Bluetooth devices fingerprinting using low cost SDR

Étienne Helluy-Lafont, Alexandre Boé, Gilles Grimaud, Michaël Hauspie

Bluetooth devices fingerprinting using low cost SDR

Étienne Helluy-Lafont*, Alexandre Boé†, Gilles Grimaud* and Michaël Hauspie*
* Univ. Lille, CNRS, Centrale Lille, UMR 9189 - CRIStAL
Centre de Recherche en Informatique Signal et Automatique de Lille
† Univ. Lille, CNRS, ISEN, Univ. Valenciennes, UMR 8520 - IEMN, F-59000 Lille, France

***Abstract*—Physical fingerprinting is a trending domain in wireless security. Those methods aim at identifying transmitters based on the subtle variations existing in their handling of a communication protocol. They can provide an additional authentication layer, hard to emulate, to improve the security of systems. Software Defined Radios (SDR) are a tool of choice for the fingerprinting, as they virtually enable the analysis of any wireless communication scheme. However, they require expensive computations, and are still complex to handle by newcomers. In this paper, we use low cost SDR to propose a physical-layer fingerprinting approach, that allows recognition of the model of a device performing a Bluetooth scan, with more than 99.8% accuracy in a set of ten devices.**

I. Introduction

More and more connected devices are taking place in our homes and offices for various reasons. The widespread deployment of those devices made them a target of choice for large-scale attacks, as the Mirai malware has demonstrated, capable of taking control of many similar-designed devices with weak default configurations [6].

Low-end internet-connected appliances should therefore be closely monitored, especially when they can provide an internal network access from an external attacker. Hopefully, these kind of attacks might be thwarted by existing intrusion detection systems (IDS), since they can be observed at the Internet gateway level.

More concerns may arise when those devices are not only using wired or Wi-Fi Internet access, but also form Wireless Personal Area Networks (WPAN) based on one of the many IoT-friendly RF protocols available (*e.g.* Bluetooth, Zigbee, Z-Wave, etc.). In these configurations, any of those communication channels might be targeted by an external attacker or a Trojan horse to gain some sort of internal network access.

When building an IDS, it is useful to associate a unique identifier to the source of a transmission. On a wired network it could be the Medium Access Control (MAC) address. In wireless networks, collecting the contents of packets is not always enough to obtain a unique identifier for security application since they could be easily forged.

On the other hand, physical characteristics of a wireless device hardware are harder to alter. Thus, using these characteristics to compute a wireless device fingerprint is in wireless networks. Such a technique may help in preventing impersonation attacks, to which wireless communications are, by nature, more exposed.

Wireless device fingerprinting is a trending topic in the literature on IDS. It aims at recognizing different wireless transmitters using only the subtle differences in the way they handle the protocol. The differences might be observed passively at the physical or MAC layer; or actively in response to non standard events. Physical layer fingerprints are believed to be harder to fake for an attacker than digital identifiers, because their modification often requires firmware or hardware modifications. For this reason, we believe physical layer fingerprinting would significantly increase the difficulty for an attacker to introduce rogue devices in a targeted environment.

Software Defined Radios (SDR) are wireless communication systems where a significant part of the frequency conversion and signal processing, are implemented as software. A unique SDR can handle various arbitrary waveforms for transmission or reception of data. Therefore, almost any RF communication scheme can be supported by writing the appropriate software.

The main contribution of this paper is the demonstration of the use of a low cost SDR to discriminate devices sending the same payload. The experimentations were carried on the Bluetooth (BT) inquiry process [12]. To scan for nearby slaves, a Bluetooth master sends out many inquiry packets on different channels. In the BT specification, an inquiry packet is always made of the same bits. After capturing inquiry packets from different BT masters, we focus on three features : frequency-error, preamble duration, and hopping clock skew. We show that these features allow the discrimination of different chip manufacturers or device implementations.

The paper is organized as follows. Section II introduces the bluetooth protocol and the fingerprinting techniques. Section II-D describes the fingerprinting applied to the BT protocol and the experimentations. Sections IV and V describe the analysis of the collected data. Section VI describes the effect of the temperature on the fingerprint process. Finally, section VII presents the conclusions.

