Short: How Talkative is your Mobile Device? An Experimental Study of Wi-Fi Probe Requests

Julien Freudiger
PARC (A Xerox Company)

ABSTRACT

The IEEE 802.11 standard defines Wi-Fi probe requests as a active mechanism with which mobile devices can request information from access points and accelerate the Wi-Fi connection process. Researchers in previous work have identified privacy hazards associated with Wi-Fi probe requests, such as leaking past access points identifiers and user mobility. Besides several efforts to develop privacy-preserving alternatives, modern mobile devices continue to use Wi-Fi probe requests. In this work, we quantify Wi-Fi probe requests' threat to privacy by conducting an experimental study of the most popular smartphones in different settings. Our objective is to identify how different factors influence the probing frequency and the average number of broadcasted probes. Our conclusions are worrisome: On average, some mobile devices send probe requests as often as 55 times per hour, thus revealing their unique MAC address at high frequency. Even if a mobile device is not charging and in sleep mode, it might broadcast about 2000 probes per hour. We also evaluate a commercially deployed MAC address randomization mechanism, and demonstrate a simple method to re-identify anonymized probes.

1. INTRODUCTION

Mobile devices regularly broadcast Wi-Fi probe requests in order to advertise their presence and actively discover Wi-Fi access points in proximity [15]. A new breed of service providers takes advantage of Wi-Fi probe requests to monitor user activities, such as shopping habits [2, 4, 8, 20, 23, 25], or to reduce the cost of sharing content across smartphones [27]. Wi-Fi probe requests are unique identifiers, as they contain the MAC address of mobile devices, and may also include a list of preferred Wi-Fi networks (viz., SSIDs) accessed by these devices in the past.

In previous work, researchers have identified the privacy dangers associated with Wi-Fi probe requests and warned against potential abuse of user location tracking [7, 21] and profiling [6, 24]. In particular, several researchers showed that a passive adversary could track user locations via probe requests [7] and profile users by analyzing SSIDs included in probe requests [5, 6, 24]. In fact, probe requests may even contain users' names [17].

WiSec'15 June 22-26 2015, New York City, NY, USA
ACM 978-1-4503-3623-9/15/06.

Several countermeasures exist and most consist in modifying the Wi-Fi service discovery process. Lindqvist et al. [18] propose a privacy-preserving Wi-Fi discovery process using cryptographic challenge-responses on top of probe requests. Kim et al. [16] propose broadcasting different probes depending on context. Beresford and Stajano [3] suggest changing MAC addresses over time. Greenstein et al. [12] propose removing link-layer identifiers altogether. Unfortunately, these approaches have not been adopted in practice. Recently, Apple released a privacy feature that uses short-term and changing MAC addresses for Wi-Fi probes in certain conditions [19]. This approach is reminiscent of the concept of mix zones [3, 10]. Overall, users are mostly unaware of the fact that their devices broadcast probe requests, and lack control over it.

To the best of our knowledge, none of the existing work has characterized how often mobile devices broadcast Wi-Fi probes and the extent of the threat. Quantification of privacy threats is essential to design effective privacy protection mechanisms, and researchers in previous work did so for location privacy [11, 26], deanonymization [22], and Wi-Fi fingerprinting [9] among others. In this work, we seek to quantify the privacy threat of probe requests by measuring how often and under what conditions mobile devices broadcast probe requests. Our hope is that a deeper understanding of the privacy risks associated with Wi-Fi probes will increase awareness and transparency about potential privacy concerns, and fuel the release and adoption of stronger protection mechanisms.

Contributions. Our main contributions are as follows:

1. We experiment with several packet capture configurations (e.g., antennas and network interfaces) in order to identify an efficient approach to collect Wi-Fi probe requests;
2. We measure the number of probe requests broadcasted by mobile devices under different configurations and quantify the privacy risk;
3. We study the effectiveness of a currently deployed MAC randomization mechanism.

