THE PENTEST EXPERTS.

IT SECURITY KNOW-HOW

Gerhard Klostermeier and Matthias Deeg

Case Study: Security of Modern Bluetooth Keyboards

SySS IT Security Research Project

June 2018

Schaffhausenstraße 77, 72072 Tübingen, Germany

+49 (0)7071 - 40 78 56-0

info@syss.de

www.syss.de

1 Project Objectives

In the course of this research project, SySS GmbH analyzed three currently popular wireless keyboards using Bluetooth technology that can be bought on the Amazon marketplace for security vulnerabilities.

The following three devices were tested for security issues from different attacker perspectives.

1. 1byone Keyboard
2. Logitech K480
3. Microsoft Designer Bluetooth Desktop, Model 1678, 2017

Besides security issues concerning these three specific Bluetooth keyboards and their implementations, SySS GmbH also searched for security vulnerabilities within the Bluetooth specifications [1] regarding used Bluetooth technologies like Bluetooth Classic (BR/EDR), for example Bluetooth 3.0, and Bluetooth Low Energy (BLE) as well as in Bluetooth stack implementations of current operating systems.

2 Summary

During this research project with a total duration of 15 person-days, SySS GmbH could identify some security issues concerning the three tested Bluetooth keyboards.

The secret pairing information stored on the keyboards can be easily extracted by an attacker with physical access. The credentials in this information can be used to conduct further attacks on the host.

The 1byone keyboard does not require authentication when pairing to a Windows 10 host and the communication of the Microsoft Designer Bluetooth keyboard can be decrypted if an attacker passively eavesdrops on the pairing process.

Furthermore, by continuously sending pairing requests to some operating systems, an attacker can prevent other devices from pairing (denial-of-service).

General recommendations on how to improve the security when using Bluetooth devices can be found in Section 5.

All found security issues are listed in the following table.

Risk level	Finding	Recommendation	Reference
M1.1	**All keyboards – Insufficient protection of sensitive data**: It was possible to extract the cryptographic keys for the Bluetooth RF communication of all keyboards with direct hardware access	Protect the cryptographic keys on the hardware against unauthorized read access, for example by using memory chips with enabled read-back protection features	4.1 Page 8
M1.2	**Hosts – Changes in device capabilities**: Some hosts (Android, iOS, Mac OS X) do not care if an already paired device changes its capabilities	Hosts should ignore capabilities of devices which were not there during the initial pairing process	4.2 Page 14
M1.3	**Microsoft Designer Bluetooth Desktop – Pairing security**: An attacker can passively eavesdrop on the initial pairing process and extract the Long Term Key (LTK) for decrypting further Bluetooth communication	Do not use Bluetooth Low Energy devices prior to version 4.2 and only pair Bluetooth devices in a secure environment	4.6 Page 16
L1.1	**All keyboards – Simulating host**: An attacker can simulate the host of a victim to prevent the keyboard from connecting correctly (denial-of-service)	Monitor the wireless environment to detect suspicious Bluetooth hosts	4.4 Page 16
L1.2	**Windows 10 Host – Pairing request**: An attacker can continuously send pairing requests to a Windows 10 host which makes it impossible to pair other devices (denial-of-service)	Configure the host to be non-pairable by default	4.5 Page 16
L1.3	**1byone – Authentication**: The keyboard does no offer an authentication method for pairing by itself	Force keyboard to use authentication; do not use devices which by default do not use any authentication	4.7 Page 17

Remarks:
Security flaws are highlighted in color. SySS GmbH defines the risks as follows:

High risks	Examination or manipulation of data which should not be seen or manipulated. The intrinsic value is not considered here.
Medium risks	Security gaps which only cause a security issue together with other, including human components.
Low risks	Security flaws which cannot cause any changes by non-authenticated third parties, e.g. encryption methods retraceable solely under laboratory conditions or support for debugging options (for example the HTTP methods TRACE and TRACK).
Information Leaks	Are shown in the field "Info" and contain, for instance, information on utilized software. Although information leaks do not represent a direct risk, they provide potential attackers with too detailed data.
Anomalies	This is the term for unusual configurations (e.g. a DNS service on UDP port 3000). This behavior does not describe a security flaw.
Risk	Security vulnerabilities rectified during the term of the project or since the last test are not marked in color, but are listed due to reasons of completeness.
Not checkable	Due to missing prerequisites (e.g. other corrected findings), the security weakness could no longer be checked. SySS GmbH can neither confirm the weakness nor its rectification.

3 Test Information

This research project concerning current wireless keyboards using Bluetooth technology had a total duration of 15 person-days. The research project started on February 26, 2018, and ended on April 27, 2018.

All research and test activities were performed by the two SySS IT Security Consultants Gerhard Klostermeier and Matthias Deeg.

3.1 Test target

The primary test target of this research project were the following three popular wireless keyboards using Bluetooth technology of three different manufactures.

1. 1byone keyboard[1]
2. Logitech K480[2]
3. Microsoft Designer Bluetooth Desktop[3], Model 1678, 2017

These three devices were analyzed for security vulnerabilities from different attacker perspectives, for example as an attacker without direct physical access to the hardware devices using wireless radio communication, as an attacker with direct physical access, and as an attacker with both capabilities.

1 https://www.amazon.de/1byone-ODE00-0713-Tastatur-Bluetooth-kabellos/dp/B0131YNTWW/
2 https://www.amazon.de/Logitech-kabellose-Bluetooth-Tastatur-Computer-Smartphone/dp/B00MWNDUDM/
3 https://www.amazon.de/Microsoft-Designer-Bluetooth-deutsches-Tastaturlayout/dp/B00YIO7LF6/

Furthermore, SySS GmbH also searched for security vulnerabilities within the Bluetooth specifications regarding used Bluetooth technologies like Bluetooth Classic, for example Bluetooth 3.0, and Bluetooth Low Energy (BLE) as well as in Bluetooth stack implementations of current operating systems.

Figures 1, 2, and 3 show the three tested Bluetooth keyboards.

Figure 1: Tested 1byone wireless keyboard with Bluetooth 3.0 technology

Figure 2: Tested wireless keyboard Logitech K480 with Bluetooth 3.0 technology

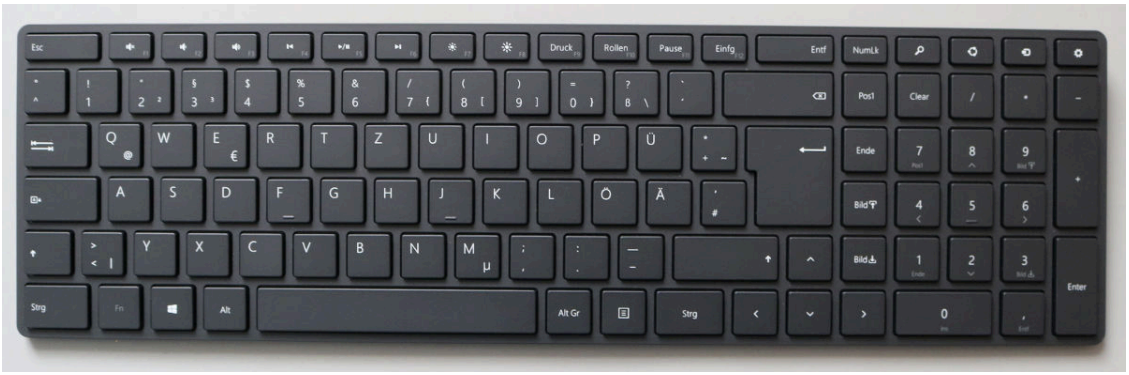

Figure 3: Tested wireless keyboard Microsoft Designer Bluetooth Desktop with Bluetooth LE technology

The three tested Bluetooth keyboards were tested in combination with host systems using the following operating systems:

- Arch Linux
- Microsoft Windows 10
- Google Android 8
- Apple iOS 11.2.6 and 11.3
- Apple Mac OS X 10.13.4

3.2 Test methodology

For the security analysis of the three Bluetooth keyboards, we used the following general three-step approach:

1. Hardware analysis
2. Firmware analysis
3. Radio-based analysis

In the course of the hardware analysis, we opened the keyboards, had a closer look at the printed circuit boards (PCBs) and identified interesting chips that are important for the functionality of the devices, i.e. Bluetooth transceivers and flash memory chips. After reading the specification of the identified chips, we tried to access the firmware and device configuration of the Bluetooth devices.

