Valkyrie: A Generic Framework for Verifying Privacy Provisions in Wireless Networks

Guillaume Celosia
Univ Lyon, INSA Lyon, Inria, CITI
F-69621 Villeurbanne, France
guillaume.celosia@insa-lyon.fr

Mathieu Cunche
Univ Lyon, INSA Lyon, Inria, CITI
F-69621 Villeurbanne, France
mathieu.cunche@insa-lyon.fr

ABSTRACT

Wireless communications integrated in connected devices can expose their users to tracking via the exposure of link layer identifiers (e.g. MAC addresses). To counter this threat, it has been proposed to replace those permanent identifiers with periodically changing random pseudonyms [17]. This practice, called *address randomization* has been progressively adopted by vendors [28, 36] and has even made its way to wireless standards [1, 35]. However, an effective implementation of address randomization requires more than periodically rotating the link layer identifier. Indeed, several works [8, 11, 12, 16, 27, 28, 36] identified issues with address randomization implementation, where in-frames counters and identifiers can undermine the anti-tracking measure.

In this paper, we address the problem of verifying the correctness of an address randomization implementation. To this end, we introduce an approach to identify issues based on a capture of the traffic generated by a device. This approach relies on rules specifying requirements for a correct implementation of address randomization. Then, we prototype *Valkyrie* (*Verification of Addresses LinKabilitY in address Randomization ImplemEntations*), a software tool that, based on a set of rules, verifies that a given sequence of frames generated by a device does not compromise the address randomization scheme. Finally, we evaluate this tool on a corpus of frame captures corresponding to 60 devices implementing address randomization for Wi-Fi and Bluetooth Low Energy (BLE).

CCS CONCEPTS

• **Networks** → **Network privacy and anonymity**; • **Security and privacy** → *Mobile and wireless security*.

KEYWORDS

Privacy; Internet of Things; Tracking; Address randomization.

ACM Reference Format:
Guillaume Celosia and Mathieu Cunche. 2020. Valkyrie: A Generic Framework for Verifying Privacy Provisions in Wireless Networks. In *13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '20), July 8–10, 2020, Linz (Virtual Event), Austria.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3395351.3399340

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WiSec '20, July 8–10, 2020, Linz (Virtual Event), Austria
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8006-5/20/07…$15.00
https://doi.org/10.1145/3395351.3399340

1 INTRODUCTION

Connected devices are found in many applications and domains from healthcare, quantified self, entertainment as well as end-devices such as smartphones, tablets and laptop computers. All those devices rely on wireless technologies such as Wi-Fi or Bluetooth to communicate. As a result, in their daily lives, users are carrying wireless-enabled devices. Because of the ever active discovery mechanisms, those devices periodically emit messages that include identifiers such as *MAC addresses* for Wi-Fi and *device addresses* for Bluetooth. Those identifiers can be passively collected and leveraged to track users in the physical world [10, 30, 33]. Wireless-based physical tracking has found diverse applications such as customers analytics in shops [14, 22], commuters monitoring [31] and urban planning [25] to name a few.

In response to growing privacy concerns, it has been proposed to replace those permanent identifiers with periodically changing random pseudonyms [17]. This practice, called *address randomization*, has been adopted by vendors [28, 36] and has even made its way to wireless standards [1, 35]. Address randomization has become a default feature included in mobile operating systems (OS) [3, 4] and that can be found in many devices [11].

The adoption of address randomization, has led to the discovery of several issues with its implementation. Studies showed that other elements of the frame can defeat the randomization scheme and thus undermine the privacy protection. Those implementation issues are mainly coming from counters and identifiers that are not rotated with the device address [8, 11, 12, 16, 27, 28, 36]. For instance, in Wi-Fi, the evolution of the `Sequence Number`, a counter incremented at each frame, is not always modified when the address is changed. Thus, it is clear that protection against tracking requires more than just periodically rotating the link layer of the device.

To improve the effectiveness of address randomization implementation, we focus on the problem of verifying the correctness of those implementations. More specifically, the objective is to check whether an implementation is affected by one or several issues.

Then, we consolidate the properties required by address randomization that have been produced in recent research efforts. To this end, we define a framework to automatically verify those properties based on a network capture.

