Link Key
- Referred to Device B
- Encryption Algorithm
- SRES = E(randA, LK, BD_ADDR)
- Check SRES = SRES'
- Device A Verifier
- Device B Claimant
1. Known and unknown risks...

- Secure simple pairing

1a. Generate a random number Na.
2a. Set \( r_A = r_B = 0 \)
3a. Compute and check \( C_b \)
4a. Compute \( V_a = g(P_Ka, P_Kb, Na, Nb) \)

1b. Generate a random number Nb.
2b. Set \( r_A = r_B = 0 \)
3b. Compute \( C_b = f(P_Kb, P_Ka, Nb, 0) \)
4b. Compute \( V_b = g(P_Ka, P_Kb, Na, Nb) \)

Proceed if users confirm \( V_a = V_b \)
Known and unknown risks...

- **BlueChop, following BlueSnarf**

  **What?**
  It disrupts any bluetooth piconet from the outside

  **Provided**
  Master must support multiple connections

  **Spoof a random slave out of the piconet**

  **Contact the master**

  **Confusion of the master’s internal state**

  **Piconet disruption 😊**
• Bluetooth LE encryption bypass, by Mark Ryan:
  – Eavesdropping vs Decrypting
  – 3 different keys needed to establish a connection, TK, STK, LTK
  – If we are able to save the key exchange procedure, we are done 😊

Pairing

TK

STK

TK

LTK

• What if I get TK?
Known and unknown risks...

• TK, 128 bit AES key, depends on the pairing mode:

  - Just Works
    - TK = 0
  - 6-digit PIN
    - TK = 128-bit number
  - Out Of Band (OOB)
    - TK = #fuckyourself

• Bruteforce is the way. Intel i7, just one core → less than 1 sec

  - TK ➔ STK
  - STK ➔ LTK
  - LTK ➔ 42

• The whole procedure may be computed offline
Officially introduced with Android 5.0 it enables to unlock the smartphone without user interaction if at least one of the following conditions apply:

1. The smartphone is in range of a previous saved NFC tag.
2. The smartphone recognize the face of the owner, which must be previously saved.
3. The smartphone is in contact with a body.
4. The smartphone is within a certain location.
5. A previous enabled bluetooth device is connected to the smartphone.
Bluetooth Unlock

This may be the most interesting and most used function of all the above. The user set a paired bluetooth device as Trusted, and from now on every time that device is linked to the smartphone the lockscreen is bypassed.

Good, so what is the problem?
In Android < 5.1 the LK (LinkKey) is not checked to verify the Bluetooth device.
Bluetooth Unlock…
Now the question is:

How to get the 4 bytes of the MAC address required?

Two possible solutions:

- **Bruteforce**
  - Slow
  - Expensive
  - Not such a good idea

- **Sniffing**
  - Requires vicinity
  - Target can become aware
  - Authentication process is required
Bruteforce...

- **Slow**
  
  We cannot bruteforce the MAC address offline, we need to try a new connection everytime.

- **Expensive**
  
  We can speed it up parallelizing it but costs increase.

- **Not such a good idea**
  
  42 bits will definitely require too much time.
Sniffing…

- Requires vicinity
  - Target must be near enough for our ubertooth to intercept packets

- Target can become aware
  - Target can be suspicious of strange guy with big antenna(s)

- Auth process is required
  - Usually only 3 bytes of MAC address are transmitted
Hybrid is always the solution

- Android automatically sends out ‘beacons’ of paired BT devices.
- The trusted device **must** be a paired device.
- We can intercept beacons to retrieve 3 bytes of the MAC address.
- Brute-force the remaining… 1 byte = 256 possible MAC addresses.
Demo Time!

<video demo>
Android 5.1 adds a new nice feature...
Demo Time!

<video demo>
New findings...

Is it fixed?

It depends…

Android >= 5.1
SmartUnlock is fixed
API are still vulnerable

Android <= 5.0.X
SmartUnlock is not fixed
API are vulnerable
API does not have a safe method to check if a device is connected with a proper LK.

Android Security Team told us that there is a method for this, but it was not yet in SDK, as 27th April, 2015. And it still not present.
Why fixing the API is important if SmartUnlock function is fixed?

3rd party applications!

Demo time!
1. Index
2. What the hell is bluetooth?
3. Known and unknown risks...
   1. ...<risk1>
   2. ...<risk2>
   3. ...<our vuln>
      1. Demo Time!
4. Future works...
Bluetooth is everywhere, we are focusing on:

- IoT Devices
- Smart Locks
- Fit Band
- etc
Thank you
Q&A Time...