SySS
THE PENTEST EXPERTS.

IT SECURITY KNOW-HOW

Matthias Deeg and Gerhard Klostermeier

RIKKI DON'T LOSE THAT BLUETOOTH DEVICE

Exploiting the Obvious: Bluetooth Trust Relationships

July 2018

Schaffhausenstraße 77, 72072 Tübingen, Germany

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

info@syss.de

www.syss.de

1 Introduction

Trust is a tricky thing. If you trust the wrong people or the wrong things, you might get into trouble. That is the reason why it is generally a good idea to reconsider one's trust relationships from time to time, because the assumptions these trust relationships are based on may not be true anymore, or maybe never were.

In this article, we want to present an example of exploiting a trust relationship between two technical devices that can put the confidentiality of sensitive data or the integrity of a computer system at risk. This trust relationship we are going to exploit exists between two Bluetooth devices: On the one side a computer system you want to remain secure and you don't want to be compromised, for example your laptop, or your smartphone, and on the other side a Bluetooth device you usually do not consider worth protecting with special diligence as it simply is an output device of a specific kind and does not persistently store any of your valuable data locally, for example headphones.

2 Bluetooth Security Basics

The wireless technology standard known as Bluetooth for exchanging data over short distances is actually more than one standard that uses a multitude of protocols and exists currently in different versions from Bluetooth 1.0 to Bluetooth 5.0.

Bluetooth is managed by the Bluetooth Special Interest Group (SIG) and specified in Bluetooth SIG documents like the *Bluetooth Core Specification Version 5.0* [1].

In order to understand the attack vector presented in this paper, some basic knowledge about the Bluetooth security architecture and the Bluetooth security model is required. Hence, we will give a very brief introduction in this section that summarizes information from the aforementioned *Bluetooth Core Specification Version 5.0* and the excellent *Guide to Bluetooth Security* [2] by the National Institute of Standards and Technology (NIST).

The Bluetooth security model includes the following five security features:

1. **Pairing**: The process for creating one or more shared secret keys
2. **Bonding**: The act of storing the keys created during pairing for use in subsequent connections in order to form a trusted device pair
3. **Device authentication**: Verification that the two devices have the same keys
4. **Encryption**: Message confidentiality
5. **Message integrity**: Protects against message forgeries

During the evolution of the Bluetooth technology, also its security architecture evolved, and different cryptographic algorithms were used for different purposes like authentication or encryption.

In general, there is a distinction between Bluetooth BR/EDR (Basic Rate/Enhanced Data Rate) and Bluetooth LE (Low Energy, BLE, a.k.a. Bluetooth Smart) devices. The key differences of these two Bluetooth device types are shown in Table 2[1].

1 from Section 2.1, *Guide to Bluetooth Security* [2]

Characteristic	Bluetooth BR/EDR		Bluetooth Low Energy	
	Prior to 4.1	*4.1 onwards*	*Prior to 4.2*	*4.2 onwards*
RF physical channels	79 channels with 1 MHz spacing		40 channels with 2 MHz spacing	
Discovery/connect	Inquiry/paging		Advertising	
Number of piconet slaves	7 (active)/255 total		Unlimited	
Device address privacy	None		Private device addressing available	
Max data rate	1-3 MBps		1 MBps via GFSK modulation	
Pairing Algorithm	*Prior to 2.1:* E21/E22/SAFER+ *2.1 to 4.0:* Elliptic Curve P-192[2], HMAC-SHA-256	P-256 Elliptic Curve, HMAC-SHA-256	AES-128	P-256 Elliptic Curve, AES-CMAC
Device authentication algorithm	E1/SAFER	HMAC-SHA-256	AES-CCM	
Encryption algorithm	E0/SAFER+	AES-CCM	AES-CCM	
Typical range	30 m		50 m	
Max output power	100 mW (20 dB)		10 mW (10 dB)/100 mW (20 dB)	

Table 2: Key differences between Bluetooth BR/EDR and Low Energy (source: *Guide to Bluetooth Security* [2])

In order to communicate with each other, two Bluetooth devices have to share a secret. This shared secret, a cryptographic key, is created during the pairing process. How the actual shared key is generated depends on the IO capabilities, of the Bluetooth devices shown in Table 4.

Capability	Type	Description
NoInput	Input	Device does not have the ability to indicate "yes" or "no"
Yes/no	Input	Device has at least two buttons that can be easily mapped to "yes" and "no" or the device has a mechanism whereby the user can indicate either "yes" or "no"
Keyboard	Input	Device has a numeric keyboard that can input the numbers 0 through 9 and a confirmation Device also has at least two buttons that can be easily mapped to "yes" and "no" or the device has a mechanism whereby the user can indicate either "yes" or "no"
NoOutput	Output	Device does not have the ability to display or communicate a 6 digit decimal number
Numeric output	Output	Device has the ability to display or communicate a 6 digit decimal number

Table 4: Bluetooth device IO capabilities (source: *Bluetooth Core Specification* [1])

Table 6 illustrates the mapping of IO Capabilities to the used key generation method.