Concerning the firmware analysis, we only had a superficial look at the extracted firmware and the device configuration of the tested devices due to the limited time available for this research project.

Regarding the radio-based analysis of the Bluetooth devices, we used Bluetooth USB dongles with the two different chipsets CSR8510 and BCM20702A0 to communicate with the Bluetooth keyboards and to analyze the Bluetooth communication using sniffing capabilities of the Linux Bluetooth stack BlueZ on the host controller interface (HCI). Additionally, we also used the open source 2.4 GHz wireless development platform Ubertooth One[4] by Great Scott Gadgets for analyzing the Bluetooth radio communication.

During the research project, we also developed a software tool based on available software projects for emulating Bluetooth keyboards in order to perform specific tests and actual attacks. The source code of this software tool is available on GitHub [2].

4 Test Results

This section covers all security-related findings discovered during the research on the three Bluetooth keyboards. Although the keyboards were the main focus of this analysis, the findings include security issues of Bluetooth stacks of common operating systems as well.

4.1 Extraction of cryptographic key material

During the security test, SySS GmbH analyzed the hardware security of the three tested wireless keyboards and checked what kind of attacks an attacker with direct physical access to such a device could perform. The most interesting question was how sensitive data like cryptographic keys for secure Bluetooth RF communication or the device firmware itself was stored on the devices. For this, SySS GmbH opened up all three wireless keyboards and analyzed their hardware design and configuration.

Figure 4 shows the PCB of the tested 1byone wireless keyboard with a BCM20730 Bluetooth transceiver and an EEPROM chip.

4 https://greatscottgadgets.com/ubertoothone/

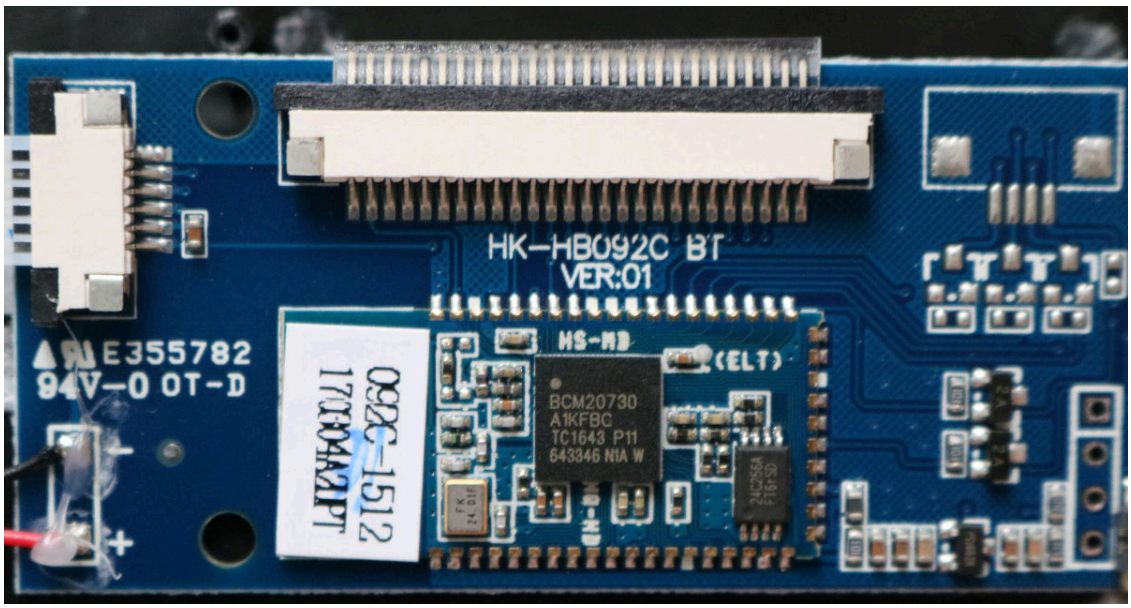

Figure 4: PCB of the tested 1byone wireless keyboard with BCM20730 Bluetooth transceiver and EEPROM chip

Figure 5 shows the PCB of the tested wireless keyboard Logitech K480 which had a similar hardware setup to the 1byone Bluetooth keyboard with a CYW20730[5] Bluetooth transceiver and an EEPROM chip. Actually, the BCM20730 and the CYW20730 are the same transceiver that is now produced by Cypress and was formerly produced by Broadcom.

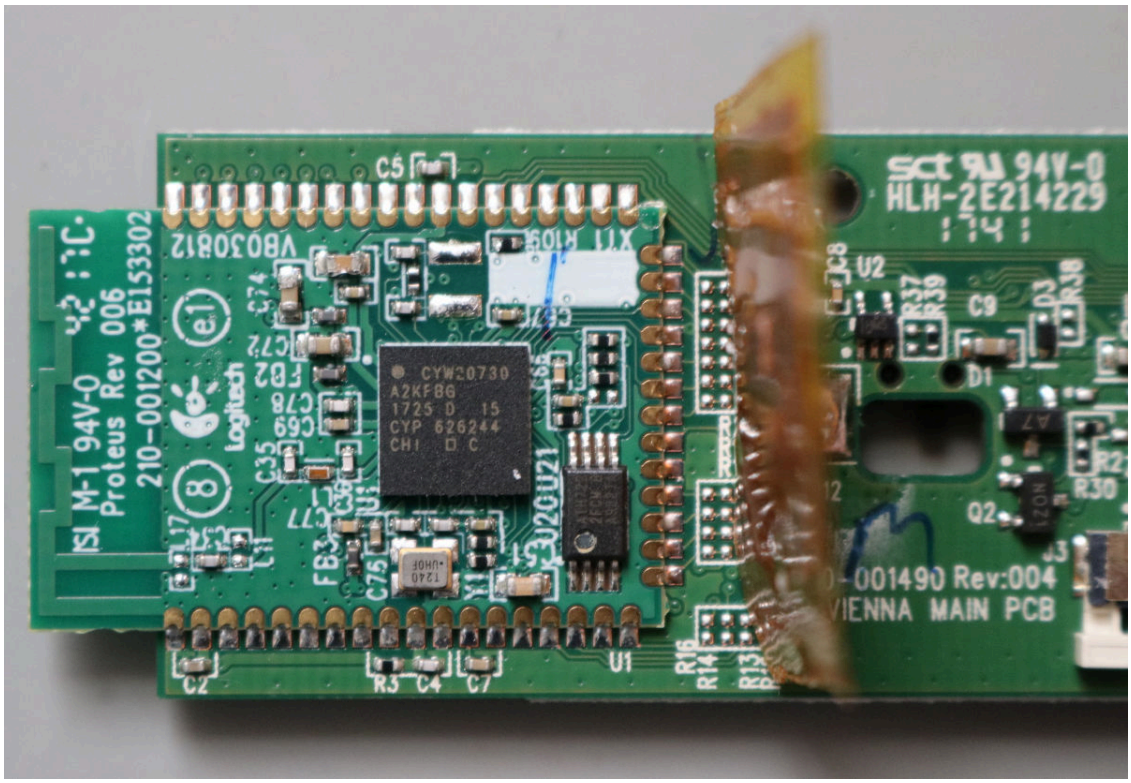

Figure 5: PCB of the tested wireless keyboard Logitech K480 with CYW20730 Bluetooth transceiver and EEPROM chip

5 http://www.cypress.com/file/298211/download

Figure 6 shows the PCB of the tested Microsoft wireless keyboard with a nRF51822[6] system-on-a-chip (SoC) by Nordic Semiconductor.