In this paper, we present the first approach at automatically verifying implementation of address randomization schemes. Our contributions are outlined as follows:

- We formalize the concept of frame unlinkability that is the objective sought by address randomization (Section 3);
- We present the design and implementation of *Valkyrie* (*Verification of Addresses LinKabilitY in address Randomization*

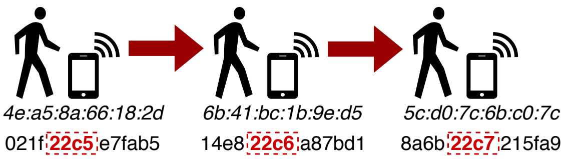

Figure 1: Example of the device address linking via a non-reset counter field. The device is randomizing its address (in italic) over time while incrementing a counter (in bold) in the broadcasted data. Such a 2-byte long counter can be leveraged by a passive attacker to link together frames generated with the three different device addresses.

ImplemEntations), a framework to automatically verify privacy properties based on a network capture (Section 4);
- We evaluate *Valkyrie* using a representative set of Wi-Fi and Bluetooth Low Energy (BLE) devices (Section 5).

Finally, we discuss related work in Section 6, and give concluding remarks in Section 7.

2 ADDRESS RANDOMIZATION AND ITS LIMITATIONS

Following the appearance of wireless tracking [2, 10, 18, 26, 30, 37], address randomization has been introduced to protect users' privacy. Address randomization idea is to replace the link layer identifier[1] with a temporary and random one. This countermeasure denies the link layer identifier to be used as a reliable element for tracking.

In Wi-Fi, address randomization has been adopted in various OS, such as iOS, Android, Windows and Linux, and only very recently in the 802.11 standard [1]. In Bluetooth, address randomization was introduced in 2010 through the version 4.0 of the standard [34, Vol 3, Part C, sec. 10.7], and recent works suggest that address randomization is included in a significant part of BLE devices [8, 11].

Despite a large adoption of this anti-tracking measure, several works showed that using a rotating link layer identifier is not enough to prevent tracking. In particular, the remaining of the frame may include other unique identifiers or artifacts that can be used for fingerprinting or tracking a device over address changes [28, 36].

For instance, a counter (`Sequence Number`) included in 802.11 frames was not reset upon the address change in the early randomization implementation in iOS [16]. Thus, it was possible to link together two consecutive address fields just by observing the increasing values of the `Sequence Number` field (see Figure 1).

Another example is found in BLE where an identifier (`Auth Tag`), included in `advertisement packets` of *Apple* devices, sometimes overlaps two consecutive addresses used by a device [12]. As such, this field can be also leveraged to link together distinct addresses of a device.

As a consequence, even if the link layer identifier is correctly rotated, overlooked elements and implementation errors can undermine the privacy protection.

[1]In general, a globally unique identifier.

3 PRIVACY PROPERTIES OF NETWORK TRAFFIC

In this section, we discuss properties necessary to prevent tracking in face of a passive attacker.

3.1 Frame unlinkability

The objective of measures such as address randomization is to avoid an observer from tracking a device over an extended period. We argue that, to achieve this objective, it is mandatory to prevent the attacker from linking together frames generated by a single device.

Frame linking can be done based on their content [28, 36], timing [29] or even their properties at the physical layer [38]. In this paper, we only focus on the frame content, because the two other approaches are less reliable [29] or require specialized hardware [38].

In the context of wireless traffic, unlinkability [32] of frames means that the attacker cannot distinguish whether they are related or not. This indistinguishability can be expressed as follows:

$$P(f_1 \sim f_2) = P(f_1 \nsim f_2) = 1/2$$

where $f_1 \sim f_2$ means that f_1 and f_2 are related and $f_1 \nsim f_2$ means that they are not.

Let us consider that those frames are composed of n fields $\{h_i\}_{1 \le i \le n}$. Assuming that values of those in-frame fields are independent[2], unlinkability at the frame level and at the field level is equivalent:

$$P(f_1 \sim f_2) = P(f1 \nsim f2) \Leftrightarrow \bigwedge_{1 \le i \le n} P(f_1.h_i \sim f_2.h_i) = P(f_1.h_i \nsim f_2.h_i)$$

To enforce unlinkability of frames, it is thus sufficient to ensure that fields are unlinkable. In other words, if for each field h_i, the value of $f_1.h_i$ is unlinkable with $f_2.h_i$ then f_1 and f_2 are unlinkable.

3.2 Empirical unlinkability properties

The above mentioned properties can be used as a design help, but are not suitable for an empirical verification that would be performed on a sequence of frames. Indeed, the evaluation of the probabilities will be limited by practical constraints such as the duration of the observation and the frequency of identifier rotation. Therefore, we derived properties that can be applied to a sequence of limited size.