Responder	*Initiator* DisplayOnly	Display YesNo	Keyboard Only	NoInput NoOutput	Keyboard Display
Display Only	Just works Unauthenticated	Just works Unauthenticated	Passkey entry Authenticated	Just works Unauthenticated	Passkey entry Authenticated
Display YesNo	Just works Unauthenticated	*For LE* *Legacy pairing:* Just works Unauthenticated *For LE* *Secure connections:* Numeric comparison Authenticated	Passkey entry Authenticated	Just works Unauthenticated	*For LE* *Legacy pairing:* Passkey entry Authenticated *For LE* *Secure connections:* Numeric comparison Authenticated
Keyboard Only	Passkey entry Authenticated	Passkey entry Authenticated	Passkey entry Authenticated	Just works Unauthenticated	Passkey Entry Authenticated
NoInput NoOutput	Just works Unauthenticated	Just works Unauthenticated	Just works Unauthenticated	Just works Unauthenticated	Just works Unauthenticated
Keyboard Display	Passkey entry Authenticated	*For LE* *Legacy pairing:* Passkey entry Authenticated *For LE* *Secure connections:* Numeric comparison Authenticated	Passkey entry Authenticated	Just works Unauthenticated	*For LE* *Legacy pairing:* Passkey entry Authenticated *For LE* *Secure connections:* Numeric comparison Authenticated

Table 6: Mapping of IO capabilities to key generation method (source: *Bluetooth Core Specification* [1])

For understanding the attack vector described in the following section, the details of the different key generation methods, the different security key hierarchies, and used key derivation functions do not matter. It is only important to know that each paired, or more precisely bonded, Bluetooth device locally stores one or more shared secret keys created during the pairing process for use in subsequent connections in order to form a trusted device pair.

When talking about Bluetooth BR/EDR devices, this persistently stored shared cryptographic key material is called Link Key (LK), when talking about Bluetooth LE devices, it is called Long Term Key (LTK).

Besides this cryptographic key material, bonded Bluetooth devices also store at least the Bluetooth device address (`BD_ADDR`) of the devices they share secret keys with.

3 Exploiting the Obvious

During a research project concerning Bluetooth keyboards [3], we made the following two observations that combined result in an interesting attack vector regarding Bluetooth trust relationships, at least in our opinion:

1. Cryptographic key material of bonded Bluetooth devices can be extracted by an attacker with physical access without much difficulties.
2. Most of the Bluetooth stacks of modern operating systems do not strictly bind specific properties of a bonded Bluetooth device with its pairing information.

We developed a software tool named Bluetooth Keyboard Emulator [4] based on existing, publicly available software projects in order to emulate Bluetooth Classic (BR/EDR) keyboards for different test cases.

The Bluetooth Keyboard Emulator used the Bluetooth HID (Human Interface Device) Profile and worked flawlessly using extracted link keys of tested Bluetooth Classic keyboards without any complaints by the paired client systems with different operating systems (Arch Linux, Microsoft Windows 10, Apple Mac OS X, iOS 11, and Android 8). So we wondered, what happens if we do not replace a real Bluetooth keyboard by an emulated one, but another type of Bluetooth device by an emulated keyboard.

So we made some tests with Bluetooth headphones and replaced an output device (headphones) with an input device (emulated keyboard) using at least the same Bluetooth device address and the same cryptographic key material (link key).

During the tests with Bluetooth keyboards, we noticed that most Bluetooth stacks of the operating systems we used do not care if bonded Bluetooth devices change their attributes. For example, a device can change its name, vendor ID, product ID, or serial number, and the paired host will not complain, as long as the Bluetooth device address and the link key stay the same. But the more interesting observation is that some Bluetooth stacks also do not care if a device changes its device class and its capabilities. And being able to change the behavior of a device, for instance from an output device like headphones to an input device like a keyboard, that has a trust relationship with another device, for example a smartphone or a laptop, is very interesting for an attacker.

The following steps outline the possible attack scenario:

1. The victim buys Bluetooth headphones and pairs them to its 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, or the attacker simply has physical access to them for a couple of minutes).
3. The attacker extracts the pairing information from the headphones (Bluetooth device address and link key).
4. The attacker uses the extracted pairing information to establish a valid connection to the victim's computer or smartphone with an emulated device.
5. Depending on the Bluetooth stack of the victim's computer or smartphone, the emulated Bluetooth device can behave as something completely different – for example as Bluetooth keyboard instead of headphones – which enables an attacker to perform malicious actions on the victim's device, for example in order to gain access to sensitive data

SySS GmbH tested the described attack vector using Pioneer SE-MJ553BT-K Bluetooth Classic headphones which were previously paired to different client systems (laptops or smartphones) with different operating systems.

In three of the five test cases, the attack using an emulated Bluetooth keyboard was successful. The corresponding client systems accepted and established a connection with the emulated Bluetooth keyboard which was using the extracted cryptographic key (link key) of the Bluetooth headphones. Although the originally paired device were headphones, the tested Android, iOS, and Mac OS X client systems were not concerned with the fact that the trust relationship to headphones now suddenly was a trust relationship to a keyboard, and the emulated Bluetooth keyboard worked flawlessly with those client systems.