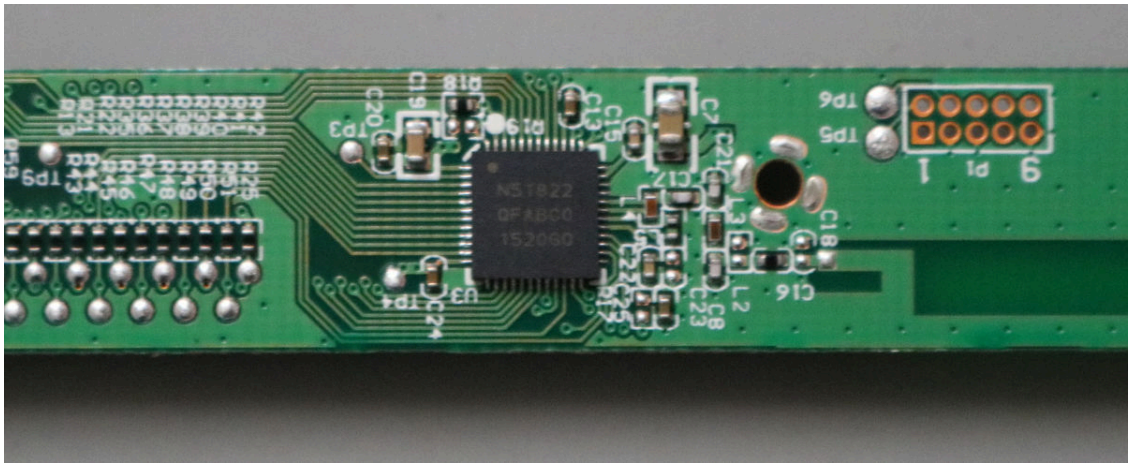

Figure 6: PCB of the tested Microsoft wireless keyboard with nRF51822 SoC

The 1byone and the Logitech K480 keyboard used Bluetooth 3.0 technology (Bluetooth classic) provided by the BCM20730/CYW20730 transceiver, the Microsoft Designer Bluetooth Desktop keyboard used Bluetooth Low Energy (BLE) technology provided by the nRF51822 SoC. M1.1

Concerning the 1byone and the Logitech K480 keyboard which used an BCM20730/CYW20730 transceiver in combination with an EEPROM chip, SySS GmbH found out that sensitive device data like the Bluetooth link key for secure Bluetooth RF communication was stored as plain text on the EEPROM chip. As these kind of memory chips do not offer any protection features against an attacker with direct physical access, SySS GmbH could simply dump the complete content of the EEPROM chip of both, the 1byone and the Logitech K480 keyboard.

Figure 7 exemplarily illustrates a part of the dumped EEPROM content of a paired Logitech K480 Bluetooth keyboard.

6 https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF51822

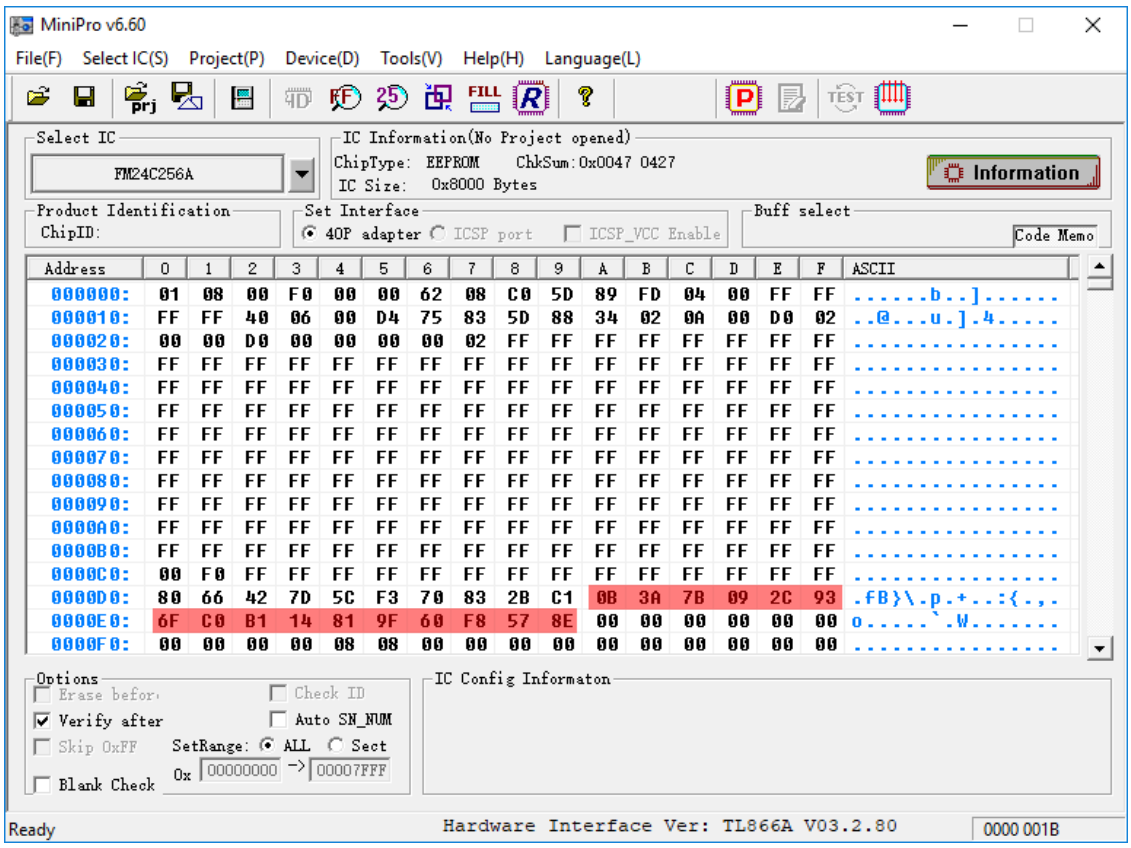

Figure 7: Memory dump of the EEPROM chip containing the Bluetooth link key of the paired device

The following output shows the content of the BlueZ configuration file that was generated on a Linux host after successfully pairing the Logitech K480 wireless keyboard.

```
[General]
Name=Keyboard K480
Class=0x002540
SupportedTechnologies=BR/EDR;
Trusted=false
Blocked=false
Services=00001000-0000-1000-8000-00805f9b34fb;00001124-0000-1000-8000-00805f9b34fb;
    ;00001200-0000-1000-8000-00805f9b34fb;

[LinkKey]
Key=8E57F8609F8114B1C06F932C097B3A0B
Type=5
PINLength=0

[DeviceID]
Source=2
Vendor=1133
Product=45885
Version=10243
```

Thus, the link key (8E57F8609F8114B1C06F932C097B3A0B) for secure Bluetooth RF communication of the tested Logitech K480 keyboard was stored as plain text at the offset 0xDA of the EEPROM memory dump in reversed byte order.