Let us consider a device d generating a sequence of frames f_i, each frame including a link layer identifier $f_i.addr$ as well as a set of n fields $\{f_i.h_j\}_{1 \le j \le n}$.

For any two consecutive frames f_{i_1} and f_{i_2} for which $f_{i_1}.addr \neq f_{i_2}.addr$ (link layer identifier rotation), the fields $\{h_j\}_{1 \le j \le n}$ of f_{i_1} and f_{i_2} must satisfy the following:

(1) if h_j is an identifier or a data field: $f_{i_1}.h_j \neq f_{i_2}.h_j$
(2) if h_j is a counter field modulo m: $d_m(f_{i_1}.h_j, f_{i_2}.h_j) > \delta$

where $d_m(x, y) = x - y$ if $x > y$ and $x + m - y$ otherwise, measures the distance between two values modulo m.

In the following, we will employ those empirical properties to identify issues in address randomization implementations.

[2]This is usually the case in Wi-Fi and BLE discovery traffic.

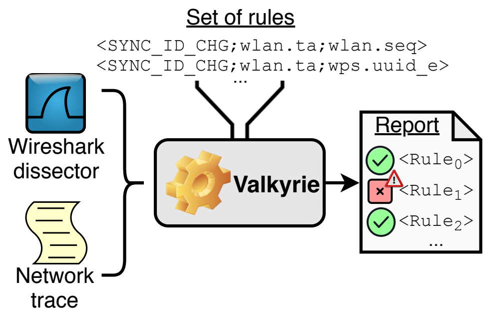

Figure 2: Functional diagram of *Valkyrie*. A network trace along with a set of rules are provided as inputs to the tool. The *Wireshark* dissector is leveraged for the protocol field denomination that must be specified in rules. At the end of the analysis, *Valkyrie* outputs a report specifying verified rules and breached ones with detailed warning messages.

4 DESIGN AND IMPLEMENTATION

In this section, we present the design of *Valkyrie* (*Verification of Addresses LinKabilitY in address Randomization ImplemEntations*), a software tool that can verify the enforcement of privacy properties on (wireless) network traffic traces. As inputs, *Valkyrie* takes a network traffic trace generated by a device as well as a set of rules to be checked. Then, it verifies those rules independently and produces a set of warning messages for each breached rule (see Figure 2). The code is available online[3].

4.1 Rules syntax

To specify rules presented in Section 3, we designed a custom syntax. A rule specifies the link layer field that is rotating and upon which a property must be enforced: this field is called `address` in our syntax. Then, the rule needs to specify the field that must satisfy the property: this field is called the `target`. Each rule is also associated with a type, noted `type`, which defines the type of property that needs to be satisfied. Currently, our tool includes two rule types: `SYNC_CNT_CHG` and `SYNC_ID_CHG`, which respectively cover the counter and identifier/data properties. Optional parameters can be appended to those rules: for instance, the distance δ in the case of the `SYNC_CNT_CHG` rule. Finally, a rule has the following form:

`type, address, target` <, `optional parameters` >

Valkyrie leverages on `pyshark` [24], a python wrapper for `tshark` (the command line version of *Wireshark* [13]), for the naming of frames and fields. This means that the protocol field denomination used in the rules corresponds to the *Wireshark* one. Thus, the tool can be applied to any of the *Wireshark* supported protocols, and even more by using *dissectors* which are frame parsers that can be written for any protocol. In this study, we wrote our custom dissector[4] to parse *Apple* and *Microsoft* BLE messages (see Section 5.3).

This syntax is then used to translate formal rules defined in Section 3 into practical ones. For instance, in the case of 802.11, the counter rule applied to the `Sequence Number` can be written as:

< `SYNC_CNT_CHG; wlan.ta; wlan.seq` >

[3] https://github.com/celosiag/valkyrie
[4] https://github.com/celosiag/joker

where `wlan.ta` designates the transmitter address of the device and `wlan.seq`, the `Sequence Number`.

4.2 Verification process

Given a network trace along with a set of rules, *Valkyrie* will verify that those rules are satisfied. Algorithm 1 describes this process which, for each rule, is performed as follows: for each consecutive frames f_1 and f_2 having distinct `address`, verify that f_1.`target` $\neq$ f_2.`target` in the case of `SYNC_ID_CHG`, and $d(f_1$.`target`, f_2.`target`$)$ $> \delta$ in the case of `SYNC_CNT_CHG`. To compute the distance between two values of a counter field, we consider the counter is *looping*, i.e. will go back down to zero after having reached the maximum value. Thus, the distance can be computed as presented in Section 3.2.