This attack vector did not work against our Windows 10 and Arch Linux test systems. 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 behavior of a bonded Bluetooth device, or it might just be due to technical issues and missing features in our developed Bluetooth keyboard emulator.

Table 8 summarizes the test results regarding this attack against different tested client systems.

Client operating system	OS version	Successful attack
Android	7.1.2	✓
Android	8.1.0	✓
Arch Linux	4.16.13-2-ARCH #1	X
Apple iOS	11.2.6	✓
Apple iOS	11.3	✓
Apple iOS	11.4	✓
Apple Mac OS X	10.13.4	✓
Apple Mac OS X	10.13.5	✓
Microsoft Windows 10	1709 (OS Build 16299.125)	X

Table 8: Attacks against different client systems

A quite similar attack named *DirtyTooth* [5] exploiting Bluetooth trust relationships and changing Bluetooth device behavior via Bluetooth profiles against Apple iOS devices prior version 11.2 was published last year by ElevenPaths. In the *DirtyTooth* attack scenario, the victim pairs his iOS device with a malicious Bluetooth device, for instance a Bluetooth speaker. The trick here is that the malicious Bluetooth speaker does not only provide speaker functionalities via the initially announced Bluetooth profile A2DP (Advanced Audio Distribution), but that it can change its functionality after the completed device pairing by switching to another Bluetooth profile like PBAP (Phone Book Access Profile).

ElevenPaths found out that iOS devices prior version 11.2 accepted such a profile change without any user notification and enabled the synchronization of contacts by default giving the malicious Bluetooth device access to all phone book entries.

Our test results using our developed Bluetooth keyboard emulator show, that current iOS versions still accept profile changes of bonded Bluetooth devices which may be exploited by an attacker. And not only iOS devices show this behavior, but also Android, and Mac OS X devices.

In the remainder of this section, we want to illustrate an actual attack exploiting the trust relationship between Pioneer SE-MJ553BT-K Bluetooth headphones shown in Figure 1 and an Android tablet running Android version 7.1.2.

Figure 2 shows the Android Bluetooth menu of the Android tablet with the paired headphones.

Figure 1: Pioneer Bluetooth headphones SE-MJ553BT-K

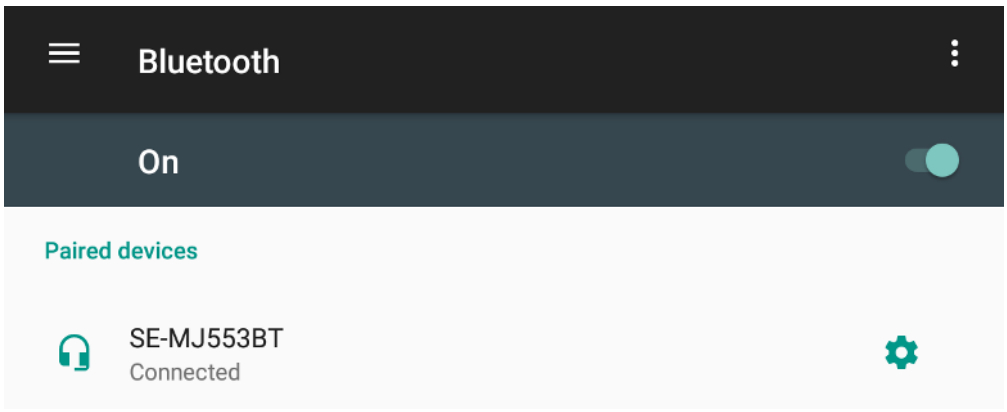

Figure 2: Android Bluetooth settings with paired Pioneer SE-MJ553BT-K headphones

Figure 3 shows the detailled Bluetooth settings regarding the paired Pioneer SE-MJ553BT-K headphones. It is interesting to note that by default the headphones are not only used for audio functionality by the Android operating system but also for contact sharing. This is because the headphones with its integrated microphone support the two audio-related Bluetooth profiles A2DP[3] and AVRCP[4] as well as the two Bluetooth profiles HFP[5] and HSP[6] which can make good use of phonebook access. This fact allows for more attacks without further ado by the attacker than the one described previously in the attack vector, as we will see below.