The following output shows the content the BlueZ configuration file that was generated on a Linux host after successfully pairing the 1byone wireless keyboard.

```
[General]
Name=1byone Keyboard
Class=0x000540
SupportedTechnologies=BR/EDR;
Trusted=false
Blocked=false
Services=00001000-0000-1000-8000-00805f9b34fb;00001124-0000-1000-8000-00805f9b34fb;
    ;00001200-0000-1000-8000-00805f9b34fb;

[DeviceID]
Source=2
Vendor=1256
Product=28705
Version=283

[LinkKey]
Key=65EAA8D540AB82C9B9152886B156411B
Type=4
PINLength=0
```

As the following excerpt of the EEPROM memory dump of the 1byone keyboard shows, the link key (65EAA8D540AB82C9B9152886B156411B) of this Bluetooth device could be found as plain text at the offset 0xCA in reversed byte order.

```
00000000  01 08 00 f0 00 00 62 08  c0 5d 89 fd 04 00 ff ff  |......b..]......|
00000010  ff ff 40 06 00 1e 25 25  00 73 20 02 0a 00 c0 02  |..@...%%.s .....|
00000020  00 00 c0 00 00 00 00 02  00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000000c0  80 20 42 b7 5c f3 70 83  2b c1 1b 41 56 b1 86 28  |. B.\.p.+..AV..(|
000000d0  15 b9 c9 82 ab 40 d5 a8  ea 65 00 00 00 00 00 00  |.....@...e......|
000000e0  00 00 00 08 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

Regarding the tested Microsoft Bluetooth keyboard, which did not have an external EEPROM chip, SySS GmbH was also able to extract cryptographic key material in form of the used Long Term Key (LTK) of this paired Bluetooth Low Energy device.

For this, SySS GmbH used a Segger J-Link Pro Debug Probe[7] that was connected to available debug pins on the PCB of the Microsoft Bluetooth Designer Desktop keyboard, as shown in Figure 8.

7 https://www.segger.com/products/debug-probes/j-link/

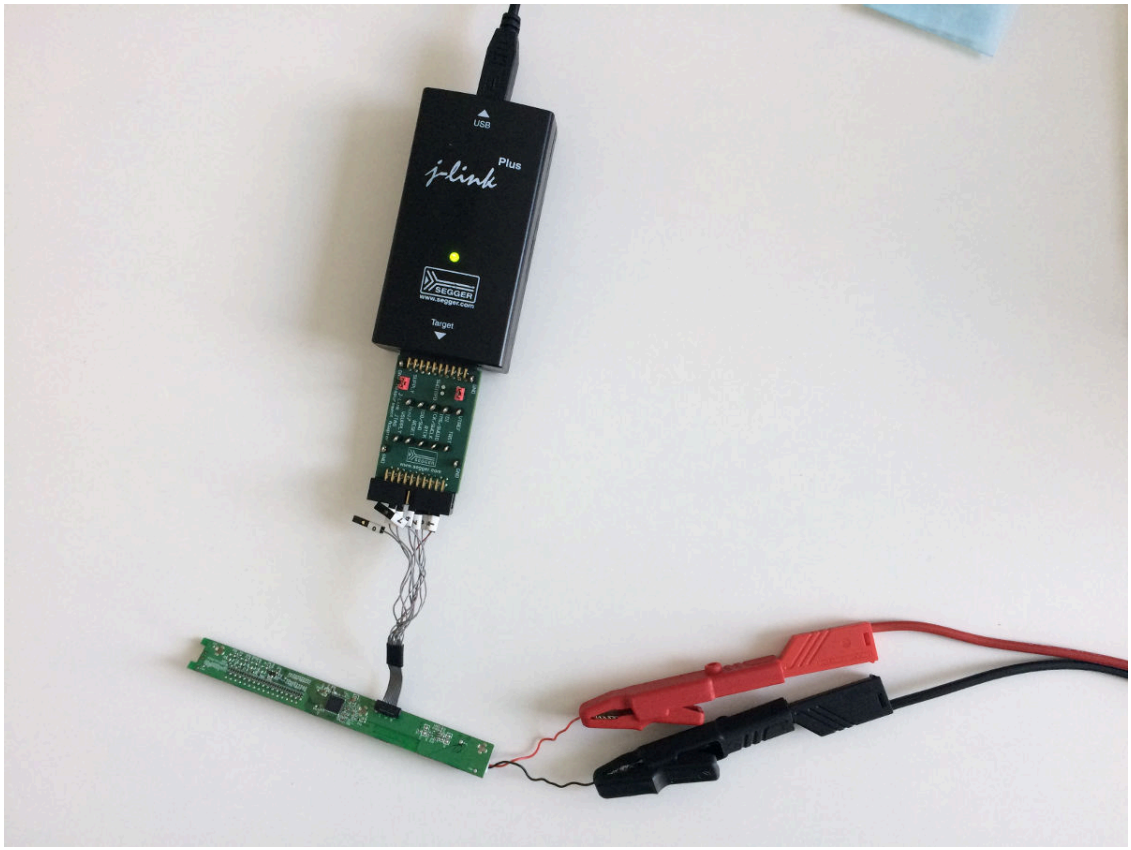

Figure 8: SWD (Serial Wire Debug) connection to Microsoft keyboard using Segger J-Link Pro

With this debug connection, for instance, it was possible to read the memory content of the nRF51288 SoC, as the following output illustrates.

```
SEGGER J-Link Commander V6.20i (Compiled Nov 17 2017 17:47:07)
DLL version V6.20i, compiled Nov 17 2017 17:46:22

Connecting to J-Link via USB...O.K.
Firmware: J-Link V10 compiled Nov 28 2017 11:45:53
Hardware version: V10.10
S/N: 600104371
License(s): RDI, FlashBP, FlashDL, JFlash, GDB
VTref = 3.327V


Type "connect" to establish a target connection, '?' for help
J-Link>connect
Please specify device / core. <Default>: NRF51822_XXAA
Type '?' for selection dialog
Device>device NRF51822_XXAB
Please specify target interface:
  J) JTAG (Default)
  S) SWD
  F) FINE
  I) ICSP
  C) C2
TIF>S
```

```
Specify target interface speed [kHz]. <Default>: 4000 kHz
Speed>
Device "NRF51822_XXAB" selected.


Connecting to target via SWD
Found SW-DP with ID 0x0BB11477
Scanning AP map to find all available APs
AP[1]: Stopped AP scan as end of AP map has been reached
AP[0]: AHB-AP (IDR: 0x04770021)
Iterating through AP map to find AHB-AP to use
AP[0]: Core found
AP[0]: AHB-AP ROM base: 0xF0000000
CPUID register: 0x410CC200. Implementer code: 0x41 (ARM)
Found Cortex-M0 r0p0, Little endian.
FPUnit: 4 code (BP) slots and 0 literal slots
CoreSight components:
ROMTbl[0] @ F0000000
ROMTbl[0][0]: E00FF000, CID: B105100D, PID: 000BB471 ROM Table
ROMTbl[1] @ E00FF000
ROMTbl[1][0]: E000E000, CID: B105E00D, PID: 000BB008 SCS
ROMTbl[1][1]: E0001000, CID: B105E00D, PID: 000BB00A DWT
ROMTbl[1][2]: E0002000, CID: B105E00D, PID: 000BB00B FPB
ROMTbl[0][1]: F0002000, CID: 00000000, PID: 00000000 ???
Cortex-M0 identified.
J-Link>savebin C:\Users\syss\Documents\nrf51_code.dump 0 0x20000
Opening binary file for writing... [C:\Users\syss\Documents\nrf51_code.dump]
Reading 131072 bytes from addr 0x00000000 into file...O.K.
J-Link>
```

The following output shows the content the BlueZ configuration file that was generated on a Linux host after successfully pairing the Microsoft Bluetooth Designer Desktop keyboard.