Input: - Set of n rules $R = \{r_i\}_{0 \le i < n}$
- Network trace T composed of frames f_i

Output: Boolean vector V whose element $V[i]$ describes the satisfaction of rule r_i

foreach $r_i \in R$ **do**
  $V[i] = false$;
  foreach f_1 ***and*** $f_2 \in T$ **do**
    if f_1.`address` != f_2.`address` **then**
      if (r_i.`type` == `SYNC_ID_CHG` ***and*** f_1.`target` != f_2.`target`) ***or*** (r_i.`type` == `SYNC_CNT_CHG` ***and*** $d(f_1$.`target`, f_2.`target`$) > \delta$) **then**
        $V[i] = true$;
      end
    end
  end
end

Algorithm 1: Verification algorithm of *Valkyrie*.

4.3 Address reuse detection

In addition to those properties on the frame fields, *Valkyrie* also verifies that device addresses are not reused. More specifically, once used during a time interval, an address should not be reused later in order not to lead a passive eavesdropper to trivially link distinct frames broadcasted by the device. To this purpose, we provided *Valkyrie* with a feature that is able to detect address reuse by recording addresses appearing within a trace.

5 EXPERIMENTAL EVALUATION

In this section, we perform an evaluation of *Valkyrie* based on wireless traffic generated by real-world devices. To this end, we focus on two prominent Internet of Things supported wireless technologies implementing address randomization: Wi-Fi and BLE.

5.1 Tested devices

The evaluation is based on a set of 60 devices, equipped with a Wi-Fi and/or a BLE interface, that can be categorized into three types: laptop, smartwatch and smartphone. This set covers major manufacturers such as *Apple*, *Google* and *Motorola*. Some smartphones are tested with different OS versions. For instance, the *Apple* iPhone XR has been evaluated with iOS versions 12.1.2 and 12.4.1, while Android 7.1 and 9 have been experimented with the *Google* Pixel XL. Table 2 details the full list of tested devices that constitutes a representative sample of devices used in the world. Note that, all those devices are owned by the researchers or their institutions.

5.2 Traffic capture protocol

For each device, a traffic capture was obtained by isolating the device in a Faraday cage, and was then stored in `pcap` format. This rules out the possibility that devices were connected to another device or an access point. Thus, they only generated discovery traffic: `probe requests` for Wi-Fi and `advertisement packets` for BLE. Moreover, during captures, devices were left untouched with their wireless interface (Wi-Fi or BLE) enabled. Note that, each capture lasts 20 minutes or gathers 200 frames, whichever is first.

5.3 Rules specifications

The verification process is based on a set of rules. Leveraging the language designed in Section 4.1, Table 1 specifies five rules corresponding to the five main issues affecting address randomization according to the literature (see Section 2). The three first rules cover issues related to identifiers in the frame body such as the `WPS UUID` field in Wi-Fi (①), and the `Auth Tag` and `Device Hash` respectively found in *Apple* Nearby Info (②) and *Microsoft* CDP (③) BLE messages. The last two rules cover predictable fields, namely `Sequence Numbers` in Wi-Fi (④) and `IV` in *Apple* Handoff BLE messages (⑤).

5.4 Results

For each capture, we ran *Valkyrie* loaded with the rules set corresponding to the wireless technology: rules ① and ④ for Wi-Fi, and ②, ③ and ⑤ for BLE. For each rule, Table 2 gathers raised issues. Note that, an issue is raised if the rule is unsatisfied at least once.

A first observation is that all devices are affected by at least one issue, and that more than 73% are affected by two or more.

In Wi-Fi, the most prevalent issue is the non-reset `Sequence Number` (④), which affects 98.3% of devices. Results on smartphones experimented with different versions of their OS such as *Apple* iPhone 5S and *Google* Pixel XL show that software updates hampered tracking based on the `Sequence Number`. However, although this issue was supposed to be corrected in version 8 of Android [21], some devices running this OS version such as *Huawei* P20 Lite and *Sony* Xperia XZ1 are still affected. In [28], Martin et al. already identified this address randomization misimplementation that seems to be manufacturer related. Finally, 8.3% of tested devices are prone to the static `WPS UUID` issue (①).