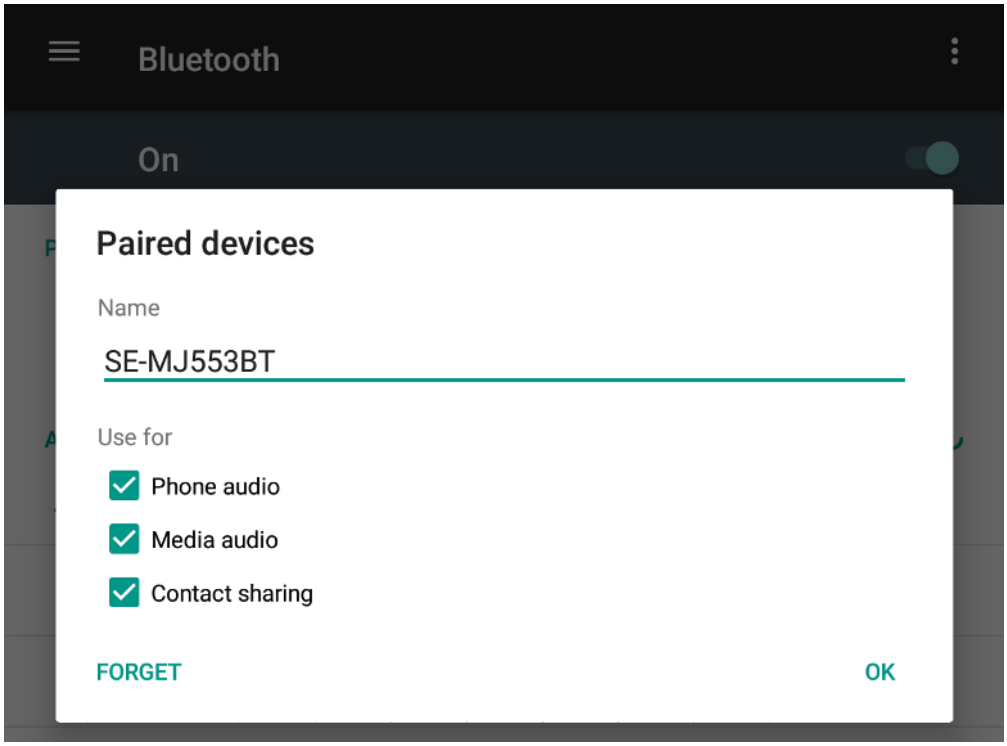

Figure 3: Detailled Bluetooth settings for Pioneer SE-MJ553BT-K headphones

In order to perform an attack with an emulated Bluetooth device like a Bluetooth keyboard, an attacker has to extract the required pairing information, i.e. Bluetooth device address and cryptographic key (link key), from the headphones.

3 Advanced Audio Distribution Profile
4 Audio/Video Remote Control Profile
5 Hands-Free Profile
6 Headset Profile

Concerning the Pioneer SE-MJ553BT-K headphones, this task is quite easy as all required information is stored on a SPI serial flash chip soldered on the PCB.

Figure 4 shows a top view of the whole PCB which is located in the left ear cup and accessible by unscrewing three screws. Pioneer SE-MJ553BT-K headphones use the Qualcomm CSR8635 single-chip Bluetooth Audio Platform[7] in combination with a SPI serial flash chip (type 25LV512).

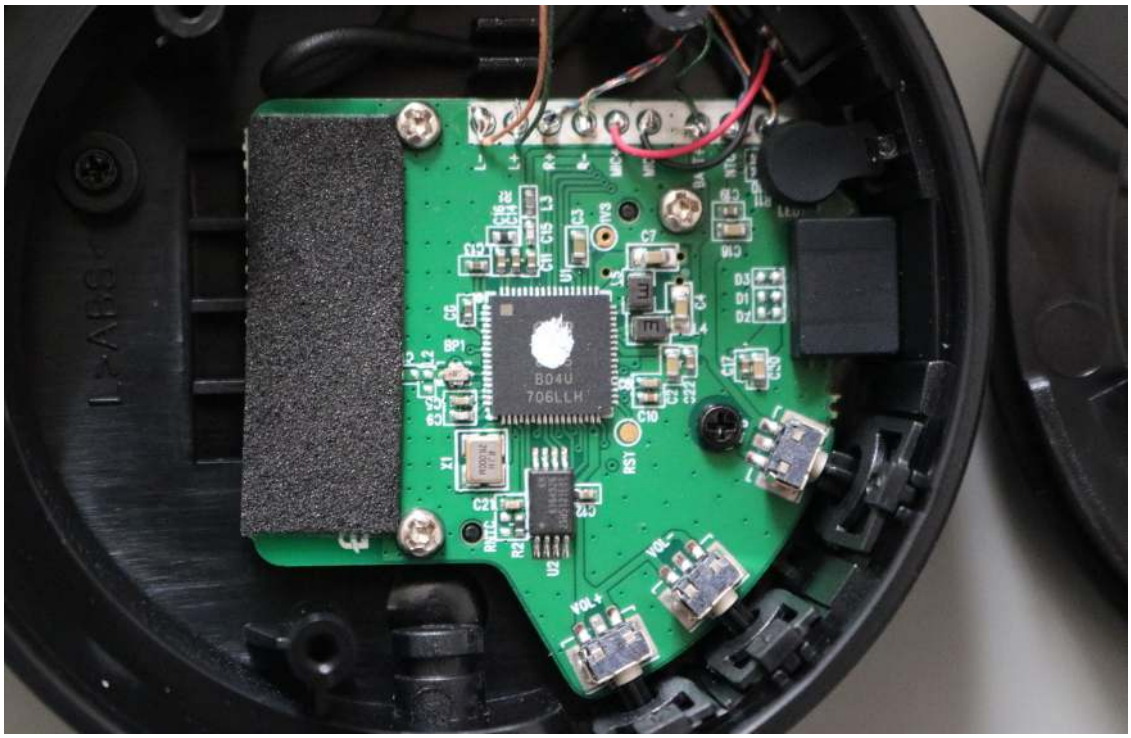

Figure 4: Pioneer headphones PCB

In order to read the content of the SPI serial flash chip, an attacker with physical access has the following two options:

1. **Chip-off**: desoldering the chip from the PCB and reading its content with a suitable programming/reading device
2. **In-circuit reading**: reading the chip content using a suitable adapter in combination with a suitable programming/reading device

Of course, the chip-off method is the more invasive and more time-consuming choice for an attacker to gain access to the Bluetooth pairing information compared to the in-circuit reading. But if there is no limit for the time window regarding physical access to the target device, this method works fine.

