On the process of fixing privacy issues in Wi-Fi enabled devices

Clément Lagneau-Donzelle, Mathieu Cunche

On the process of fixing privacy issues in Wi-Fi enabled devices

Clément Lagneau-Donzelle
INSA-Lyon, University of Lyon
Villeurbanne, France
clement.lagneau-donzelle@insa-lyon.fr

Mathieu Cunche
INSA-Lyon, Inria, University of Lyon CITI, Lab.
Villeurbanne, France
mathieu.cunche@insa-lyon.fr

***Abstract*—Several practical privacy issues have been uncovered in Wi-Fi technologies since 2016 [20], [23]. Those issues allow an attacker to defeat address randomization, an anti-tracking mechanism, and thus to track users of wireless devices in the physical world. As recently demonstrated by Martin et. al [12], OS developers and vendors are deploying fixes, and the number of vulnerable devices is progressively decreasing.**

After presenting a comprehensive overview of three active attacks, we conduct a number of experiments to assess whether those issues have been fixed. Using a sample of Wi-Fi enabled devices we found that although some issues have been corrected, some devices remain vulnerable and expose their users to tracking. We analyze the implementation of countermeasures in `wpa_supplicant`, the software component in charge of Wi-Fi features in Linux and Android. We found that, even if countermeasures have been available for several years in `wpa_supplicant`, they are still not always included or activated in deployed OS. We discuss the reasons behind this lack of protection.

***Index Terms*—Wi-Fi, 802.11, tracking, privacy, address randomization, operating system, mobile, Android, Linux, iOS**

I. Introduction

Wireless technologies have become ubiquitous as they are integrated in many devices and deployed in many locations. Wi-Fi is one of the most popular wireless technologies and is today integrated in 16.4 billion devices worldwide[1]. Many of the devices including a Wi-Fi interface are mobile devices, such as smartphones, that are carried all day long by users.

Since its early days, Wi-Fi has been the subject of various security issues [9], [22]. More recently, practical privacy issues have demonstrated how personal data can be exposed through Wi-Fi [11], [14], [21]. In particular, Wi-Fi expose users to tracking and this threat has been exploited to monitor individuals in the physical world [13], [18].

Address randomization [15] has been introduced to protect users against tracking. However, numerous flaws have been identified in the first implementations of address randomization [10], [20], [23]. A recent study [12] have shown that, 4 years after the publication of the attacks, some devices are still vulnerable.

This work has been supported by the ANR-BMBF PIVOT project (ANR-20-CYAL-0002), H2020 SPARTA project and the INSA-Lyon SPIE ICS IoT Chair.

[1]https://www.wi-fi.org/discover-wi-fi/value-of-wi-fi

In this paper, we investigate the challenge of fixing those vulnerabilities in deployed devices. By focusing on a set of active attacks, we confirm that some devices remain vulnerable, and analyze the development of fixes and their deployment in devices. The contributions of this paper are the following:

- We present a comprehensive overview of three previously published active attacks that can be used to defeat address randomization.
- Using a set of Wi-Fi enabled devices, we assess their vulnerability to those attacks. We find that those attacks are still valid since some devices remain vulnerable to them.
- We then analyze the fix of those issues in `wpa_supplicant`, the software component in charge of Wi-Fi operations in Linux and Android operating systems. We find that there is a significant delay between the publication of the attack and the deployment of fixes in deployed devices.
- Finally, we discuss challenges associated with the creation and deployment of those patches as well as the information available on those processes.

This paper is organized as follows. Section II introduces background information on Wi-Fi and address randomization. Then, an overview of the considered attacks is presented in Section III, and the vulnerability of a sample of devices is evaluated in Section IV. Section V presents the implementation of fixes and countermeasures, and Section VI discusses the issues associated with the deployment of patches. Section VII presents the related works and Section VIII concludes the paper.