In BLE, all *Apple* devices except the *Apple* MacBook Pro laptop match with corresponding rules ② and ⑤. In fact, the *Apple* MacBook Pro is not affected by rule ② as it does not contain any `Auth Tag` in its emitted *Apple* Nearby Info BLE messages. Similarly, rule ③ is only raising an issue with the *Dell* G3 laptop broadcasting *Microsoft* CDP frames, which is the only device running Windows. As a result, *Valkyrie* verified expected non-reset counter and static identifier concerns in which all *Apple* and *Microsoft* benched BLE devices expose their owners to tracking.

Lastly, *Valkyrie* detected that 45% of devices reuse random device addresses, especially smartphones of manufacturers *Apple*, *ASUS*, *Blackberry*, *HTC*, *Huawei*, *LG*, *Motorola*, *Sony*, *Xiaomi* and *ZTE* (see Table 2). De facto, it is unclear why a device reuses an address. Possible explanations include: poor PRNGs used for address generation, or a switch to a static address of the device.

Note that, given the limited length (20 minutes) of the capture, results may include false negatives: some devices might be breaking one of the rules, but the capture was not long enough to capture this behavior. For instance, the `Sequence Number` issue (④) was not found in the capture from the *Apple* iPhone 5S running iOS 11.2.1 while it appears not to have been fixed until iOS 13.1 at least.

5.5 Evaluation summary

To put in a nutshell, the evaluation demonstrates that the current implementation of *Valkyrie* is usable and allows to detect most privacy-threatening behavior such as non-reset counters and static identifiers. Furthermore, the proposed rule specification language was flexible enough to express associated requirements.

6 RELATED WORK

Identification of privacy threats in wireless traffic has been the subject of many research works. In particular, a number of those works focused on tracking but also on weaknesses of address randomization schemes in Wi-Fi [16, 28, 36] and Bluetooth [8, 11, 15, 27]. Our contribution capitalizes on those works and provides a way to automatize the detection of known issues in wireless traffic.

Several works have considered automated verification of system properties [7, 9, 19, 23]. As our approach, they rely on passive testing techniques to check the conformance of a system with regards to its specifications. However, those approaches are oriented toward system rather than network and do not focus on privacy properties.

Using formal methods, Arapinis et al. analyzed [5] security properties of 3G protocols. Especially, they exposed two privacy threats aiming to trace and identify mobile telephony subscribers, and proposed fixes satisfying the unlinkability and anonymity properties.

In [6], Barnes et al. used an emulated environment to verify the binary implementation compliance of network stack. Leveraging this framework, they are able to verify protocol properties declared through a formal language. Another implementation verification was presented in [20] where an implementation extracted model was then checked using formal methods. In our case, a binary or a model of the implementation is not readily available and emulation environment would be difficult to setup.

7 CONCLUSION

This work presented the first attempt at automatically verifying the correctness of address randomization implementation. To this purpose, we discussed requirements for protecting users against tracking and derived a list of properties leveraging works done by the community. Then, we prototyped *Valkyrie*, a versatile tool able to verify properties written in a *Wireshark* based language. We showed that properties associated with main issues found within address randomization can be expressed using this language. Finally, relying on a representative set of Wi-Fi and BLE enabled devices, we evaluated the proposed tool demonstrating that *Valkyrie* was able to detect issues in the generated wireless traffic.

As such, the developed approach can be applied by vendors to verify that privacy properties are enforced by their devices. In addition, this approach can be included as a part of a certification process to verify that some devices are meeting privacy requirements. Finally, this approach can be adapted to any protocol, provided that it is supported by *Wireshark* or that a dissector exists.

Table 1: List of specified rules for the experimental evaluation.

Tracking source	Benched element	Rule (○ : applied to Wi-Fi / ○ : applied to BLE)	Reported in
Static identifier	`WPS UUID` in Wi-Fi	① <SYNC_ID_CHG;wlan.ta;wps.uuid_e>	[28, 36]
	`Auth Tag` in *Apple* Nearby Info BLE messages	② <SYNC_ID_CHG;bthci_evt.bd_addr;apple_nearby_info.auth_tag>	[8, 11, 12, 27]
	`Device Hash` in *Microsoft* CDP BLE messages	③ <SYNC_ID_CHG;bthci_evt.bd_addr;microsoft_cdp.device_hash>	[8, 11]
Non-reset counter	`Sequence Number` in Wi-Fi	④ <SYNC_CNT_CHG;wlan.ta;wlan.seq>	[16, 28, 36]
	`IV` in *Apple* Handoff BLE messages	⑤ <SYNC_CNT_CHG;bthci_evt.bd_addr;apple_handoff.iv>	[8, 11, 12, 27]

As future work, we plan to extend the approach to automatically identify issues that were not previously known.