In our tests, we could successfully extract the required Bluetooth device address of the paired device and the cryptographic key from the Pioneer SE-MJ553BT-K headphones using both methods.

Figure 5 exemplarily shows the desoldered SPI serial flash chip with TSSOP[8]-8 form factor soldered to a breakout board.

Figure 6 shows the breakout board with the SPI serial flash chip mounted in the used MiniPro TL866A universal programmer[9] demonstrating the chip-off method.

Figure 7 shows a suitable TSSOP-8 adapter that we used in combination with the MiniPro TL886A universal programmer in order to successfully read the content of the SPI serial flash without desoldering it, demonstrating the in-circuit reading method.

7 `https://www.qualcomm.com/products/csr8635`
8 Thin Shrink Small Outline Package
9 `http://autoelectric.cn/EN/TL866_main.html`

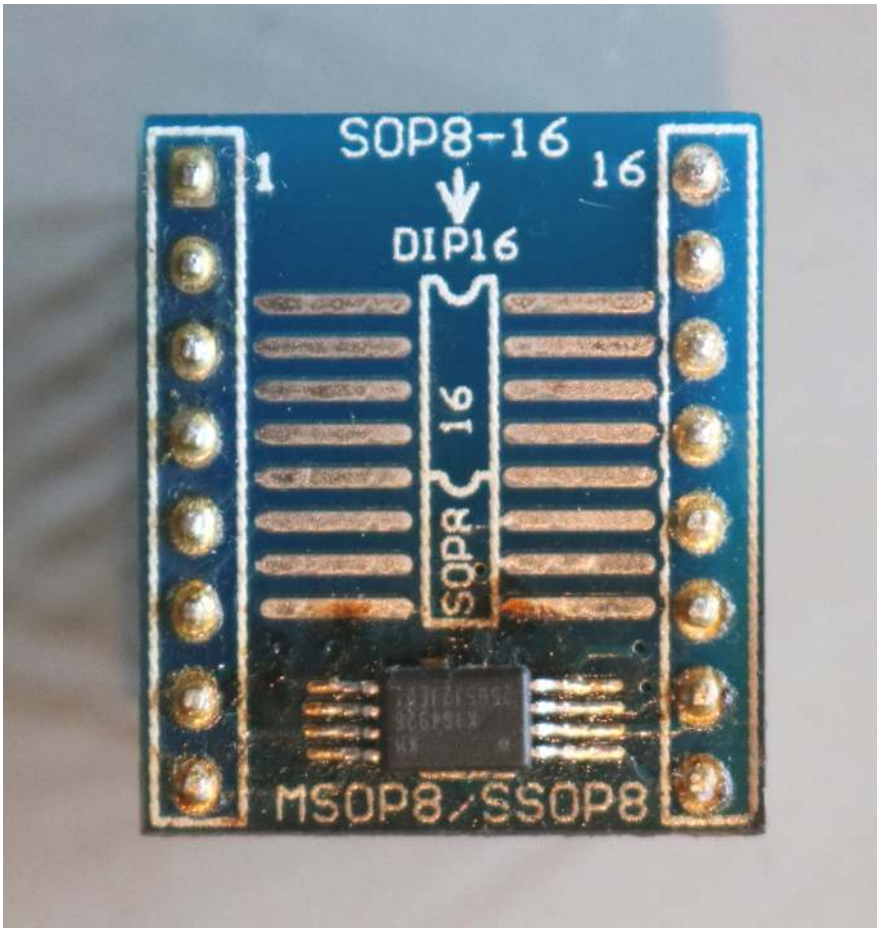

Figure 5: TSSOP-8 SPI serial flash chip on a breakout board

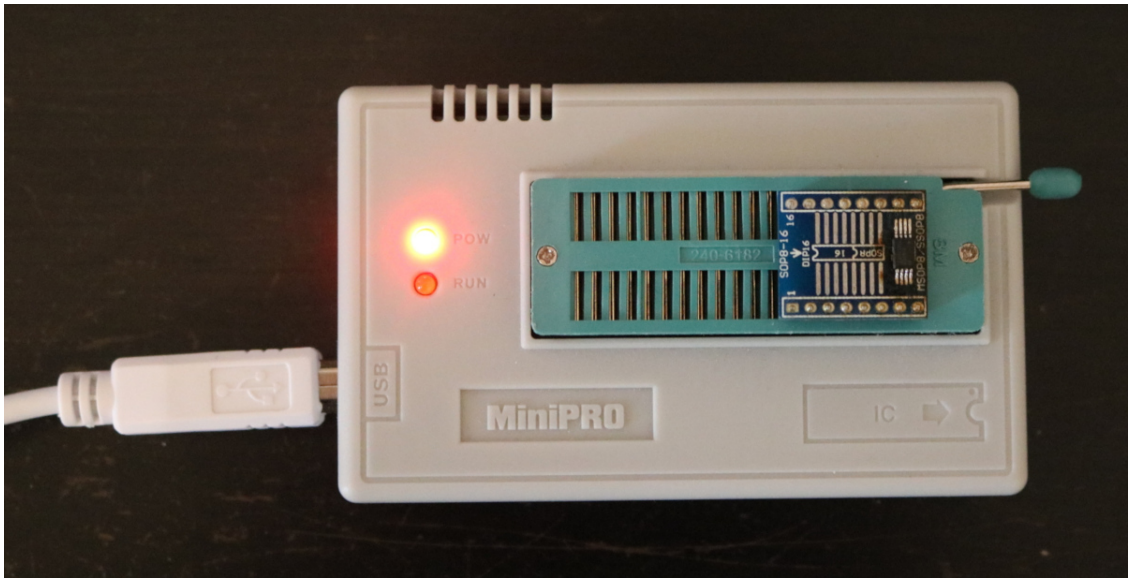

Figure 6: MiniPro TL866A universal programmer

Figure 7: TSSOP-8 adapter used for in-circuit reading of the SPI serial flash

Using the MinPro TL866A universal programmer and the software tool MiniPro v6.71, we were able to read the contents of the headphones' SPI serial flash, as Figure 8 shows.