Results. We reach a number of surprising conclusions. First, we find a large discrepancy in the amount of Wi-Fi probes collected with different packet capture configurations. We find that researchers in previous work are likely to have experimented on about 57% of the data that we are able to collect, and might thus have obtained lower bounds on privacy concerns. Second, we find that the number of broadcasted probe requests depends on the OS and the number of networks known to a mobile device. For example, Android L 5.0.1 broadcasts about 1500 probes per hour in general. With iOS 8.1.3, about 120 are broadcasted per hour with a small increase in scenarios with a larger number of networks known. In contrast,

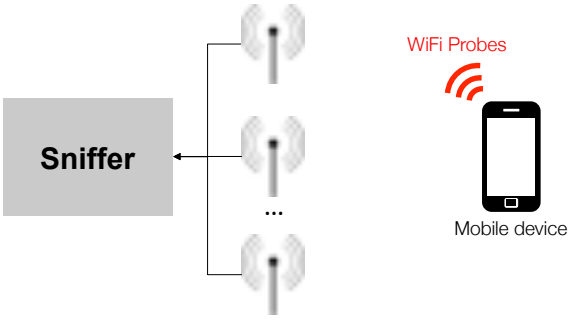


Figure 1: Threat model. A sniffer uses Wi-Fi-compatible antennas to collect Wi-Fi probe requests broadcasted by a nearby mobile device on 802.11b/g/n channels.

Android Kitkat 4.4.2 linearly increases the number of broadcasted probe requests depending on the number of known networks. We also observe that probe requests are bursty in nature. With 4 known networks, Android L 5.0.1 broadcasts probes on average every 66 seconds, Android 4.4.2 every 72 seconds, whereas iOS 8.1.3 broadcasts on average every 330 seconds. These measurements indicate that an adversary could track devices at the granularity of minutes. Third, we make a few positive observations. We notice that SSIDs are not systematically broadcasted with recent mobile OSes, and only included for hidden networks. We also learn that BlackBerry OS 10.3.1 does not broadcast probe requests unless the network is hidden. Finally, we find that the implementation of Apple iOS 8.1.3 MAC randomization can be defeated: It is possible to link probe requests coming from the same device using different MAC addresses by using Wi-Fi probe requests' sequence number and other vendor specific information.

In summary, as of 2015, billions of smartphones in the world are publicly broadcasting their unique identifiers at a high frequency. Although wireless network discovery is an important problem, privacy consequences seem at odds with technical gains associated with active network discovery.

2. PRELIMINARIES

We describe the considered scenario and associated threat model.

2.1 System Model

The IEEE 802.11 standard [15] defines guidelines for Wi-Fi communications. Legacy 802.11b/g operates in the 2.4GHz frequency band over 11 channels (in the US). Out of these 11 channels, three are non-overlapping (channels 1, 6, and 11). A recent update to the standard (802.11n) also supports 21 non-overlapping channels at higher frequencies (5GHz). These new channels have much higher capacity, but are prone to interference with other systems and are not always available: They are assigned using a Dynamic Frequency Selection (DFS) technique. In practice, only 4 channels out of the 21 are guaranteed. Most modern mobile devices switch between 802.11b/g and n, using a technique known as *band steering*, depending on traffic demands, existing network congestion, as well as their distance to the access point. The 2.4GHz band provides longer range compared to the 5GHz channel (which struggles to operate through walls).

An important challenge of wireless networks is service discovery. Mobile devices and Wi-Fi Access Points (AP) need a mechanism to efficiently announce their presence to each other. The IEEE 802.11 standard [15] defines two mechanisms to do so: A passive mechanism in which APs periodically advertise their presence to mobile devices using beacons, and an active mechanism in which mobile devices actively search for APs using probe requests.

2.1.1 Wi-Fi Beacons

Wi-Fi access points announce their presence by broadcasting *beacon* management frames that contain network configuration parameters, such as the Service Set Identifier (SSID) and supported data rates. Mobile devices listen for beacons on 802.11b/g/n channels, and passively detect nearby APs. The beaconing rate depends on the desired network discovery speed, and acceptable bandwidth overhead. Most APs set a beaconing interval of about 100ms. Mobile devices can respond to beacons with Wi-Fi association frames.

2.1.2 Wi-Fi Probe Requests and Responses

In addition to passive discovery, mobile devices can proactively discover APs by sending *probe request* management frames on 802.11b/g/n channels. Probe requests include a unique device identifier (e.g., the MAC address), a device's capabilities (e.g., supported 802.11 standards), and can be directed to a specific AP (by indicating its SSID), or broadcasted to all APs within range. Table 1 provides an example of probe requests including the different headers, such as the sequence number (SEQ). Most probe requests are sent in vain as there might not be an AP in proximity. Upon receiving a probe request, an AP informs the client of its presence using a *probe response*.