II. Background on Bluetooth protocol and fingerprinting

A. Wireless attacks

For obvious reasons, wireless devices are prone to remote attacks. Different architectures are leading to different exposures. A review revealed several vulnerabilities in Bluetooth host implementations, in Linux, Android, and Apple Bluetooth stacks [11]. Those vulnerabilities were located in the higher layers of the protocols. This is not the worst case to work with for an intrusion detection scenario, as the vulnerable code is running on the host, which is convenient to modify, instrument, or sandbox. The attacks targeting the lower protocol

layers, executed in a dedicated baseband chip containing large proprietary firmware, are harder to be monitored by a host-based IDS. In the latest years, more and more researchers targeted baseband processors, and identified severe flaws in widely deployed firmwares: cellular baseband [14], or Broadcom Wi-Fi chips [8]. Baseband research also brings opportunities when instrumentation of existing firmwares can allow monitoring and injection of traffic at the lowest layers, to provide cheap "debug" devices to study and experiment with security at all layers, the firmwares can be instrumented. *Osmocom-bb* is an open-source firmware for early TI chipsets, and allow full control over all layers of GSM [15]. Seemo-lab provided the *Nexmon* framework to instrument Broadcom Wi-Fi chips [10] and more recently, *InternalBlue* for Broadcom Bluetooth baseband. With *InternalBlue*, they were able to add BT Device Address filtering in the baseband, to reduce its attack surface [7].

B. Bluetooth insights

In this Section, we recall some fundamental prerequisites about the physical layer of the Bluetooth Basic Rate to give a better understanding of the next Sections. Because of its early adoption by mobiles phones, BT has become one of the most popular protocols for short-range wireless communications. It is widely used to pair accessories to a mobile phone, to pilot home automation, or control connected devices.

In its first version (Basic Rate), packets are modulated using *Guaussian Frequency Shift Keying* (GFSK), at a baud rate of 1 Mbps. There are 79 BT channels, ranging from 2402 MHz to 2480 MHz. The physical layer uses *Frequency Hopping Spread Spectrum* (FHSS) and *Time Division Duplexing* (TDD). A FHSS communication system frequently switches its transmission channel using a pseudorandom sequence known by both ends of the communication.

In BT, the hopping rate is equal to 1600 hops per seconds (or can be 3200 during Paging or Inquiry). TDD is used to share the transmission channel between the master and the slave. Time is uniformely divided into *slots* ($1/1600s = 625\mu s$). All transmissions must be aligned on the start of a time slot. Since BT version 1.2, *Adaptive Frequency Hopping* (AFH) can improve the performances, by dynamically updating the hopping channel map, to avoid channels with too much interferences.

C. Bluetooth monitoring

Bluetooth is known to be challenging to be listen to by passive monitoring tools. In 2007, the BlueSniff paper [13] was the first to provide practical methods to capture Bluetooth traffic using a SDR receiver built upon the GNU Radio signal processing framework. It enlightened the most severe issues for passive monitoring of Bluetooth. First, the full MAC address of the master must be known to follow the traffic on a piconet. When the device is set in discoverable mode, this can be done by performing a scan. However, full passive solutions would need to rely on tricks presented in BlueSniff. Moreover, AFH (since BT 1.2) can make the task harder, since a passive eavesdropper would have to follow the channel map to be able to synchronize to the piconet. Secondly, the channel hopping timing requirements are hard to meet by a SDR-based solution. To the best of our knowledge, this is still the case in 2020. Haataja [3] presented a good overview of Bluetooth threats and possible countermeasures. In 2014, Huang *et al.* proposed BlueID [5] that can recognize Bluetooth masters based on the skew of their hopping clock. They used an Ubertooth to record traffic on one arbitrary chosen Bluetooth channel, which is enough to capture a representative portion of the traffic, and estimate clock skew for surrounding Bluetooth piconets. They showed that the clock skew was distinctive enough to accurately classify 56 different devices.