```
[General]
Name=Designer Keyboard
Appearance=0x03c1
AddressType=static
SupportedTechnologies=LE;
Trusted=false
Blocked=false
Services=00001800-0000-1000-8000-00805f9b34fb;00001801-0000-1000-8000-00805f9b34fb;
    ;0000180a-0000-1000-8000-00805f9b34fb;0000180f-0000-1000-8000-00805f9b34fb;
    ;00001812-0000-1000-8000-00805f9b34fb;

[ConnectionParameters]
MinInterval=12
MaxInterval=12
Latency=30
Timeout=300

[DeviceID]
Source=2
Vendor=1118
Product=2054
Version=277
```

```
21
22 [IdentityResolvingKey]
23 Key=279E6109528494AF50D5BB1F84BB3F7E
24
25 [LocalSignatureKey]
26 Key=EA4F01EDDB611C8B6B9062FF30B96403
27 Counter=0
28 Authenticated=false
29
30 [LongTermKey]
31 Key=7D051EC826CFF51F80D9CD023C87A402
32 Authenticated=0
33 EncSize=16
34 EDiv=34510
35 Rand=14289962735818054629
```

Figure 9 shows the part of the memory dump at offset 0x1F410 that contained the used Bluetooth Long Term Key (7D051EC826CFF51F80D9CD023C87A402) of the paired Microsoft keyboard.

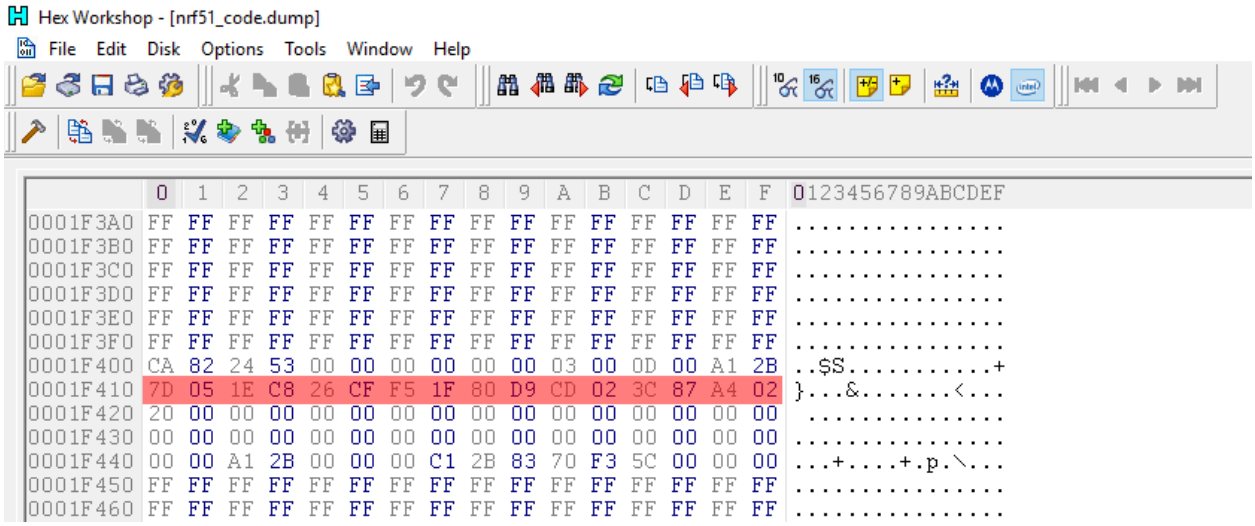

Figure 9: Memory dump of the tested Microsoft keyboard with stored Bluetooth Long Term Key (LTK)

The nRF51822 by Nordic Semiconductor has a read-back protection feature that can be enabled during programming of the chip in order to prevent unauthorized access to SoC's internal memory. However, Microsoft did not enable the read-back protection during the manufacturing process of the tested Microsoft keyboard which enables an attacker with direct physical access to read and write the content of the nRF51822 memory using the provided debug port as described above.

With access to cryptographic key material like the Bluetooth link key (Bluetooth Classic) or the Bluetooth Long Term Key (Bluetooth Low Energy), an attacker can perform further attacks (like sniffing attacks) against the encrypted Bluetooth RF communication between a Bluetooth keyboard and a host system, for example a PC or a smart phone. Furthermore, an attacker can also simply use extracted cryptographic keys to establish a secure Bluetooth connection to corresponding paired host systems by spoofing a Bluetooth device, for instance a human interface device (HID), in order to perform malicious actions like executing code.

SySS GmbH recommends better protecting the cryptographic keys on the wireless keyboards against unauthorized read access, for example by using memory chips with enabled read-back protection features that do not allow an attacker to simply dump memory contents by having a proper physical connection to the memory chip.

4.2 Changes in device capabilities

M1.2

SySS GmbH noticed that most Bluetooth stacks do not care if paired devices change their attributes. For example, a device can change its name, vendor ID, product ID and serial, and the host will go along with this as long as the Bluetooth address and the link key stay the same.

In some implementations, this can be taken to extremes by also changing the capabilities and behavior of a device in a fundamental way. Knowing this, SySS GmbH designed an attack vector which then was validated during this research using the developed Bluetooth keyboard emulator [2].

The following steps outline the attack scenario:

1. A victim buys Bluetooth headphones and pairs them to his computer or smartphone.
2. The attacker gets hold of the headphones (e.g. the victim loses, disposes, or sells them, or they get stolen by the attacker).
3. The attacker extracts the pairing information from the headphones (Bluetooth address and link key).
4. The attacker uses the extracted pairing information to establish a valid connection to the victims computer or smartphone with an emulated device.
5. Depending on the Bluetooth stack of the victim, the emulated device can behave as something completely different – for example as Bluetooth keyboard instead of headphones. This enables the attacker to gain access to the host system.

SySS GmbH demonstrated this attack vector not only by using the two tested Bluetooth Classic keyboards (1byone and Logitech K480), but also by using a different device class in form of Pioneer SE-MJ553BT-K headphones which were previously paired to an Android, OS X, or iOS device. The host systems accepted and established a connection with the emulated Bluetooth keyboard, which was using the extracted cryptographic link key. Although the originally paired device were headphones, the host systems were not concerned with the fact that the emulated Bluetooth device using the known link key was suddenly a keyboard.

This attack vector did not work against test systems with Windows 10 and with a modern Linux distribution (Arch Linux). However, the cause for the failed attack concerning these two operating systems is still unclear and a topic for further research. It might be the case that the Bluetooth stacks of the Windows 10 and the Arch Linux operating system do not allow such fundamental changes in the device behavior or it might just be due to technical issues and missing features in our developed Bluetooth keyboard emulator.

However, regardless of the used operating system and the corresponding Bluetooth stack, some general recommendations can be derived which all come down to user awareness. Each Bluetooth device – host or peripheral – contains secret cryptographic key material of paired devices. Losing a device is somehow equivalent to losing a key. If a device is lost or is disposed, the corresponding cryptographic key material on the other paired device should be deleted, too. On a Windows 10 host, for instance, this can be achieved by using the “remove device” button in the Bluetooth settings.

4.3 BlueBorne

BlueBorne is a collection of several security issues in Bluetooth host stacks discovered by Armis Inc. that were published in September 2017. The vulnerabilities include logical flaws, memory leakage, and even remote code execution. As part of this research, SySS GmbH evaluated the attack vectors in order to assess their practicability and the impact.

Armis released details for all exploits in their technical white papers which can be obtained from [3]. These documents have all necessary information to implement all of the attacks. However, some proof-of-concept exploits are already published by Armis on GitHub [4]. Therefore, SySS GmbH considers these attacks as very practicable in real-world attack scenarios.

The impact of these attacks are highly related to the patch and update management of the affected Bluetooth stack. There are updates to patch the vulnerabilities, but not all affected devices are updatable in the same manner. For example, Armis demonstrated a remote code execution for Android-based devices. This security issue has been fixed, but not all manufacturers of Android devices are implementing and rolling out this patch in their updates.

On the other hand, it is not always that easy to successfully perform the attacks. Since it is most likely that, for example, Android versions of different manufacturers for different devices are slightly different, an exploit that worked well on one Android device with a vulnerable Android version might not simply work on another device.

In the course of the research project, SySS GmbH adapted the BlueBorne Android memory leak exploit (CVE-2017-0785) and was able to successfully gather data from an Android smartphone with the Android version 6.0.1, but not from an Android tablet with the Android version 5.1.1.

The following output exemplarily illustrates the successful execution of this memory leak exploit.