Network discovery based on beacons tends to be slow and energy consuming. By sending active probes, a mobile device can keep the Wi-Fi radio on for just a few milliseconds, the amount of time it takes for a probe response to be received, and quickly discover nearby networks. This is particularly helpful in scenarios where a mobile device roams across APs from the same provider. Similarly, probes help maintain Wi-Fi connections even when mobile devices are asleep. Finally, probe requests are the only solution to connect to hidden networks (viz., APs that do not broadcast beacons). Technically, the 802.11 standard does not impose a broadcasting frequency for probe requests and beacons.

2.2 Threat Model

Since probe requests are broadcasted before association with APs, they are sent in the clear. They are thus easy to collect with commodity hardware, such as wireless cards in monitoring mode, and standard software, such as Airodump, TCPDump, or Wireshark.

We assume a passive adversary that aims to collect as many Wi-Fi probe requests as possible using a number of wireless antennas in monitoring mode (Fig. 1). For each device broadcasting probe requests, the adversary learns the MAC address of the device, possibly preferred SSIDs, and signal strength of received packets. The adversary can then analyze the collected data in order to extract further information about the owner of a mobile device.

For example, an adversary can setup sniffing material near a grocery store and monitor shopping habits. By linking MAC addresses and signal strength to different locations inside the store, an adversary can track mobile devices' whereabouts and shopping interests. Worse yet, an adversary can setup multiple sniffing stations and link users across locations, threatening location privacy. An adversary might also infer the identity or social ties of a mobile device owner via analysis of SSIDs in probe requests [6, 7].

3. EXPERIMENTAL EVALUATION

We design an experiment to efficiently collect Wi-Fi probe requests from mobile devices, and then quantify the privacy threat.

3.1 Setup and Assumptions

Because we seek to measure the risk to privacy, we aim to collect as many Wi-Fi probe requests as possible. Collecting *all* wireless

Frame Ctrl	Duration	Destination	Source	BSSID	SEQ	SSID	FCS
...	...	ff:ff:ff:ff:ff:ff	14:10:9F:d5:04:01	ff:ff:ff:ff:ff:ff	...	null	...
...	...	ff:ff:ff:ff:ff:ff	88:30:8a:49:db:0d	ff:ff:ff:ff:ff:ff	...	"PARC Visitor"	...

Table 1: Example of probe requests. One probe request is broadcasted by device with MAC address 14:10:9F:d5:04:01. Another is directed to network with SSID "PARC Visitor" by device with MAC address 88:30:8a:49:db:0d. SEQ is the sequence number of the probe request and FCS is a redundancy check code. Frame Ctrl indicates the type of the frame. For example, probe requests are management frames (type=0x00), with subtype 0x04.

communications from a specific device is difficult for various reasons. First, mobile devices might send their messages on different wireless frequencies over time (over the 11 b/g and 21 n channels), and packet capture software (e.g., TCPDump or Wireshark) must be configured to use antennas and listen to those channels if possible. Second, some packets may be lost due to the noisy nature of the wireless medium (i.e., 802.11 uses an open access CSMA/CA protocol, and collisions occur frequently).

Our setup includes three dedicated Wi-Fi network interfaces each with one antenna connected to a Linux machine. We use existing software (i.e., Aircrack-ng on Ubuntu 14.10 with Linux 3.16) in order to collect Wi-Fi probe requests into .cap files using the three antennas. We developed Python scripts to automatically parse and analyze the collected data. Since the same Wi-Fi probe request might be captured by different antennas, we remove duplicates by compare timestamps of different Wi-Fi probes as well as their sequence numbers. If two packets with the same sequence number are collected within the same burst of probe requests, we consider them as duplicates. We also filter out frames that contain errors (i.e., incomplete or with erroneous timestamps).

We consider all 802.11b/g/n channels (2.4GHz and 5GHz). Note that there are 21 non-overlapping channels in the 5GHz band, making them particularly difficult to monitor.