D. Wireless devices fingerprinting

Wireless device fingerprinting is a collection of techniques to identify wireless transmitters, based on subtle differences in their handling of the communication scheme. Those differences, called features, are typically undetectable for the upper layers, as long as they stay within the specification error tolerance, but can be seen by carefully observing the lower layer behavior. Fingerprints might be collected at the physical, MAC, or sometimes application layer [16]. In this study, we exclusively focus on a physical layer fingerprinting. There are two main approaches for physical layer fingerprinting: in the waveform domain, or in the modulation domain. The waveform approach is focused on the signal itself as a collection of samples over time. For instance, the magnitude of the turn-up transient is known to be highly distinctive, even among devices of the same model [4]. However, as it occurs in a very short lapse of time, it must be sampled at very high rates to provide satisfying results. Theoretically, fingerprinting in the waveform domain can be performed independently of the modulation scheme, making it easier to adapt to any modulation. However, it can be improved by taking advantage of the knowledge of the protocol. Reising *et al.* performed waveform analysis on the phase trace of GMSK bursts, in the mid-amble region [9]. Methods in the modulation domain are taking advantage of the knowledge of the modulation. They work by comparing the received packet with an ideal simulated packet. They have been used with a variety of protocol, such as Wi-Fi [1] or GSM [17]. They simplify the collection of relevant features with smaller dimensionalities, which is known to be more resistant to noise than waveform based approaches. On the other hand, they are more complicated to implement as a full demodulator is needed, and they are less effective to differentiate different devices of the same model.

III. Bluetooth ID packets fingerprinting

In this Section, we evaluate indicators to differentiate between different Bluetooth-enabled devices, based on the individual properties of their physical layer.

A. Experimental setup

We chose the Bluetooth Inquiry procedure to evaluate the collection of physical layer fingerprints. During an Inquiry

procedure, the scanning (*master*[1]) device sends many *ID packets* with the predefined *General Inquiry Access Code* (GIAC) while hopping on different channels. The *Access Code* is the only variable field in an ID packet. Thus, all ID packets sent as parts of an inquiry are made of the same bits. This makes a good test case for physical layer fingerprinting, where all the payloads are the same, and the only observable differences can come from the physical layer.

We collect Bluetooth packets using a LimeSDR mini, with a sample rate of 28 MHz, to collect a significant portion of the Bluetooth channels (channels 5 to 32 = 28/79 BT channels). The center frequency was set to 2420 MHz to avoid nearby WiFi channels and cover only the BT spectrum.

The tested devices were placed at four meters from the receiver antenna. An inquiry procedure has been triggered fo each device (by exerting the Bluetooth scan functionality), while collecting samples. We performed three different captures for each device.

During a BT inquiry procedure, a master sends the same inquiry packet at a constant rate. Instead of analyzing single packets, we can use series of such packets to assess the features with more accuracy. To analyze series of packets, we must first ensure that they were all emitted by the same device. It is not a trivial matter as it relates to the problem we are trying to address in the first place.

Fortunately this can be achieved quite easily during a normal inquiry procedure. Firstly, packets sent over a Bluetooth piconet can easily be clustered together as they are all synchronized to the master clock. Secondly, the inquiry procedure is a particularly ideal situation for clustering: only two devices can talk. The master sending packets with the GIAC Access Code, and eventually a slave replying with its own Lower Address Part (LAP). After the connection is established, all subsequent packets will be prefixed with the current connection master LAP. Therefore we can consider that all packets received on a single piconet, and containing a GIAC are sent by the same transmitter.

B. Features description

1) Preamble duration: A synchronization preamble is usually a sequence of alternated bits preceding a packet to perform the clock synchronization at the receiver side. Many wireless protocols are requiring the presence of such preambles, and Bluetooth makes no exception. The number of synchronization bits is fixed to four in the BT specification[2]. However, we observe that the exact number of bits transmitted varies across different devices manufacturers. A small delay exists between the start of transmission, denoted by the amplitude rising, and the actual sending of the modulated data[3].

We observe that the Bluetooth devices we tested exhibits different behaviors during the preamble. The number and quality of synchronization bits is not always the same. The delay between the amplitude rise and the first synchronization bit also varies. To take into account for both the start-up period, and synchronization bits count, we call *preamble duration*, the interval of time between the amplitude rise denoting the start of a packet (Start peak on Figure 1), and the actual position of the *access code* in the demodulated trace (AC pos on Figure 1).