Highlighted in green, you can see the Bluetooth device address of the paired device – in this case of the Android tablet with the Bluetooth address `AC:22:0B:D6:F5:E4`. Highlighted in red, you can see the 128 bit shared secret, the link key. The actual data format of Bluetooth device addresses and link keys and how they are stored in EEPROM or serial flash chips is device- and firmware-dependent. So for instance, the byte order (endianness) of link keys may vary from device to device, as well as the actual data format used for storing Bluetooth device addresses.

By having extracted the link key and the Bluetooth device addresses of the paired device and the headphones themselves, an attacker is now able to use this information to emulate a device and exploit the existing Bluetooth trust relationship between the headphones and the Android tablet.

On a modern Linux operating system with the BlueZ[10] 5 Bluetooth stack and utilities, for example Arch Linux[11], the attacker has to create a file named `info` containing the link key and storing it within the appropriate directory structure of the Linux operating system.

In this example, the BlueZ info file `/var/lib/bluetooth/F4:0E:11:76:71:AD/AC:22:0B:E4:D6:F5/info` has to be created with the following content.

```
[LinkKey]
Key=03E88F84C3008E7CAB3F78FB7E61FED9
```

`AC:22:0B:E4:D6:F5` is the Bluetooth device address of the Android tablet and `F4:0E:11:76:71:AD` is the Bluetooth device address of the headphones that is required for Bluetooth address spoofing.

Now, the attacker can perform different attacks against the Nexus tablet by exploiting the existing Bluetooth trust relationship, for example using our developed Bluetooth keyboard emulator shown in Figure 9.