ACKNOWLEDGEMENTS

This work was supported by the INSA Lyon - SPIE ICS IoT chair and the H2020 SPARTA Cybersecurity Competence Network project. The authors would like to thank the Furious MAC research group at the US Naval Academy for sharing their corpus of traffic traces, as well as for their insightful comments.

REFERENCES

[1] 2018. *802.11aq-2018 - IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Preassociation Discovery*. Technical Report. IEEE standard association. https://standards.ieee.org/standard/802_11aq-2018.html

[2] Naeim Abedi, Ashish Bhaskar, and Edward Chung. 2013. Bluetooth and Wi-Fi MAC address based crowd data collection and monitoring: benefits, challenges and enhancement. (2013).

[3] Android. 2020. Privacy: MAC Randomization. (2020). https://source.android.com/devices/tech/connect/wifi-mac-randomization Accessed: 2020-03-12.

[4] Apple. 2020. About the security content of iOS 8. (2020). https://support.apple.com/en-us/HT201395 Accessed: 2020-03-12.

[5] Myrto Arapinis, Loretta Mancini, Eike Ritter, Mark Ryan, Nico Golde, Kevin Redon, and Ravishankar Borgaonkar. 2012. New privacy issues in mobile telephony: fix and verification. In *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 205–216.

[6] Calypso Barnes, François Verdier, Alain Pegatoquet, Daniel Gaffé, and Jean-Marie Cottin. 2016. Wireless sensor network protocol property validation through the system's simulation in a dedicated framework. In *2016 10th International Conference on Signal Processing and Communication Systems (ICSPCS)*. IEEE, 1–9.

[7] Emmanuel Bayse, Ana Cavalli, Manuel Nunez, and Fatiha Zaidi. 2005. A passive testing approach based on invariants: application to the WAP. *Computer networks* 48, 2 (2005), 247–266.

[8] Johannes K Becker, David Li, and David Starobinski. 2019. Tracking Anonymized Bluetooth Devices. *Proceedings on Privacy Enhancing Technologies* 1 (2019), 17.

[9] Abdelghani Benharref, Rachida Dssouli, Roch Glitho, and Mohamed Adel Serhani. 2006. Towards the testing of composed web services in 3 rd generation networks. In *IFIP International Conference on Testing of Communicating Systems*. Springer.

[10] Bram Bonné, Arno Barzan, Peter Quax, and Wim Lamotte. 2013. WiFiPi: Involuntary tracking of visitors at mass events. In *2013 IEEE 14th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 1–6.

[11] Guillaume Celosia and Mathieu Cunche. 2019. Saving Private Addresses: An Analysis of Privacy Issues in the Bluetooth-Low-Energy Advertising Mechanism. In *MobiQuitous 2019 - 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. 1–10.

[12] Guillaume Celosia and Mathieu Cunche. 2020. Discontinued Privacy: Personal Data Leaks in Apple Bluetooth-Low-Energy Continuity Protocols. *Proceedings on Privacy Enhancing Technologies* 2020, 1 (2020), 26–46.

[13] Gerald Combs. 1998. Wireshark - The world's foremost and widely-used network protocol analyzer. (1998). https://www.wireshark.org/ Accessed: 2020-03-12.

[14] Levent Demir, Mathieu Cunche, and Cédric Lauradoux. 2014. Analysing the privacy policies of Wi-Fi trackers. In *Proceedings of the 2014 workshop on physical analytics*. ACM, 39–44.

[15] Kassem Fawaz, Kyu-Han Kim, and Kang G Shin. 2016. Protecting Privacy of BLE Device Users. In *USENIX Security Symposium*. 1205–1221.

[16] Julien Freudiger. 2015. How talkative is your mobile device?: an experimental study of Wi-Fi probe requests. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 8.

[17] Marco Gruteser and Dirk Grunwald. 2005. Enhancing location privacy in wireless LAN through disposable interface identifiers: a quantitative analysis. *Mobile Networks and Applications* 10, 3 (2005), 315–325.

[18] Marc Haase, Matthias Handy, et al. 2004. BlueTrack - Imperceptible tracking of bluetooth devices. In *Ubicomp Poster Proceedings*. 2.

[19] Hesham Hallal, Sergiy Boroday, Andreas Ulrich, and Alexandre Petrenko. 2003. An automata-based approach to property testing in event traces. In *IFIP International Conference on Testing of Software and Communicating Systems*. Springer.