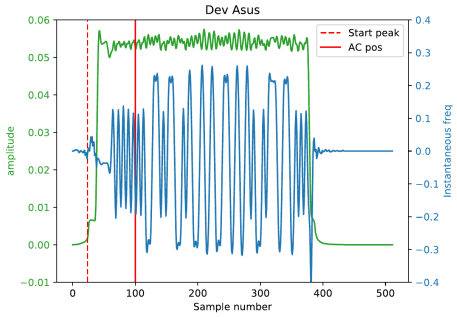

Fig. 1. Averaged amplitude and frequency traces of 1464 inquiry packets (Asus BT-400)

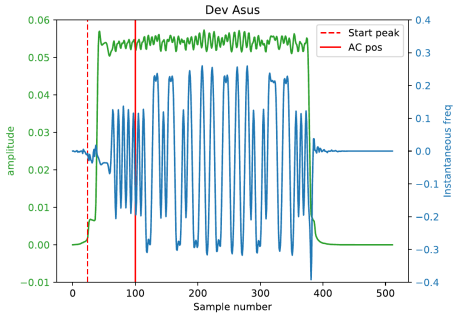

Fig. 2. Another scan with 1734 packets (same Asus BT-400)

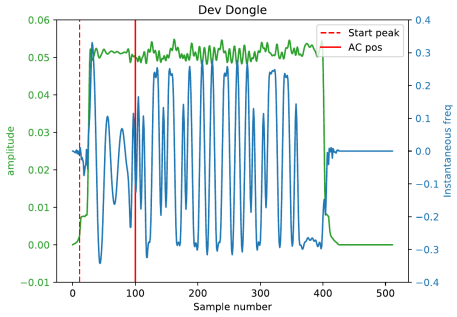

Fig. 3. Averaged amplitude and frequency traces of 1082 inquiry packets (Unbranded dongle)

Figures 1 and 3 clearly exhibit that preamble duration is different for different devices. On contrary, similar preambles can be shown on Figures 1 and 2. The preamble duration is a good candidate for fingerprinting.

2) Hopping clock skew (HCS): Bluetooth transmissions are following a hopping scheme based on the master clock. All

[1]By convention we call *master* the device performing an inquiry

[2]Bluetooth Core v5, Vol 2, Part B, 6.3.2: Preamble

[3]This small period itself is extensively studied in the transient fingerprinting literature

transmissions are supposed to begin on a *time slot* boundary. The hopping frequency can be 1600 or 3200 hops per seconds. As in [5], we measured the hopping clock skew, comparing the arrival time of two consecutive ID packets. To refine the measurements, we use the *access code time* (AC pos in Figure 1), which is obtained from the demodulated trace, and is more noise resistant than the peak detection. All packets must be aligned on the duration of a time slot, $Hdur = \frac{1}{3200}$. $Tcur$ and $Tprev$ are the time of arrival of the current and previous packets. The expected delay $Texp$ in the interval rounded to the nearest time slot boundary ($Hdur$) is equal to:

$$Texp = Hdur \times \left\lfloor \frac{Tcur - Tprev}{Hdur} \right\rceil.$$

HCS is expressed as:

$$HCSppm = 1\text{M} \times \left(\frac{Texp}{Tcur - Tprev} - 1 \right).$$

in parts per million (ppm[4]),

When operating on offline captures, where samples could have been lost, a precise time reference is needed. Sometimes, aberrant HCS between two packets are observed. To eliminate those packets, a threshold equal to 100 ppm has been set. Values exceeding this threshold are thrown away . During the experiments, this problem rarely arises (only 7 measurements over 47184). This correction is not required when capturing live samples from a SDR receiver offering proper timestamps.

In our experiments, the Bluetooth packets are resampled at a 4 Msps rate, to store four samples per symbol as in [13]. For one packet pair, we obtain one measurement with a resolution of $250\text{n}s$. To enhance the accuracy, the data are averaged on several measurements.

3) Carrier clock skew (CCS): The BT protocol exploits 79 channels, each 1 MHz wide. The offset between the center frequency of a packet and the expected center of the channel is computed and converted into ppm to obtain the *carrier clock skew* (CCS).

$$\Delta F = \frac{4\text{M}}{2\pi} \times (\overline{IF} - \overline{idealIF}),$$

$$CCSppm = 1\text{M} \times \frac{\Delta F}{ChannelFreq}.$$

IF are the frequency-demodulated samples in radians per seconds, and idealIF is the locally-generated frequency trace of a GIAC.