10 http://www.bluez.org/
11 https://www.archlinux.org/

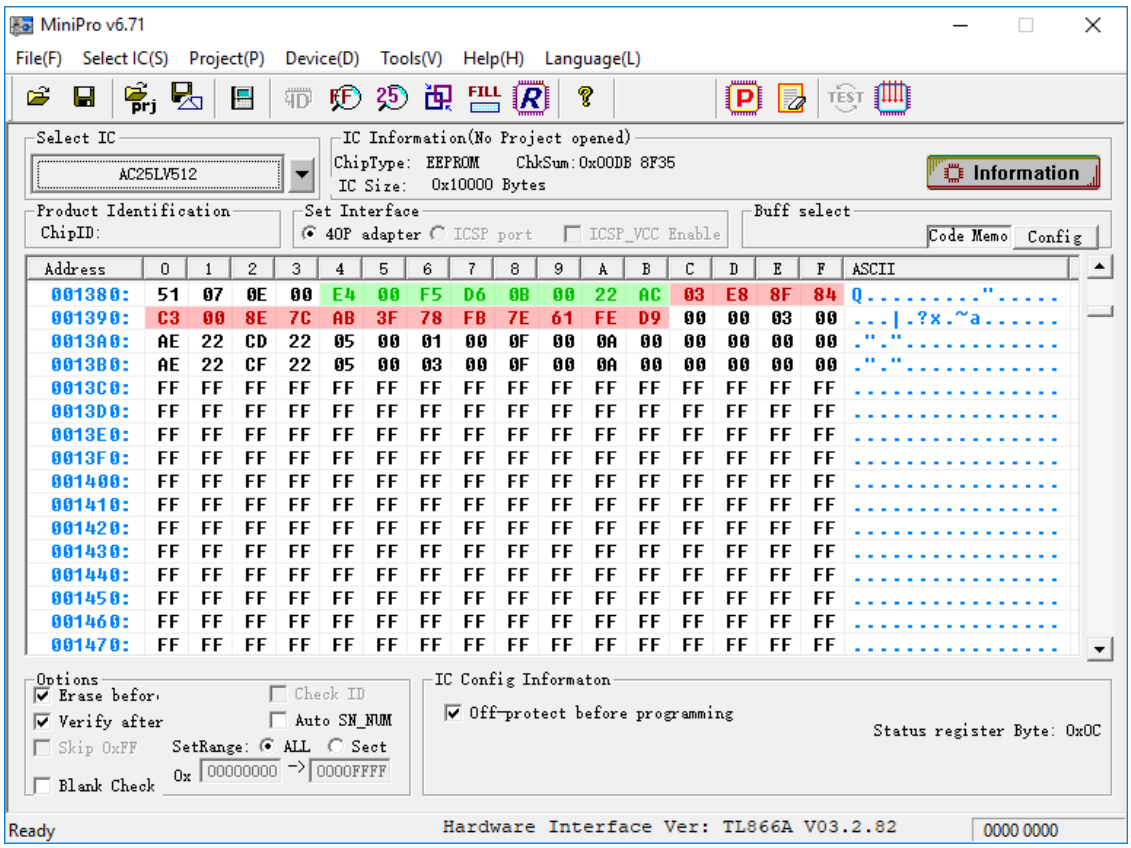

Figure 8: Memory dump of the bonded headphones' SPI serial flash containing the target Bluetooth device address and the link key

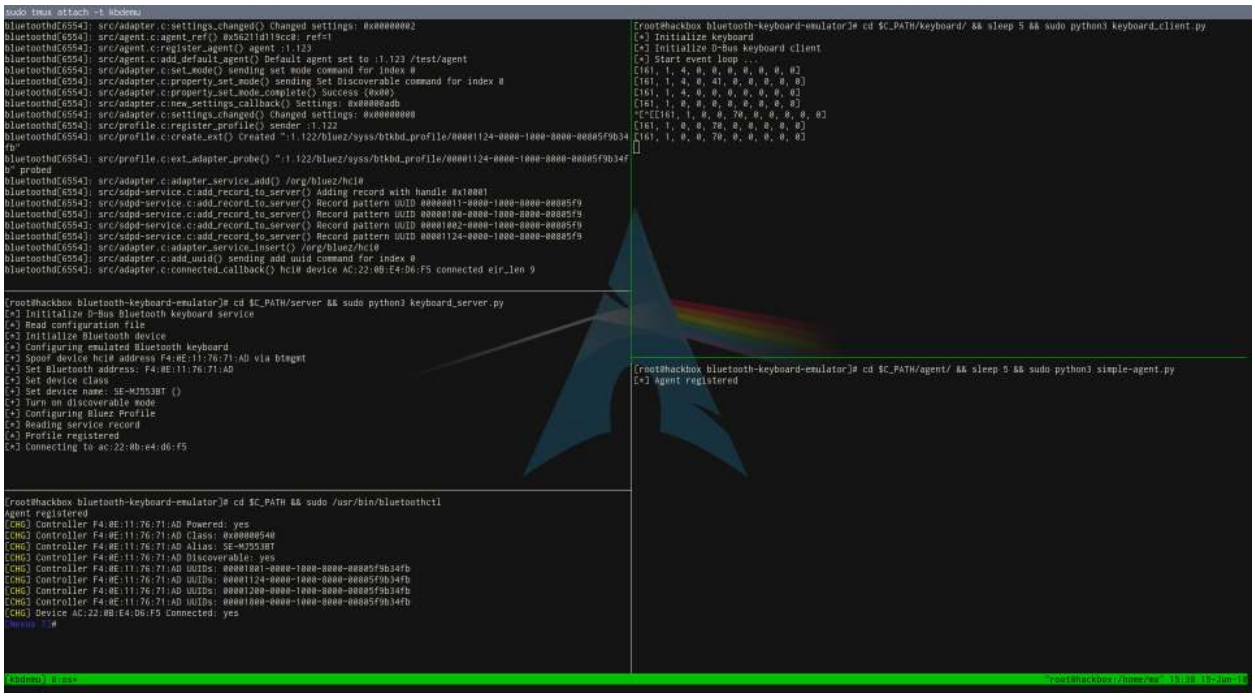

Figure 9: Developed Bluetooth keyboard emulator in use

By emulating a Bluetooth keyboard using the Bluetooth HID[12] profile, an attacker within the Bluetooth radio range (see Table 2) can remotely control the Android tablet in our example and use many of the device features that users who acutally hold the device in their hands can use has at this point in time.

The example of exploiting the Bluetooth trust relationship with a keyboard is of course most interesting for an attacker if he or she can attack the device at very moment when its screen is unlocked. Because attacks to screen-locked devices, no matter if it is a smartphone, a tablet, a laptop, or a desktop computer, offer much less possibilities, at least using an emulated keyboard. But if an attacker can attack a device that is in an unlocked state, it is possible to send arbitrary keypresses, and for example change device settings, exfiltrate data using available e-mail, messenger, or web browser apps, or trigger unwanted actions with the identity of the device owner.