II. Background

A. Wi-Fi (IEEE 802.11)

Wi-Fi is a wireless technology based on the IEEE 802.11 standards [16], which is in general operated in infrastructure mode: a set of client stations connected to an access point (AP). Wi-Fi frames feature a header which includes an address field designating the source of the device. This address field usually contains a MAC address, a globally unique 48-bit identifier, but can be set to another value as discussed in the following section (section II-B).

The discovery mechanism of Wi-Fi, which allows devices to detect nearby AP, can be done either through beaconing

or through a probe request/response approach. The latter is widely used by portable devices; it involves client station broadcasting probe requests to which AP answer with probe responses, thus declaring their presence and characteristics. As a result, client station periodically broadcast probe requests including their address.

B. Address randomization

The source address included in Wi-Fi frames can be leveraged to identify devices and track users. To protect users against wireless-based tracking, the concept of address randomization has been proposed [15] and then included in a wide range of implementations. Starting with Apple's iOS8 [8] and followed by other operating systems [5], the source addresses included in the headers of probe requests has been replaced with a random and periodically changing identifier.

The source address field is usually set to the MAC address of the transmitting device. When address randomization is used, the source address is set to a random value[2] that is rotated at regular interval. Currently, most implementations renew the address every 15 minutes.

This anti-tracking measure has managed to thwart the tracking [6] but implementations often include mistakes that renders them vulnerable to attacks [20], [23].

III. Active tracking attacks

Among the attacks against address randomization presented in the literature, there are a number of active ones, which relies on the transmission of wireless packets. More specifically, those attacks leverage the mechanisms in protocol implementations to bypass the protection provided by address randomization.

A. Attacker models & assumption

We consider an attacker that is active, i.e., he is able to monitor the wireless communications but also generate traffic. For instance, the attacker can forge and inject frames or capture and replay frames transmitted by other devices. In the case of Wi-Fi, those characteristics can easily be achieved using commodity hardware and open-source software. Wireless dongles capable of capturing and injecting Wi-Fi packets can be acquired for approximately $20.

The objective of the attacker is to track devices that are using address randomization. Based on the attacks presented in the literature, we present a classification of active attacks into two types:

- **Type A - Static identifier reveal:** the attacker forces the device to expose one of its static identifiers.
- **Type B - Linkage with a known identifier:** the attacker forces the device to expose its link to a given identifier.

[2]Some implementations only randomize the rightmost part of the address, i.e. the NIC.

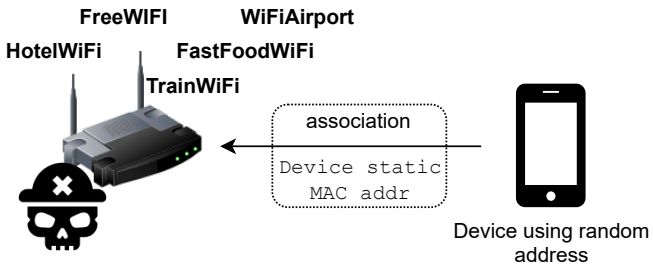

Fig. 1. Karma attack [23, sec 6.1] forcing a device to reveal its static MAC address. The attacker creates a fake Access Point advertising popular SSIDs. A device using address randomization for probing will recognize one of the SSIDs and will attempt to connect using its static MAC address.

B. Revived Karma attack [23, sec 6.1]

This attack [23, sec 6.1] is based on the service discovery in 802.11 networks and allows to obtain the real MAC address of a station even if it is using address randomization for probing (discovery mechanism - see Section II). It relies on the automatic connection mechanism found in most 802.11 devices: whenever a station detects an AP with a known SSID (SSID of a network configured in the device), it will automatically connect to the access point (AP); when associating to an AP, the device uses its real address and not a random one. By advertising several popular SSIDs, the attacker will force nearby devices to initiate a connection, thus revealing their real MAC address (see Figure 1).