The CCS take into account for all the samples composing the access code. The estimation is more precise as we have a large number of samples to work with in each packet.

We observe that almost all of the device models we tested are exhibiting a consistent frequency offset, ranging from ± 22 KHz around the expected center frequency. In our experiments, the measured offset stayed invariant across several measurements for each device. This consistency has been already reported in for the HCS in [5]. Section VI exhibits the effect of the temperature on the measured frequency.

[4]Clocks drift are usually measured in parts per million or billion

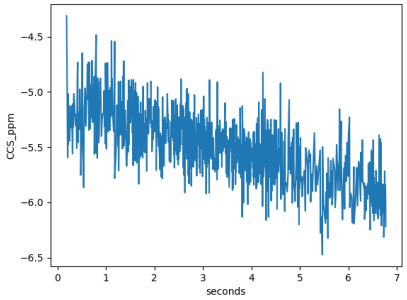

Fig. 4. CCS of Unbranded Dongle over a complete scan

Only one device showed variability in the CCS and HCS measurements. In experiments on the unbranded low cost BT USB dongle, we observed that the CCS and HCS are consistently varying over time (as shown in figures 4 and 5). For this device the clock variation rate itself is a distinctive feature. This is not very surprising since we are looking for imperfections, and the low-end devices are naturally more prone to it.

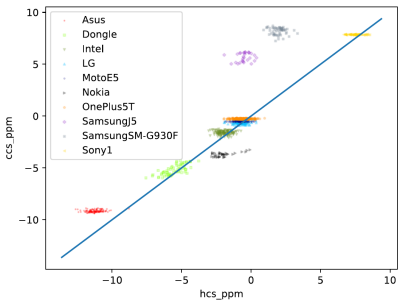

Fig. 5. HCS vs CCS on 50 packets sets.

IV. Analysis

In this Section, the observations we made while analyzing the dataset are discussed. For some devices, HCS and CCS seem to have the same value. We also show that the preamble duration allows a discrimination of the device manufacturers. Finally, the use of a very large number of packets to create high SNR features is discussed.

A. Clock skews relation

Figure 5 exhibits the Hopping clock skew versus Carrier clock skew for ten devices. The blue line represents the identity. We can see that for the majority of devices, the HCS and the CCS are closely correlated. For the two Samsung devices we tested (Exynos based), the two clock skews are clearly unrelated, with a high CCS of around 5 ppm, and a lower HCS of ± 0.5 ppm.

The devices with a HCS close to the blue line have their BT peripheral governed by a single oscillator. It is involved in the generation of both the carrier frequency and the hopping

timer. In other designs, the hopping timer can be generated by the CPU or based on a different oscillator. Further close up analysises of the device architectures are needed to confirm or infirm this hypothesis.

B. Preamble duration and manufacturer

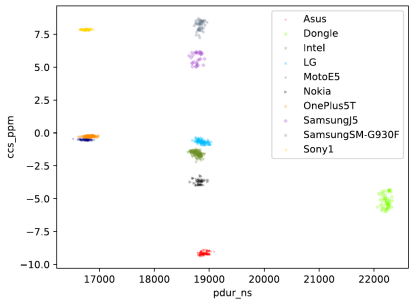

Fig. 6. Preamble duration vs CCS on 50 packets sets

Figure 6 represents the preamble duration on the x-axis, and the CCS on the y-axis. The preamble durations are grouped within two main clusters: one around 16.5 μs, and one around 18.7 μs. The unbranded dongle is the only one which exhibits a preamble duration of around 22 μs. We must point out that all Qualcomm based phones are in the left cluster, while all phones with a Broadcom device are on the middle cluster.

C. High SNR pictures

Figures 1 to 3 were obtained by averaging traces of all packets collected during the inquiry procedures performed by an Asus BT-400 adapter and an unbranded USB dongle. To compute the amplitude traces, we configured our filtering tool to output raw samples in addition to the frequency trace.