If the Bluetooth trust relationship of the attacked Bluetooth device already allows for using more functionality of the paired target device, like in our example with the Pioneer headphones and the Android tablet with default contact sharing, an attacker can also simply read the whole device phone book via the offered Bluetooth PBAP[13] service. This phone book access is exemplarily illustrated with the used Android tablet in the following output using the freely available open-source software tool `pbapclient` [6].

```
$ python2 pbapclient.py
Welcome to the PhoneBook Access Profile!
pbap> connect AC:22:0B:E4:D6:F5
2018-06-25 16:11:24,004 __main__ INFO     Finding PBAP service ...
2018-06-25 16:11:27,777 __main__ INFO     PBAP service found!
2018-06-25 16:11:27,778 __main__ INFO     Connecting to pbap server = (AC:22:0B:E4:D6:F5, 4)
2018-06-25 16:11:27,994 __main__ INFO     Connect success
pbap> pull_vcard_listing telecom/pb
2018-06-25 16:11:35,417 __main__ INFO     Requesting pull_vcard_listing with parameters {'name': 'telecom/pb', 'self': <__main__.PBAPClient instance at 0x7f2d3225aa28>, 'list_startoffset': 0, 'search_value': None, 'search_attribute': 0, 'order': 0, 'max_list_count': 65535}
2018-06-25 16:11:35,896 __main__ INFO     Result of pull_vcard_listing:
<?xml version="1.0"?><!DOCTYPE vcard-listing SYSTEM "vcard-listing.dtd"><vCard-listing version="1.0"><card handle="0.vcf" name="My name"/><card handle="1.vcf" name="Donald Duck"/><card handle="2.vcf" name="Mickey Mouse"/><card handle="3.vcf" name="Scrooge McDuck"/><card handle="4.vcf" name="Daisy Duck"/><card handle="5.vcf" name="Gyro Gearloose"/><card handle="6.vcf" name="Minnie Mouse"/><card handle="7.vcf" name="Goofy"/><card handle="8.vcf" name="Clarabelle Cow"/><card handle="9.vcf" name="Horace Horsecollar"/><card handle="10.vcf" name="Magica De Spell"/><card handle="11.vcf" name="Gus Goose"/><card handle="12.vcf" name="John D. Rockerduck"/></vCard-listing>
pbap>
```

12 Human Interface Device
13 Phone Book Access Profile

4 Conclusion

The demonstrated attack scenario shows that Bluetooth devices you usually do not consider worth protecting with special diligence may be targeted by attackers in order to perform further attacks against more interesting Bluetooth devices you do not want to be compromised. Thus, you should always consider trust relationships between Bluetooth devices and check from time to time if the assumptions these trust relationships were initially based on are still true.

You should also be aware of the fact that usually Bluetooth devices contain one or more persistently stored shared secrets, called link keys or long term keys, which may be extracted by an attacker with short-time physical access, for instance a couple of minutes, without much difficulties due to insufficient protection of this sensitive data. Having access to this cryptographic key material allows for different attacks, for example exploiting the trust relationship of bonded Bluetooth devices by changing device capabilities as demonstrated with the headphones *becoming* a keyboard.

Our tests using a Bluetooth keyboard emulator for replacing previously paired Bluetooth headphones also showed that Bluetooth stacks of different operating systems behaved differently and that not all targeted client operating systems were prone to this kind of attack.

During our research concerning the security of modern Bluetooth keyboards [3] where we found the described security issue, not all our initial questions could be conclusively answered and some new interesting questions have been raised that we are planning to address in future research projects.

With this paper, we hope to raise the awareness for Bluetooth security issues regarding Bluetooth trust relationships, especially with peripheral devices that usually get less attention when it comes to IT security and whose life-cycles are generally neglected.

If you want to learn more about Bluetooth security vulnerabilities, threats, risks, mitigations, and countermeasures, we highly recommend the *Guide to Bluetooth Security* [2] by the National Institute of Standards and Technology (NIST).

References

[1] Bluetooth SIG, Bluetooth Core Specification Version 5.0, https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=421043, 2016 1, 2, 3

[2] John Padgette et al., National Institute of Standards and Technology (NIST), NIST Special Publication 800-121 Revision 2, Guide to Bluetooth Security, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-121r2.pdf, 2017 1, 2, 13

[3] Matthias Deeg and Gerhard Klostermeier, SySS GmbH, Security of Modern Bluetooth Keyboards, https://www.syss.de/fileadmin/dokumente/Publikationen/2018/Security_of_Modern_Bluetooth_Keyboards.pdf, 2018 4, 13

[4] Matthias Deeg and Gerhard Klostermeier, SySS GmbH, Bluetooth Keyboard Emulator, https://github.com/SySS-Research/bluetooth-keyboard-emulator, 2018 4

[5] Chema Alonso et al.,ElevenPaths, DirtyTooth, http://dirtytooth.com/, 2017 5

[6] Kannan Subramani, BMW Car IT GmbH, Python implementation of Phone Book Access Profile (PBAP), https://github.com/bmwcarit/pypbap, 2018 12

THE PENTEST EXPERTS

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

WWW.SYSS.DE