We consider four different mobile devices: iPhone 6 with iOS 8.1.3, Google Nexus 5 with Android L 5.0.1, Samsung Galaxy S3 with CyanogenMod 11 (based on Android Kitkat 4.4.2), and BlackBerry Q10 with BlackBerry OS 10.3.1. We select these mobile devices because they represent more than 95% of mobile OS usage. For Android devices, we assume that Google Services are enabled.

Metrics. We collect data for exactly one hour in each experiment, repeat each experiment 5 times, and then average our measurements. We measure the total *number of probes* transmitted by mobile devices as it reflects the absolute number of packets broadcasted over the air. We also defines *bursts* as groups of probes sent contiguously over time, and measure the *frequency of bursts*.

3.2 Experiment 1: Capture Configurations

We consider different packet capture configurations that aim at maximizing the number of successfully collected Wi-Fi probes.

Description. We tested six different configurations: **1) 1.dynamic:** One antenna hopping across 802.11b/g channels, other antennas unused; **2) 1.static:** One antenna set to one of three non-overlapping 802.11b/g channels (1, 6, or 11), other antennas unused; **3) 3.dynamic:** Three antennas hopping across 802.11b/g channels in a coordinated fashion (i.e., same hopping sequence shifted by one step); **4) 3.dynamic.s:** Three antennas separately hopping across 802.11b/g channels; **5) 3.dynamic.s.n:** Three antennas separately hopping across 802.11b/g/n channels; and **6) 3.static:** Three antennas each set to a non-overlapping 802.11b/g channel (1, 6, 11).

For simplicity, we focus on one device, namely the Samsung Galaxy S3 with Android Kitkat 4.4.2, and configure it as follows: It is connected to a charger, without any applications running, locked, and with 4 SSIDs in memory.

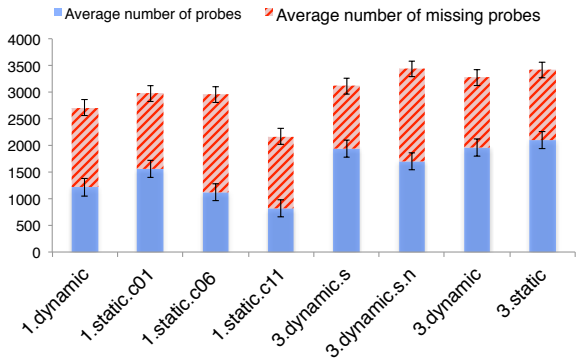


Figure 2: Average number of probes collected with different antenna configurations. The higher the plain blue bar is, the more probe requests are collected. The estimated number of missing probes is shown in dashed red.

Average number of probe requests. In Fig. 2, we show in plain blue the number of probes that were collected in each scenarios. We find that **3.static** collects the largest number of probes (2,100), followed by **3.dynamic** and **3.dynamic.s** (1,900). Interestingly, although the hopping mechanism is designed to cover as much of the spectrum as possible, the static configuration performs better. We believe that this is due to the overlap of 802.11b/g channels: The hopping algorithm ends up monitoring the same overlapping channels with some probability. Another interesting observation is that monitoring the 5GHz band spreads our three antennas-budget too thin, and thus **3.dynamic.s.n** performs worse than focusing on 802.11b/g. Of course, someone equipped with more antennas could capture enough of the non-overlapping 5GHz channels to outperform **3.static**, but this requires investment in sniffing material (up to 21 antennas). In the following, we use **3.static**.

Previous experiments [2, 6, 7, 24] do not appear to have considered the use of multiple antennas for data collection, and are likely to have used a single antenna (corresponding to **1.dynamic**). If so, previous results capture about 57% of **3.static** and probably provide lower bounds on privacy risks of probe requests.

Average number of missing probes. In Fig. 2, we also show in dashed red the estimated number of probes that were not collected (i.e., missed). This allows us to better understand the representativity of our measurements. In order to estimate the number of missed probes, we study the sequence number of probe requests. Specifically, each probe has a sequence number (SEQ in Table 1), that is incrementally increased with each probe request. By counting sequence numbers missing from our collected data, we estimate the number of missing probes. For example, if we collected 3 probes with sequence number 1, 2, 5, we can infer that we missed two probes (3 and 4). We find that **3.static** misses about 39% of probes (i.e., we captured 61% of the probes). One possible explanation is that missing probes were transmitted on 5GHz channels not monitored by our setup. The noisy nature of the wireless medium might be another cause of missing probes.