1) Hotspot 2.0 Honeypot [23, sec 6.2]: This attack [23, sec 6.2] is similar to the Karma attack and is based on the Hotspot 2.0 (HS2.0) feature to obtain the real MAC address. Access point supporting HS2.0 advertise this support but do not advertise their full HS2.0 profile. To obtain this information, a device must query the AP through Generic Advertisement Service (GAS) - Access Network Query Protocol (ANQP) request. When performing this request, it was found that many devices use their real MAC address instead of a random one. Furthermore, devices supporting HS2.0 will automatically query an HS2.0 enabled AP as soon as it is discovered. An attacker creating an AP advertising HS2.0 capabilities will trigger nearby devices to send ANQP requests including the real MAC address of the device (see Figure 2).

C. Control frame attack [20, sec 6.7]

This attack [20, sec 6.7] leverages the RTS-CTS mechanism that is used to reduce the risk of frame collision. When receiving an RTS frame, a device will answer with a CTS frame. The RTS frame is targeted toward a single device that is identified by its MAC address specified in the RTS frame. As a result, sending an RTS frame targeting a given MAC address will force the corresponding device to reveal its presence by responding with a CTS frame. In the case of a device using address randomization, sending a RTS frame with its real MAC address will reveal its real identity (see Figure 3).

TABLE I
SUMMARY OF THE PRESENTED ATTACKS.

Technology	Attack	Type A reveal static id.	Type B link to known id.
Wi-Fi (802.11)	Revived Karma	✓	
	Hotspot 2.0	✓	
	RTC-CTS		✓

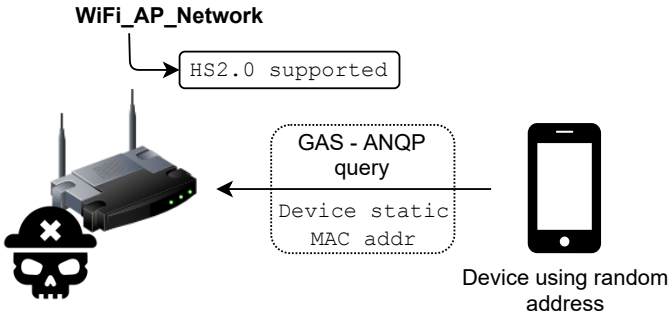

Fig. 2. Hotspot 2.0 attack [23, sec 6.2] forcing the device to reveal its real MAC address. The attacker creates a fake Access Point advertising for Hotspot 2.0 capabilities. Device supporting HS2.0 will automatically query the AP through GAS- ANQP query including its real MAC address.

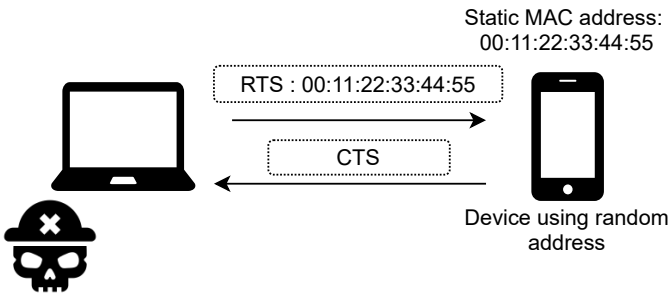

Fig. 3. RTS-CTS attack [20, sec 6.7] forcing a device using address randomization to reveal its presence. The attacker emits an RTS frame targeting the real MAC address of the victim; the victim will automatically answer with a CTS frame, thus revealing its presence.

IV. TESTING THE ATTACKS

In this section, we test whether a sample of Wi-Fi enabled devices are vulnerable to the attacks presented in Section III. All those attacks were conducted in our lab on devices own by the authors.

A. Revived Karma attack

This attack was conducted using `hostapd` to create fake access points. For each device, we used `hostapd` to create a network with an SSID matching one of the networks configured on the device. We then observe whether the device connects to the AP and whether it uses a random address or its real MAC address.