Averaging over a very large number of packets dramatically enhances the SNR, and results in highly distinctive features for individual device model. For instance, we can see in Figure 3 that the unbranded dongle sends a quite idiosyncratic preamble, and totally drops the trailing bits.

In our classification experiments, we did not use these massively averaged traces, because they are more expensive to produce, and provide few examples of high dimensionality, which would complicate the training process.

Though, this method is helpful when exploiting more distinctive features. For instance, when looking at the three high SNR picture of each device, we noticed that the magnitude vector alone is distinctive of each device model, as shown in Figures 1 to 3.

V. Classification

In the previous Section, we have showed that it is possible to identify the devices by looking at the features. However, in a real IDS, that task should be automated. We have set an automatic classification experiment using the scikit-learn python library [2]. The same dataset, with three captures of a full Inquiry procedure per device, has been used. A K-fold cross validation is used with K=5. We tested several classifiers: logistic regression, multi-layer perceptron (MLP), random-forests (RF), and support-vector classifier (SVC). At the end, RF and SVC yield comparable results, outperforming the other classifiers on our dataset. The configuration achieving the best result was the SVC with $C = 250$ and $gamma = 0.33$.

We measured the SVC performances on the same dataset, while varying the number of packets used for the feature estimation. For each run, we randomly select 1500 packets of a single capture. The packets are divided in evenly sized sets, and one feature vector is computed for each set. Therefore for bigger set sizes, we have less examples to train the classifier.

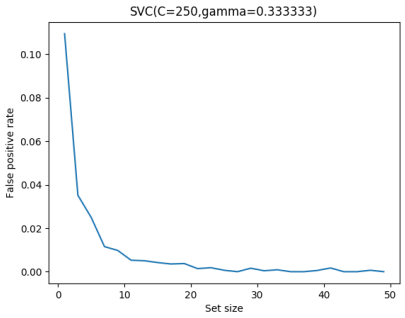

Fig. 7. SVC classifier FPR versus the set size

The classifier delivers a better classification while increasing the set sizes and achieves a more than 99.8% accuracy for all the sets containing more than twenty packets. As a full inquiry procedure typically produces thousands of packets, we are confident that capturing even a subset of a single scan would provide enough data to precisely characterize its transmitter model.

VI. Temperature effect on fingerprinting

In the previous classification experiments, we took advantage of the frequency offsets of the selected devices as discriminating features. The temperature of a crystal oscillator is known to influence its actual resonant frequency [5]. We assessed the effect of temperature on one device of the dataset.

A. Experimental setup

In [5], the room temperature was changed. We chose to modify only the temperature of the device to get a clear understanding of the phenomena. A thermal plate has been used to modify the temperature of the device ; a thermal probe attached to the back of the device has been used to accurately measure its temperature. For each measurement, an inquiry procedure is initiated and all messages containing a GIAC are recorded. Each dot represents the average HCS of all packets of a single inquiry.

In the observed range, the frequency linearly decreases with the temperature increase. This is not surprising since crystal oscillators have a lowest temperature dependance around the standard ambient temperature, which is usually $25°\mathrm{C}$.

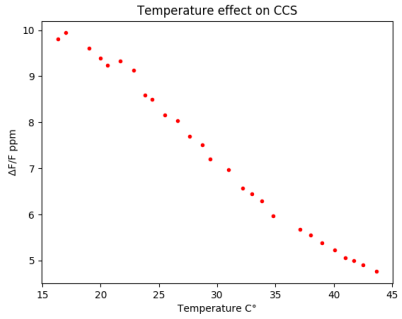

Fig. 8. Temperature effect on the frequency of a Sony phone

B. Lessons learned

We observed that for the selected device (*Sony1*), the CCS and the HCS stay the same across all the experiment. Our hypothesis of a common oscillator to generate both the carrier frequency and the hopping clock is strengthened.

Regarding the fingerprint process, the temperature has a strong effect on the measured frequency offsets (around 5 ppm on a $30°\mathrm{C}$ range, for an absolute error of 8 ppm at $25°\mathrm{C}$). The measured difference is higher than reported in [5] (around $+0.5$ ppm when cooling the room temperature by around $30°\mathrm{C}$). This important difference in the results can be explained by the different approach we took in our measurements, cooling the device itself instead of the room temperature. We presume that in BlueID experiments, the devices temperature was not as low as the room temperature, possibly because of heat dissipated by the device itself. According to our results, the frequency shift measured might be way higher, for instance when turning on a device that has been kept off in a cold place long enough.