```
sudo python2 infoleak.py hci0 00:A0:C6:02:E5:95 0 1 1000
[*] Connecting to 00:A0:C6:02:E5:95
[*] Sending L2CAP_UUID request
[*] Receiving L2CAP_UUID response
[+] Exploit: Done
00000000  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  |...|....|....|....|
*
00000050  00 00 00 00  01 00 02 00  00 07 52 b8  41 6b 56 a2  |....|....|..R.|AkV.|
00000060  18 df 52 b8  60 4f 5d a2  79 09 55 a2  80 61 6b a2  |..R.|`O].|y.U.|.ak.|
00000070  00 00 00 00  d0 47 6b a2  3d 9a 54 a2  c3 6c 56 a2  |....|.Gk.|=.T.|.lV.|
00000080  30 75 00 00  00 00 00 00  80 61 6b a2  9c 7d 6c a2  |0u..|....|.ak.|.}l.|
00000090  80 61 6b a2  9c 7d 6c a2  a0 7d 6c a2  43 6f 56 a2  |.ak.|.}l.|.}l.|CoV.|
000000a0  79 09 55 a2  80 61 6b a2  00 00 00 00  00 00 00 00  |y.U.|.ak.|....|....|
000000b0  80 61 6b a2  73 0c 55 a2  79 09 55 a2  80 61 6b a2  |.ak.|s.U.|y.U.|.ak.|
[ ... shortened ... ]
00001090  48 03 00 00  04 00 00 00  04 00 00 00  52 e5 74 64  |H...|....|....|R.td|
000010a0  0c dc 00 00  0c ec 00 00  0c ec 00 00  f4 03 00 00  |....|....|....|....|
000010b0  f4 03 00 00  06 00 00 00  04 00 00 00  2f 73 79 73  |....|....|....|/sys|
000010c0  74 65 6d 2f  62 69 6e 2f  6c 69 6e 6b  65 72 00 00  |tem/|bin/|link|er..|
000010d0  08 00 00 00  04 00 00 00  01 00 00 00  41 6e 64 72  |....|....|....|Andr|
000010e0  6f 69 64 00  17 00 00 00  04 00 00 00  10 00 00 00  |oid.|....|....|....|
000010f0  03 00 00 00  47 4e 55 00  04 4c 7f c8  05 6f fa 0e  |....|GNU.|.L..|.o..|
00001100  03 9d 65 00  bb 92 cf 48  00 00 00 00  00 00 00 00  |..e.|...H|....|....|
[ ... shortened ... ]
```

With regard to the update and patch management strategies of different vendors and with regards to the complexity of exploit development, SySS GmbH considers the impact as moderate. This is especially true for business servers, PCs and laptops. However, the threat is much higher for smartphones and other IoT devices with Bluetooth host stacks. Mitigation strategies can be found in Section 5.

4.4 Denial-of-service by simulated hosts

L1.1

A Bluetooth peripheral identifies the host it is paired with by its Bluetooth address (BD_ADDR). SySS GmbH tested the behavior of keyboards when it comes to simulated hosts. In this setup, an attacker clones the Bluetooth address of the victim's host device in order to provoke the peripheral to connect to the wrong host. Without knowing the key, the attacker obviously cannot establish a full connection and decrypt the keystrokes. But this setup can be used for denial-of-service attacks, as the keyboard might try to connect to the attackers host over and over again.

Because users tend to re-pair devices which do not work properly, this behavior can be used to perform the eavesdropping attack described in Section 4.6.

However, this described denial-of-service attack is not completely reliable. In the performed tests, the keyboards managed to connect to the correct host after several connection attempts to the simulated host.

4.5 Denial-of-service by pairing request

L1.2

If a client device actively tries to pair with a Windows 10 host system, the operating system shows the user a dialog for further instructions. Depending on the devices' capabilities, Windows may display a passkey to be entered or compared. If the device does not have such capabilities, the system will simply ask the user for his confirmation to accept the pairing request.

During the research project, SySS GmbH noticed that is is not possible to pair another device until the user has dealt with the pairing request of the first device. Furthermore, the pairing request of the second device must be repeated if it was sent while there was a pending pairing request of the first device.

An attacker might continuously send pairing request to a victim machine. Besides annoying the victim with the notification that a device is ready for pairing, this could be used for a denial-of-service attack. As long as the attacker is repeating the pairing requests with a randomized source Bluetooth address, it is hardly possible to pair another device.

Mitigating this issue can be achieved by setting the Bluetooth host into a non-pairable state and only making it pairable if desired. Although this can be easily configured on a Linux host, SySS GmbH did not find a way to do this in the Bluetooth configuration dialog in Windows 10.

4.6 Pairing of Microsoft's Designer Desktop

M1.3

The Designer Keyboard by Microsoft does not use Bluetooth Classic like the keyboards by Logitech or 1byone, but it uses Bluetooth Low Energy (BLE, also known as Bluetooth Smart). BLE differs in most ways from the classical Bluetooth specification and was added in version 4.0.

One big difference is the pairing process. Even according to the specification, BLE devices prior to version 4.2 are vulnerable to passive eavesdropping.

Mike Ryan demonstrated that sniffing the communication of a BLE device during the pairing process can be used to extract the Long Term Key (LTK). This cryptographic key can be used to decrypt the current and future Bluetooth device communication.

SySS GmbH used the Ubertooth One hardware and the CrackLE[8] software tool to evaluate whether Microsoft's Designer Keyboard is vulnerable to this kind of attack. The Ubertooth was used to passively eavesdrop on the pairing process and CrackLE was used to extract the Long Term Key and the passcode entered by the user for authentication.

8 https://github.com/mikeryan/crackle/

The following output shows the result of a successful eavesdropping attack against a pairing process of the tested Microsoft Bluetooth keyboard.

```
Analyzing connection 19:
  cc:3d:82:36:cd:3c (public) -> e4:0c:4f:54:55:98 (random)
  Found 48 encrypted packets
  Cracking with strategy 0, 20 bits of entropy

  !!!
  TK found: 969341
  !!!