The results are presented in Table II. Among the 7 tested devices, 5 are still vulnerable to the attack[3]

B. Hotspot 2.0

This attack was conducted using `hostapd` to create fake Hotspot 2.0 access points[4]. After starting `hostapd`, we used another interface in monitor mode to analyze ANQP frames in order to verify if the source address was random or not. The results are on Figure II. We found that only the Galaxy S9+ was generating ANQP requests, and that those requests were using a random address. Other devices did not generated ANQP request, suggesting that they did not support the feature.

C. RTS-CTS

This attack was conducted using `scapy` to forge RTS frame with the targeted MAC address. For each device we continuously sent RTS frames (approximately every 5ms) for at least 2 minutes. At the same time, we monitored the channel for CTS response.

The results are presented in table II. We found that out of the 6 tested devices, 3 are vulnerable to the attack.

D. Summary of test results

Focusing on the two considered iOS devices, we found that the most recent is protected against all three attacks, while the oldest was only protected against once. This suggests that the Karma and RTS-CTS issues have been corrected in iOS between version 10.3.3 (2017) and 14.1 (2020). The Galaxy S9+ is also protected against all three attacks, suggesting that the most recent of Android correctly implements countermeasures. Looking at the three Windows and Linux laptop computers (MSI, Lenovo, Thinkpad), we can observe that, even if they are running recent versions of the OS (all released in 2020), they are still vulnerable to Karma and RTS-CTS attacks.

V. FOLLOWING THE CORRECTION IN WPA_SUPPLICANT

`wpa_supplicant` [4] is a software that implements features of Wi-Fi client station such as key negotiation, authentication/association and roaming. `wpa_supplicant` is integrated in Linux and Android operating systems.

[3]Although the Galaxy S9 appears to be protected for new networks, we noticed that for a network which has been configured several years ago, MAC randomization is not used and the device is thus vulnerable. Thus, it seems that the protection only applies to networks configured after the corresponding update.

[4]We compiled `hostapd`from the sources and modified the configuration file '.config' to setup the flags 'CONFIG_INTERWORKING=y', 'CONFIG_HS20=y' and 'CONFIG_EAP_PWD=y' to enable Hotspot 2.0.

TABLE II
SUMMARY OF THE RESULTS WITH THE THREE CONSIDERED ATTACKS. A ✗ INDICATES THAT THE DEVICE IS STILL VULNERABLE TO THE ATTACK WHILE A ✓ INDICATES THAT THE DEVICE SUCCESSFULLY EVADED THE ATTACK. A '-' INDICATES THAT THE DEVICE DID NOT REACTED TO ATTACK, WHILE A LACK OF SYMBOL INDICATES THAT THE DEVICE HAS NOT BEEN TESTED.

Device name	OS version	OS release year	Karma	HS2.0	RTS-CTS
Galaxy S9+	Android 10 One UI 2.1	2020	✓	✓	✓
MSI GS60 6QC GHOST	Windows 10 18362.1139	2020	✗		
Lenovo G500s	Ubuntu 18.04.4	2020	✗	-	✗
ThinkPad T460	W10 Pro 1903 / 18 362 / 1082	2020	✗	-	✗
iPhone SE	iOS 14.1	2020	✓	-	✓
MacBook	MacOS Catalina 10.15.7	2020	✗	-	✓
iPad	iOS 10.3.3	2017	✗	-	✗

Since their publication, the issues discussed in this paper, have received the scrutiny of `wpa_supplicant`'s contributors, and patches have been created. Those patches are listed in the changelog [2] of `wpa_supplicant` and their details can be found in the `wpa_supplicant` git repository. In this section, we track the correction of those issues in `wpa_supplicant`.

A. Revived Karma attack

This attack relies on the fact that a client station will switch to its real MAC address post-association (i.e., when it is connected to an AP). A feature allowing client station to use a random address in this post-association setting has been implemented as early as September 2014 [19] and included in version 2.3 of `wpa_supplicant` released on 2014-10-09, although it was initially presented as an experimental feature.