A proper fingerprinting solution should use physical models to take into account the temperature effect. BlueID proposed to track the frequency drift across time, and update the model if its absolute value does not exceed a threshold. This approach could be improved by checking that the frequency drift is consistent with the temperature profile of the device under observation. For instance, an increase in the temperature should cause a negative frequency drift for the Sony phone in a tempered environment.

This experiment also shows that frequency based wireless fingerprinting solutions could be attacked at lower costs than expected. It is well known that simple radiometric features could be mimicked by an attacker holding a carefully programmed SDR. Modifying the temperature of an off-the shelf transmitter might be a cheaper way to forge a frequency offset.

VII. Conclusion & Future works

In this paper, a low cost software defined radio has been used to experiment on the Bluetooth Inquiry procedure. The relevance of the gathered data has been demonstrated, presenting a fingerprinting approach that can distinguish the model of a Bluetooth transmitter with a high accuracy ($> 99.8\%$), using only three features at the physical layer.

Although accurate, this method needs to be enhanced, and further tested with a broader set of equipments. We plan to run a larger measure campaign to validate that the features presented in this paper are enough to distinguish devices with a high confidence. We will also experiment our fingerprinting technique on multiple devices of the same model. We expect that it will be harder to distinguish several instances of the same devices and will look for new features that will allow us to do so.

Acknowledgment

This work has been partially funded by IRCICA (Univ. Lille, CNRS, USR 3380 IRCICA, Lille, France).

References

[1] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless device identification with radiometric signatures. *Proceedings of MOBICOM*, pages 116–127, 2008.
[2] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
[3] K. Haataja. *Security Threats and Countermeasures in Bluetooth-Enabled Systems*. PhD thesis, 2009.
[4] J. Hall, M. Barbeau, and E. Kranakis. Detecting Rogue Devices in Bluetooth Networks using Radio Frequency Fingerprinting. *ICCCN*, 2006.
[5] J. Huang, W. Albazrqaoe, and G. Xing. BlueID: A practical system for bluetooth device identification. In *INFOCOM*. IEEE, April 2014.
[6] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. DDoS in the IoT: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.
[7] D. Mantz, J. Classen, M. Schulz, and M. Hollick. Internalblue - bluetooth binary patching and experimentation framework. In *Proceedings of MobiSys*, pages 79–90, New York, NY, USA, 2019. ACM.
[8] Project Zero. Project Zero: Over The Air: Exploiting Broadcom's Wi-Fi Stack (Part 1), 2017.
[9] Donald R. Reising, Michael a. Temple, and Michael J. Mendenhall. Improved wireless security for GMSK-based devices using RF fingerprinting. *Electronic Security and Digital Forensics*, 3(1):41, 2010.
[10] M. Schulz, D. Wegemer, and M. Hollick. Nexmon: The c-based firmware patching framework, 2017.
[11] B. Seri and G. Vishnepolsky. BlueBorne Technical White Paper-1, 2017.
[12] Bluetooth SIG. Specification of the bluetooth system, v5.0. Technical report, December 2016.
[13] D. Spill and A. Bittau. BlueSniff: Eve meets Alice and Bluetooth. *USENIX WOOT*, page 10, 2007.
[14] R.-P. Weinmann. Baseband Attacks: Remote Exploitation of Memory Corruptions in Cellular Protocol Stacks. *USENIX WOOT*, pages 1–10, 2012.
[15] H. Welte, S. Munaut, and A. Eversberg. Osmocombb.
[16] Q. Xu, R. Zheng, W. Saad, and Z. Han. Device fingerprinting in wireless networks: Challenges and opportunities. *IEEE Communications Surveys and Tutorials*, 18(1):94–104, 2016.
[17] Z. Zhuang, X. Ji, T. Zhang, J. Zhang, W. Xu, Z. Li, and Y. Liu. Fbsleuth: Fake base station forensics via radio frequency fingerprinting. In *Proceedings of ASIACCS*, pages 261–272, New York, NY, USA, 2018. ACM.