  Decrypted 48 packets
  LTK found: 7a7bb213d072e7b488f25d9f8b4f78da

Specify an output file with -o to decrypt packets!
```

As the above output of CrackLE shows, the Designer Keyboard is vulnerable. However, this attack is mitigated by the fact that the attacker must be in physical proximity at the time of pairing. If the attacker is not present during pairing, the connection and following connections are currently considered secure.

This is also true for the mouse of Microsoft's Designer Desktop. The communication is based on an encrypted BLE connection as well, but lacking the input buttons, no PIN (TK) is required.

To ensure the secure use of Bluetooth Low Energy devices, SySS GmbH recommends using only the peripheral with ECDH[9] key exchange. This key exchange was introduced in version 4.2 of the Bluetooth specification and is secure against passive eavesdropping.

Initial pairing of Bluetooth devices in a secure environment is not considered sufficient, because users might re-pair the devices later in insecure environments, e.g. due to technical issues.

4.7 Authentication on 1byone's keyboard

L1.3

The Bluetooth classic keyboard by 1byone has a different behavior during the pairing process than the one of Microsoft or Logitech. When pairing with a Windows- or Linux-based PC, the user is requested to enter a PIN on the Bluetooth keyboard, which is shown on the PC. This kind of authentication is used to ensure that the keyboard is paired with the intended host.

9 Elliptic Curve Diffie-Hellman

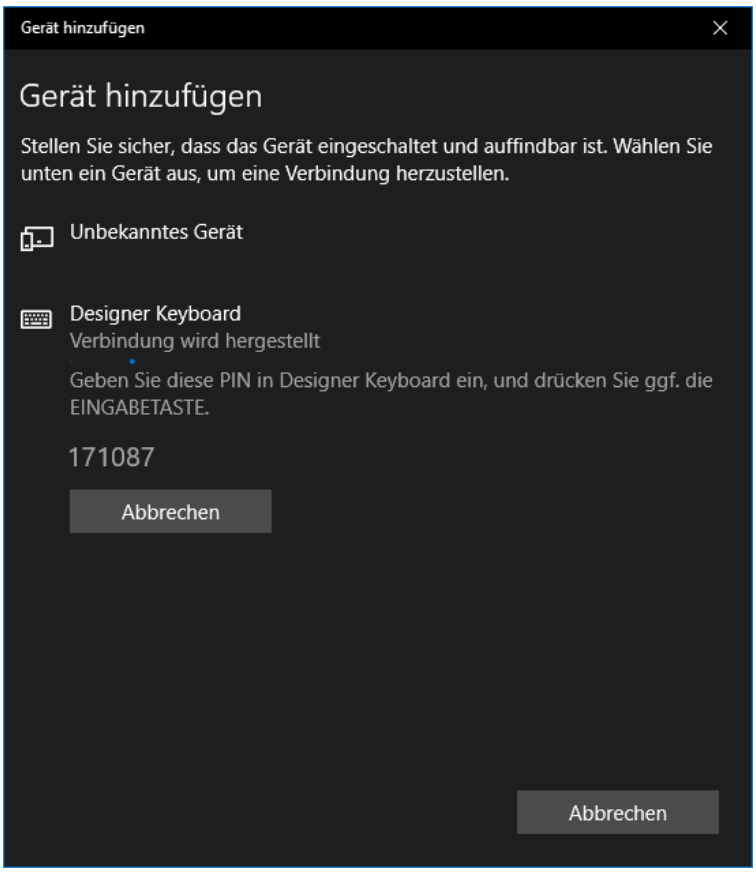

Figure 10: The PIN a user must enter upon pairing (displayed by Windows 10)

However, when pairing the keyboard of 1byone with a computer, no PIN entry is requested and needed. The keyboard pairs using the "Just Works" method of the Bluetooth specification using Simple Pairing with an unauthenticated key. Although the communication is encrypted, this method does not offer any authentication. An attacker might even emulate the host PC of a victim and the victim will not notice as the keyboard gets paired to the wrong computer. The attacker might even emulate a Bluetooth keyboard which then pairs to the victim's host. By proxying the keystrokes, the attacker is now in a man-in-the-middle position and able to log all typed keys.

The following excerpt of the captured Bluetooth communication shows the Simple Pairing process with an unauthenticated combination key (type 4) of the 1byone keyboard with a Linux Arch host.

```
Frame 46: 10 bytes on wire (80 bits), 10 bytes captured (80 bits) on interface 0
Bluetooth
    [Source: host]
    [Destination: controller]
Bluetooth HCI H4
    [Direction: Sent (0x00)]
    HCI Packet Type: HCI Command (0x01)
Bluetooth HCI Command - Link Key Request Negative Reply
    Command Opcode: Link Key Request Negative Reply (0x040c)
    Parameter Total Length: 6
    BD_ADDR: 20:73:00:25:25:1e (20:73:00:25:25:1e)
    [Response in frame: 47]
    [Command-Response Delta: 0.896ms]

Frame 47: 13 bytes on wire (104 bits), 13 bytes captured (104 bits) on interface 0
Bluetooth
```

```
    [Source: controller]
    [Destination: host]
Bluetooth HCI H4
    [Direction: Rcvd (0x01)]
    HCI Packet Type: HCI Event (0x04)
Bluetooth HCI Event - Command Complete
    Event Code: Command Complete (0x0e)
    Parameter Total Length: 10
    Number of Allowed Command Packets: 1
    Command Opcode: Link Key Request Negative Reply (0x040c)
    Status: Success (0x00)
    BD_ADDR: 20:73:00:25:25:1e (20:73:00:25:25:1e)
    [Command in frame: 46]
    [Command-Response Delta: 0.896ms]

Frame 48: 9 bytes on wire (72 bits), 9 bytes captured (72 bits) on interface 0
Bluetooth
    [Source: controller]
    [Destination: host]
Bluetooth HCI H4
    [Direction: Rcvd (0x01)]
    HCI Packet Type: HCI Event (0x04)
Bluetooth HCI Event - IO Capability Request
    Event Code: IO Capability Request (0x31)
    Parameter Total Length: 6
    BD_ADDR: 20:73:00:25:25:1e (20:73:00:25:25:1e)

Frame 49: 13 bytes on wire (104 bits), 13 bytes captured (104 bits) on interface 0
Bluetooth
    [Source: host]
    [Destination: controller]
Bluetooth HCI H4
    [Direction: Sent (0x00)]
    HCI Packet Type: HCI Command (0x01)
Bluetooth HCI Command - IO Capability Request Reply
    Command Opcode: IO Capability Request Reply (0x042b)
    Parameter Total Length: 9
    BD_ADDR: 20:73:00:25:25:1e (20:73:00:25:25:1e)
@textbf{@syblue{    IO Capability: Display Yes/No (1)}@}@
    OOB Data Present: OOB Authentication Data Not Present (0)
@textbf{@syblue{    Authentication Requirements: MITM Protection Required - Dedicated ⤸
    Bonding. Use IO Capability To Determine Procedure, No Secure Connection (3)}@}@
    [Response in frame: 50]
    [Command-Response Delta: 0.904ms]

Frame 50: 13 bytes on wire (104 bits), 13 bytes captured (104 bits) on interface 0
Bluetooth
    [Source: controller]
    [Destination: host]
Bluetooth HCI H4
    [Direction: Rcvd (0x01)]
    HCI Packet Type: HCI Event (0x04)
Bluetooth HCI Event - Command Complete
    Event Code: Command Complete (0x0e)
    Parameter Total Length: 10
```

```
    Number of Allowed Command Packets: 1
    Command Opcode: IO Capability Request Reply (0x042b)
    Status: Success (0x00)
    BD_ADDR: 20:73:00:25:25:1e (20:73:00:25:25:1e)
    [Command in frame: 49]
    [Command-Response Delta: 0.904ms]