This behavior can be controlled via the `mac_addr` parameter of `wpa_supplicant.conf` that specifies the MAC address policy, i.e., whether to use per-network address or the real MAC address in a post-association setting.

```
# MAC address policy default
# 0 = use permanent MAC address
# 1 = use random MAC address for each ESS connection
#
# By default, permanent MAC address is used unless policy is changed by
# the per-network mac_addr parameter. Global mac_addr=1 can be used to
# change this default behavior.
#mac_addr=0
```

Listing 1. Configuration of the MAC address policy via the `mac_addr` parameter in `wpa_supplicant.conf` [19]

Note that this feature does not fully solve the issue, as client station uses a constant address for a given network; therefore, even if the client station does not expose its real MAC address, it exposes an identifier that does not change over time if the SSID (network name) used by the attacker is constant.

B. Hotspot 2.0

The issue with Hotspot 2.0 is similar to the one of the revived Karma attack: the client station will switch to its real MAC address when interacting with the AP. The support for randomized address in GAS queries has been added to `wpa_supplicant` [17] on December 8th 2016 and was included in the v2.7 released on 2018-12-02[5].

[5]Note that an experimental support of this feature was available as soon as 2014 [19]

```
# MAC address policy for GAS operations
# 0 = use permanent MAC address
# 1 = use random MAC address
# 2 = like 1, but maintain OUI (with local admin bit set)
#gas_rand_mac_addr=0

# Lifetime of GAS random MAC address in seconds (default: 60)
#gas_rand_addr_lifetime=60
```

Listing 2. Configuration of the MAC address policy for GAS MAC operations via the `gas_rand_mac_addr` parameter in `wpa_supplicant.conf` [17]

Using this feature, GAS requests will use a random address that will be changed at a period controlled via the `gas_rand_addr_lifetime` parameter. Thanks to this feature, client stations are not vulnerable to Hotspot 2.0 attack.

C. RTS-CTS

The RTS-CTS attack leverages a mechanism that is not directly included in `wpa_supplicant`but in its parent component `hostapd`which implements access points features. This attack does not abuse an implementation bug but a feature that comes from a misconfiguration of the system (i.e., enabling RTS-CTS mechanism on pre-association client stations). Therefore, the correction of this issue does not require an implementation to be fixed but an appropriate configuration.

The RTS-CTS can be configured through the `rts_threshold` parameter found in the `hostapd.conf`.

```
# RTS/CTS threshold; 2347 = disabled (default); range 0..2347
# If this field is not included in hostapd.conf, hostapd will not control
# RTS threshold and 'iwconfig wlan# rts <val>' can be used to set it.
rts_threshold=2347
```

Listing 3. `rts_threshold` parameter in the configuration file of `hostapd`, `hostapd.conf` [3]

Vendors could remove the RTS-CTS problematic feature through this configuration file or simply by disabling `hostapd`when it is not needed.

VI. DISCUSSION

A. Delay in the deployment of fixes

As seen in section V, countermeasures have been available for some years and yet, devices are still vulnerable. In the case of the RTS-CTS attack the countermeasure has always

been available since it is simply disabling a standard Wi-Fi feature. For the Hotspot 2.0 and revived Karma attacks, experimental countermeasures have been implemented as soon as 2014, i.e., two years before those attacks were demonstrated on real devices.

Once implemented in software component such as `wpa_supplicant`, the new versions need to be deployed in the operating system. On Android, `wpa_supplicant` v2.3 included in Android 5.1.1 in April 2015[6] while v2.7 was included in Android 10 released in September 2019[7]. In other words, the fixes implemented in `wpa_supplicant` took between 6 and 10 months to be integrated in Android.

There is therefore a delay between the implementation of a countermeasure and its deployment in devices. This delay has two main components: the time for the patch to reach an official release of the software and the time for this release to be integrated in the OS. On top of that, one needs to add the time necessary for the issue to be identifier and patched by the developers.