[20] Youssef Hanna, Hridesh Rajan, and Wensheng Zhang. 2008. Slede: a domain-specific verification framework for sensor network security protocol implementations. In *Proceedings of the first ACM conference on Wireless network security*.

[21] Giles Hogben. 2017. Changes to Device Identifiers in Android O. (2017). https://android-developers.googleblog.com/2017/04/changes-to-device-identifiers-in.html Accessed: 2020-03-12.

[22] CB Insights. 2017. The Store Of The Future: 150+ Startups Transforming Brick-And-Mortar Retail In One Infographic. (2017). https://www.cbinsights.com/research/retail-store-tech-startups-2016/ Accessed: 2020-03-12.

[23] Thierry Jeron, Herve Marchand, Sophie Pinchinat, and M-O Cordier. 2006. Supervision patterns in discrete event systems diagnosis. In *2006 8th International Workshop on Discrete Event Systems*. IEEE, 262–268.

[24] KimiNewt. 2020. PyShark - Python packet parser using wireshark's tshark. (2020). https://kiminewt.github.io/pyshark Accessed: 2020-03-12.

[25] Constantine E Kontokosta and Nicholas Johnson. 2017. Urban phenology: Toward a real-time census of the city using Wi-Fi data. *Computers, Environment and Urban Systems* 64 (2017), 144–153.

[26] Thomas Liebig and Armel Ulrich Kemloh Wagoum. 2012. Modelling Microscopic Pedestrian Mobility using Bluetooth.. In *ICAART (2)*. 270–275.

[27] Jeremy Martin, Douglas Alpuche, Kristina Bodeman, Lamont Brown, Ellis Fenske, Lucas Foppe, Travis Mayberry, Erik C Rye, Brandon Sipes, and Sam Teplov. 2019. Handoff All Your Privacy: A Review of Apple's Bluetooth Low Energy Implementation. *arXiv preprint arXiv:1904.10600* (2019).

[28] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C Rye, and Dane Brown. 2017. A study of MAC address randomization in mobile devices and when it fails. *Proceedings on Privacy Enhancing Technologies* 2017, 4 (2017), 365–383.

[29] Celestin Matte, Mathieu Cunche, Franck Rousseau, and Mathy Vanhoef. 2016. Defeating MAC address randomization through timing attacks. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*.

[30] ABM Musa and Jakob Eriksson. 2012. Tracking unmodified smartphones using wi-fi monitors. In *Proceedings of the 10th ACM conference on embedded network sensor systems*. ACM, 281–294.

[31] Lukasz Olejnik. 2017. Privacy of London Tube Wifi Tracking. (2017). http://blog.lukaszolejnik.com/privacy-of-london-tube-wifi-tracking/ Accessed: 2020-03-12.

[32] Andreas Pfitzmann and Marit Hansen. 2005. Anonymity, unlinkability, unobservability, pseudonymity, and identity management-a consolidated proposal for terminology. (2005).

[33] T Scott Saponas, Jonathan Lester, Carl Hartung, Sameer Agarwal, Tadayoshi Kohno, et al. 2007. Devices That Tell on You: Privacy Trends in Consumer Ubiquitous Computing.. In *USENIX Security Symposium*. 55–70.

[34] Bluetooth SIG. 2010. *Bluetooth Core Specification v4.0.* https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=456433 Accessed: 2020-03-12.

[35] Bluetooth SIG. 2019. *Bluetooth Core Specification v5.2.* https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=478726 Accessed: 2020-03-12.

[36] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S Cardoso, and Frank Piessens. 2016. Why MAC address randomization is not enough: An analysis of Wi-Fi network discovery mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. 413–424.

[37] Mathias Versichele, Tijs Neutens, Matthias Delafontaine, and Nico Van de Weghe. 2012. The use of Bluetooth for analysing spatiotemporal dynamics of human movement at mass events: A case study of the Ghent Festivities. *Applied Geography* 32, 2 (2012), 208–220.

[38] Tien Dang Vo-Huu, Triet Dang Vo-Huu, and Guevara Noubir. 2016. Fingerprinting Wi-Fi devices using software defined radios. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. 3–14.

Table 2: List of evaluated devices along with their operating system (OS) versions and identified issues. Gray lines depict a software update of involved devices.