Frame 51: 12 bytes on wire (96 bits), 12 bytes captured (96 bits) on interface 0
Bluetooth
    [Source: controller]
    [Destination: host]
Bluetooth HCI H4
    [Direction: Rcvd (0x01)]
    HCI Packet Type: HCI Event (0x04)
Bluetooth HCI Event - IO Capability Response
    Event Code: IO Capability Response (0x32)
    Parameter Total Length: 9
    BD_ADDR: 20:73:00:25:25:1e (20:73:00:25:25:1e)
§R[    IO Capability: No Input, No Output (0x03)]R§
    OOB Data Present: OOB Authentication Data Not Present (0)
§R[    Authentication Requirements: MITM Protection Not Required - General Bonding. ⤸
    Numeric Comparison, Automatic Accept Allowed, No Secure Connection (4)]R§

Frame 52: 13 bytes on wire (104 bits), 13 bytes captured (104 bits) on interface 0
Bluetooth
    [Source: controller]
    [Destination: host]
Bluetooth HCI H4
    [Direction: Rcvd (0x01)]
    HCI Packet Type: HCI Event (0x04)
Bluetooth HCI Event - User Confirmation Request
    Event Code: User Confirmation Request (0x33)
    Parameter Total Length: 10
    BD_ADDR: 20:73:00:25:25:1e (20:73:00:25:25:1e)
§R[    Numeric Value: 533668]R§

Frame 53: 10 bytes on wire (80 bits), 10 bytes captured (80 bits) on interface 0
Bluetooth
    [Source: host]
    [Destination: controller]
Bluetooth HCI H4
    [Direction: Sent (0x00)]
    HCI Packet Type: HCI Command (0x01)
Bluetooth HCI Command - User Confirmation Request Reply
    Command Opcode: User Confirmation Request Reply (0x042c)
    Parameter Total Length: 6
    BD_ADDR: 20:73:00:25:25:1e (20:73:00:25:25:1e)
    [Response in frame: 54]
    [Command-Response Delta: 10.766ms]

Frame 54: 13 bytes on wire (104 bits), 13 bytes captured (104 bits) on interface 0
Bluetooth
    [Source: controller]
    [Destination: host]
Bluetooth HCI H4
```

```
    [Direction: Rcvd (0x01)]
    HCI Packet Type: HCI Event (0x04)
Bluetooth HCI Event - Command Complete
    Event Code: Command Complete (0x0e)
    Parameter Total Length: 10
    Number of Allowed Command Packets: 1
    Command Opcode: User Confirmation Request Reply (0x042c)
    Status: Success (0x00)
    BD_ADDR: 20:73:00:25:25:1e (20:73:00:25:25:1e)
    [Command in frame: 53]
    [Command-Response Delta: 10.766ms]

Frame 55: 10 bytes on wire (80 bits), 10 bytes captured (80 bits) on interface 0
Bluetooth
    [Source: controller]
    [Destination: host]
Bluetooth HCI H4
    [Direction: Rcvd (0x01)]
    HCI Packet Type: HCI Event (0x04)
Bluetooth HCI Event - Simple Pairing Complete
    Event Code: Simple Pairing Complete (0x36)
    Parameter Total Length: 7
    Status: Success (0x00)
    BD_ADDR: 20:73:00:25:25:1e (20:73:00:25:25:1e)
```

SySS GmbH recommends preferring Bluetooth equipment which is able to establish an authenticated connection by default. In some cases, the peripheral is capable to perform authentication, but not forced to do so by the host. By changing the configuration of the host, SySS GmbH was able to request the PIN entry even for the 1byone keyboard. However, configuring all Bluetooth hosts in a business environment is not considered a practical solution to this issue.

5 Conclusion and Recommendations

The test results of our research project show that the three analyzed Bluetooth keyboards are fairly secure – especially when compared to the wireless keyboards which SySS GmbH analyzed in 2015/2016 (see [5]).

Although all tested Bluetooth keyboards behave a bit different, SySS GmbH was not able to identify vulnerabilities which allow a remote attacker to decrypt the communication of paired devices. Also, no keystroke injection attack was possible as long as the cryptographic key material remained secret. However, once an attacker has physical access to one of the keyboards (even just for a couple of minutes), the cryptographic key material (Link Key or Long Term Key) can be stolen. The cryptographic key can be used to conduct further attacks against the paired host system.

Overall, the more complex Bluetooth stacks on hosts become, the more interesting and lucrative target they represent for attackers. As demonstrated by Armis, threats like BlueBorne are real and feasible. Furthermore, Bluetooth devices with more functionality or a very dedicated use outside of the Bluetooth specification (e.g. smart locks) tend to have security issues.

With all this in mind, SySS GmbH created a basic set of rules which may help to increase the security when it comes to dealing with Bluetooth-enabled devices.

- Do not leave Bluetooth peripherals unattended, because attackers might extract the cryptographic keys from the hardware devices.
- Do not leave unused Bluetooth devices in the list of paired devices as it will increase the attack surface (this is somewhat equivalent to an open and unused TCP port).
- Perform device updates often and regularly. The Bluetooth stack is complex and updates might fix serious security issues (e.g. like BlueBorne).
- Make sure that the Bluetooth device in use is actually using encryption. A lot of (smart) devices do not.
- Make sure the pairing process is authenticated (e.g. compare PINs or enter a passkey).
- When using Bluetooth Low Energy (a.k.a. Bluetooth Smart) prior to version 4.2, make sure to pair the device in a secure environment. Otherwise, an attacker eavesdropping on the pairing process can sniff the cryptographic key (Long Term Key) and decrypt current and future communication.
- Do not use Bluetooth devices before version 2.1. There are several known vulnerabilities.
- Make hosts non-pairable and only switch them to be pairable if necessary.
- Prefer Bluetooth peripherals with I/O capabilities (e.g. with a keyboard or a PIN pad) and make sure they use them for authentication during the pairing process.

The National Institute of Standards and Technology (NIST) also published a "Guide to Bluetooth Security" as "Special Publication 800-121, Revision 2" [6]. This document contains a very comprehensive Bluetooth security checklist in Section 4.4.

Since we could not address all of our open and newly raised questions concerning the security of modern Bluetooth devices during this short research project, we intend to continue our research effort regarding Bluetooth technology in the near future.

References

[1] Bluetooth SIG, Bluetooth Core Specification Version 5.0, https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=421043, 2016 1

[2] Matthias Deeg and Gerhard Klostermeier, SySS GmbH, Bluetooth Keyboard Emulator, https://github.com/SySS-Research/bluetooth-keyboard-emulator, 2018 6, 14

[3] Ben Seri and Gregory Vishnepolsky, Armis Inc., BlueBorne, https://www.armis.com/blueborne/, 2017 15

[4] Armis Inc., BlueBorne GitHub Repository, https://github.com/ArmisSecurity/blueborne, 2017 15

[5] Matthias Deeg and Gerhard Klostermeier, SySS GmbH, Of Mice and Keyboards: On the Security of Modern Wireless Desktop Sets, https://www.syss.de/fileadmin/dokumente/Publikationen/2017/2017_06_01_of-mice-and-keyboards_paper.pdf, 2017 22

[6] John Padgette et al., NIST Special Publication 800-121 Revision 2, Guide to Bluetooth Security, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-121r2.pdf, 2017 22

THE PENTEST EXPERTS

SySS GmbH 72072 Tübingen Germany +49 (0)7071 - 40 78 56-0 info@syss.de

WWW.SYSS.DE