Having the countermeasure integrated in the OS is not enough. As seen in section V, those features still need to be enabled. Those features may not be active by default in the Android code and the vendor will have to modify the configuration files to enable them.

B. Lack of information on the fixes

In order to deploy and enable countermeasures, vendors need to be aware of the threats and the corresponding protection features and patches. We found that there is little information available on the correction of those issues.

One reason is that, as opposed to security vulnerabilities, privacy issues are not tracked through *Common Vulnerabilities and Exposures* (CVE). CVE are presented in details and their fixes is well documented in changelog of software component [2] and operating systems [1]. This is not the case for most privacy issues.

The source code history of software components may also include traces of such corrections. This information only applies to open-source projects and is not applicable to closed-source systems such as Windows and iOS.

VII. Related works

Attacks against address randomization have been presented in several works. In [23], the authors introduced several passive and active attacks, including the revived Karma and Hotspot 2.0 attacks. Later, the authors of [20] consolidated this set of attacks by adding, among others, the RTS-CTS attack.

Several works have shown that Wi-Fi address randomization can be defeated using information at the physical layer. In [26], the Vo-Huu et al, present attacks against address randomization in WiFi leveraging the Carrier Frequency Offset. It has been demonstrated that the scrambler seed found in some 802.11 frames can be leveraged to defeat address randomization, first in 802.11p vehicular networks [10] and then in commodity Wi-Fi devices [23]

In [12], the authors revisited a corpus of privacy attacks affecting Wi-Fi enabled devices. They found that those issues are progressively being fixed, especially in the most recent devices, but that a number of devices are still affected. They note that the correction of issues depends on the OS and also on the manufacturer.

The software component `wpa_supplicant` has recently suffered several security issues through the discovery of a number of vulnerabilities affecting WPA encryption [24], [25].

The issue of Android ROM customization is considered in several works: in [27] by focusing on security configuration and in [7] with the problem of drivers. In both cases, the customization of Android ROM by vendors often leads to weakened security protections.

VIII. Conclusion

In this paper, we considered the challenges associated with the fix of privacy issues in Wi-Fi implementations. Through an experimental assessment, we showed that devices are still affected by issues, several years after the publication of the attacks. Focusing on `wpa_supplicant`, we analyzed the integration process of countermeasures for those privacy issues. We found that fixes were quickly implemented, and that some countermeasures were available before the publication of the attacks.

We present several elements to explain why some of current devices are still affected by the issues. First, there is a delay in the deployment of fixes in devices as new software versions go through several steps of integration. Second, some of those fixes need to be enabled through configuration files, and vendors sometime fail at doing so.

Vendors therefore play an important role in the correct deployment of privacy preserving fixes. Additional publicity around those issues and the implemented fixes could stimulate vendors to better integrate those protections in their products.

[6]https://cs.android.com/android/platform/superproject/+/android-5.1.1_r38:external/wpa_supplicant_8/wpa_supplicant/ChangeLog

[7]https://cs.android.com/android/platform/superproject/+/android10-release:external/wpa_supplicant_8/wpa_supplicant/ChangeLog

References

[1] Android Security Bulletin—May 2021. https://source.android.com/security/bulletin/2021-05-01.

[2] Changelog for wpa_supplicant.

[3] hostapd.conf in android. https://android.googlesource.com/platform/external/wpa_supplicant_8/+/87fd279308af3f806848c8f2ab65ef18c6ac4c30/hostapd/hostapd.conf. Accessed: 2021-07-01.

[4] Linux WPA Supplicant (IEEE 802.1X, WPA, WPA2, RSN, IEEE 802.11i).

[5] Privacy: MAC Randomization.

[6] MAC IOS Randomization "Fix" results in 70x drop in Passersby traffic, February 2021.

[7] Yousra Aafer, Xiao Zhang, and Wenliang Du. Harvesting inconsistent security configurations in custom android roms via differential analysis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 1153–1168, 2016.

[8] Apple. About the security content of iOS 8, November 2014.