Type	Device	OS version	Identified issue: Identifier ①	②	③	Counter ④	⑤	Addr. reuse
Laptop	Apple MacBook Pro (13", 2015)	macOS 10.13.6				✓	✓	
	Dell G3 17-3779	Windows 10 Pro - Version 1809 (OS Build 17763.1075)	✓		✓	✓		
	HP EliteBook Folio 1040 G3	Ubuntu 16.04.6 LTS				✓		
Watch	Apple Watch Series 2	watchOS 5.0.1		✓		✓	✓	
	Apple Watch Series 3	watchOS 5.1.3		✓		✓	✓	
Phone	Apple iPhone 5C	iOS 9.3.1		✓		✓	✓	
	Apple iPhone 5S	iOS 10.3.2		✓		✓	✓	
	Apple iPhone 5S	iOS 11.2.1		✓			✓	
	Apple iPhone 5SE	iOS 11.3		✓		✓	✓	
	Apple iPhone 6	iOS 12.1		✓		✓	✓	✓
	Apple iPhone 6S	iOS 11.4		✓		✓	✓	
	Apple iPhone 6S Plus	iOS 12		✓		✓	✓	
	Apple iPhone 7	iOS 11.2.6		✓		✓	✓	
	Apple iPhone 7 Plus	iOS 12.0.1		✓		✓	✓	
	Apple iPhone 8 Plus	iOS 11.4.1		✓		✓	✓	
	Apple iPhone XR	iOS 12.1.2		✓		✓	✓	
	Apple iPhone XR	iOS 12.4.1		✓			✓	
	Apple iPhone XS	iOS 13.1		✓		✓	✓	
	Apple iPhone XS Max	iOS 12.1		✓		✓	✓	
	Aquos sense	Android 8.0.0				✓		
	ASUS Zenfone 3	Android 7				✓		✓
	ASUS Zenfone 3 Deluxe	Android 6.0.1				✓		✓
	Blackberry Privilege	Android 5.1.1	✓			✓		✓
	Google Pixel XL	Android 7.1				✓		
	Google Pixel XL	Android 9						
	HTC One A9	Android 6				✓		✓
	HTC U11	Android 7.1.1				✓		
	Huawei Mate10 lite	Android 7				✓		
	Huawei Nexus 6P	Android 6.0.1	✓			✓		✓
	Huawei P10 Lite	Android 7				✓		
	Huawei P20 Lite	Android 8.0.0				✓		✓
	Huawei P9	Android 6				✓		✓
	Huawei P9 Lite	Android 6	✓			✓		
	Huawei Y7 Prime (2018)	Android 8.0.0				✓		
	LG V20	Android 7				✓		✓
	Motorola Moto G Play (6th gen.)	Android 8.0.0				✓		✓
	Motorola Moto e	Android 5.1				✓		✓
	Motorola Moto E (4th gen.)	Android 7.1.1				✓		
	Motorola Moto E Plus (4th gen.)	Android 7.1.1				✓		✓
	Motorola Moto G (3rd gen.)	Android 5.1.1				✓		✓
	Motorola Moto G (4th gen.) Plus	Android 6.0.1				✓		
	Motorola Moto G (5th gen.)	Android 7				✓		✓
	Motorola Moto G (5th gen.) Plus	Android 7				✓		✓
	Motorola Moto G4 Plus	Android 6.0.1				✓		✓
	Motorola Moto G5	Android 7				✓		
	Motorola Moto G5 Plus	Android 7				✓		✓
	Motorola Moto GS (5th gen.)	Android 7.1.1				✓		✓
	Motorola Moto Z Play	Android 6.0.1				✓		
	Motorola Moto Z Play	Android 9						
	Motorola Nexus 6	Android 7	✓					✓
	OnePlus 2	Android 6.0.1				✓		
	OnePlus 3T	Android 7				✓		
	Sony Xperia X Compact	Android 7				✓		
	Sony Xperia X Compact	Android 8.0.0						
	Sony Xperia XZ Premium	Android 8.0.0				✓		✓
	Sony Xperia XZ1	Android 8.0.0				✓		
	Xiaomi Mi 5	Android 7				✓		
	Xiaomi Mi A1	Android 7.1.2				✓		
	Xiaomi Mi A1	Android 8.0.0						
	Xiaomi Redmi 3S	Android 6.0.1				✓		✓
	Xiaomi Redmi 4A	Android 7.1.2				✓		✓
	Xiaomi Redmi 4X	Android 7.1.2				✓		✓
	Xiaomi Redmi 5 Plus	Android 7.1.2				✓		✓
	Xiaomi Redmi 5A	Android 7.1.2				✓		✓
	ZTE Blade X Max	Android 7.1.1				✓		✓
	ZTE Grand X 4	Android 6.0.1				✓		✓