[9] Andrea Bittau, Mark Handley, and Joshua Lackey. The final nail in wep's coffin. In *2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 15–pp. IEEE, 2006.

[10] Bastian Bloessl, Christoph Sommer, Falko Dressier, and David Eckhoff. The scrambler attack: A robust physical layer attack on location privacy in vehicular networks. In *2015 International Conference on Computing, Networking and Communications (ICNC)*, pages 395–400, Garden Grove, CA, USA, February 2015. IEEE.

[11] Mathieu Cunche, Mohamed-Ali Kaafar, and Roksana Boreli. Linking wireless devices using information contained in Wi-Fi probe requests. *Pervasive and Mobile Computing*, 11:56–69, April 2014.

[12] Ellis Fenske, Dane Brown, Jeremy Martin, Travis Mayberry, Peter Ryan, and Erik C Rye. Three years later: A study of mac address randomization in mobile devices and when it succeeds. *Proc. Priv. Enhancing Technol.*, 2021(3):164–181, 2021.

[13] Samuel Gibbs. Shops can track you via your smartphone, privacy watchdog warns. *The Guardian*, January 2016.

[14] Ben Greenstein, Ramakrishna Gummadi, Jeffrey Pang, Mike Y. Chen, Tadayoshi Kohno, Srinivasan Seshan, and David Wetherall. Can Ferris Bueller Still Have His Day Off? Protecting Privacy in the Wireless Era. In *HotOS*, 2007.

[15] Marco Gruteser and Dirk Grunwald. Enhancing Location Privacy in Wireless LAN Through Disposable Interface Identifiers: A Quantitative Analysis. *Mobile Networks and Applications*, 10(3):315–325, June 2005.

[16] IEEE Std 802.11-2012. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 2012.

[17] Vamsi Krishna and Jouni Malinen. Gas: Add support to randomize transmitter address. https://bitbucket.fem.tu-ilmenau.de/projects/CAMPUSWLAN/repos/hostapd/commits/1d9d21f37694d45d6d885745855eaf6e5f1bc284#wpa_supplicant/wpa_supplicant.conf, December 2016.

[18] Lukasz Olejnik. Privacy of London Tube Wifi Tracking, September 2017.

[19] Jouni Malinen. Add support for using random local mac address. https://bitbucket.fem.tu-ilmenau.de/projects/CAMPUSWLAN/repos/hostapd/commits/c267753ba2cc006907c57cf11b06d658f783682f#wpa_supplicant/wpa_supplicant.conf, September 2014.

[20] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C. Rye, and Dane Brown. A Study of MAC Address Randomization in Mobile Devices and When it Fails. *Proceedings on Privacy Enhancing Technologies*, 2017(4):365–383, October 2017.

[21] Jeffrey Pang, Ben Greenstein, Ramakrishna Gummadi, Srinivasan Seshan, and David Wetherall. 802.11 User Fingerprinting. In *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, MobiCom '07, pages 99–110, New York, NY, USA, 2007. ACM.

[22] Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin. Breaking 104 bit wep in less than 60 seconds. In *International Workshop on Information Security Applications*, pages 188–202. Springer, 2007.

[23] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S. Cardoso, and Frank Piessens. 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*, ASIA CCS '16, pages 413–424, New York, NY, USA, 2016. ACM.

[24] Mathy Vanhoef and Frank Piessens. Key reinstallation attacks: Forcing nonce reuse in WPA2. In *Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS)*. ACM, 2017.

[25] Mathy Vanhoef and Frank Piessens. Release the kraken: new KRACKs in the 802.11 standard. In *Proceedings of the 25th ACM Conference on Computer and Communications Security (CCS)*. ACM, 2018.

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

[27] Xiaoyong Zhou, Yeonjoon Lee, Nan Zhang, Muhammad Naveed, and XiaoFeng Wang. The peril of fragmentation: Security hazards in android device driver customizations. In *2014 IEEE Symposium on Security and Privacy*, pages 409–423. IEEE, 2014.