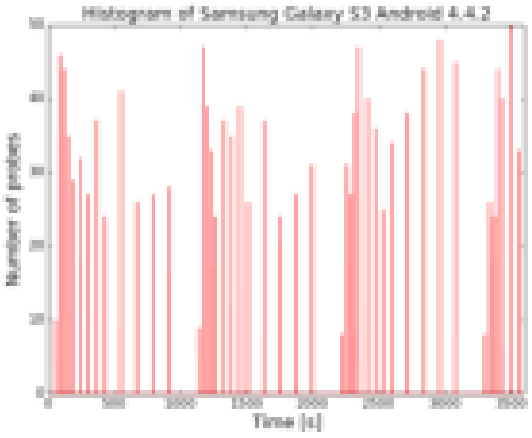


Figure 3: Number of probes broadcasted per minute for Samsung Galaxy S3 with Android Kitkat 4.4.2. Probes are sent in bursts over time, with as many as 50 probes sent in one second.

Probing bursts. In Fig. 3, we show the number of probe requests sent every minute over the course of an hour by the Samsung Galaxy S3 in one experiment. This visualization highlights the *bursty* nature of probe requests. In particular, we observe that probe requests are sent in frequent bursts with a relatively regular pattern. In the following, we study the statistical properties of such probing bursts.

3.3 Experiment 2: Known Networks

We compare the broadcasting rate of mobile devices under different numbers of SSIDs stored in memory.

Description. So far, we considered a conservative number of four SSIDs known to the devices (i.e., stored in memory). We artificially add and remove SSIDs in devices' memory, and manually test whether it affects the frequency and number of transmitted probes. We do so by first erasing all Wi-Fi APs from devices' memory, and then creating a series of 20 Wi-Fi APs with unique SSIDs. We connect and disconnect mobile devices to a number of those APs depending on the experiment. Note that we did not manually input SSIDs, as devices would then consider them as hidden networks. Instead, we connected mobile devices to a number of carefully crafted APs depending on the experiment.

Average number of probe requests. In Fig. 4, we show the number of probe requests sent on average by each mobile device given the number of known SSIDs. The number of probes helps us measure how loud a device is. We find that BlackBerry does not broadcast any Wi-Fi probes, thus providing high privacy.

The iPhone with iOS 8.1.3 broadcasts Wi-Fi probes in the low hundreds, and we see a small increase with 20 known SSIDs. For Android L 5.0.1, the number of broadcasted probes is relatively stable, but significantly larger than the iPhone (about 1500). Android 4.4.2 broadcasts the most probes, and the number of probes seems to depend linearly on the number of known SSIDs. We are able to approximate the number of probes n (dashed line in Fig. 4) as follows: $n = k \cdot l \cdot 13$, where 13 is the number of 802.11b/g channels that we monitor, k is the number of SSIDs stored in memory, and $l = 45$ is the estimated hourly rate of probing. We find that the number of probes deviates from the upper bound with 20 known SSIDs. Another interesting observation is that with 0 known SSIDs, Android 4.4.2 transmits more probe requests that with 1-5 known SSIDs. This is counterintuitive as one would expect that not knowing any SSID is better for privacy.

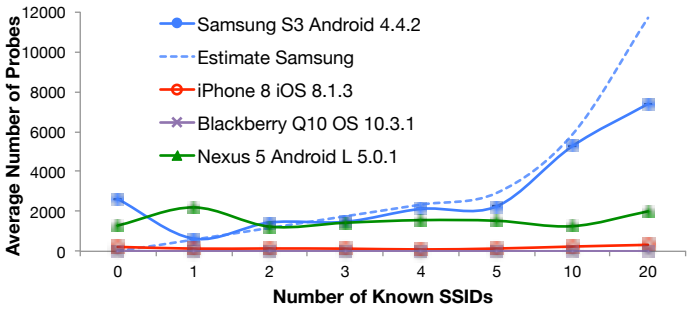


Figure 4: Average number of probe requests against the number of known SSIDs. Android devices broadcast more probes in general. Note the non-linear x-axis.

Fraction of probes with SSID. We measure the fraction of probe requests that contain the names of preferred SSIDs, and find that only Android 4.4.2 broadcasts SSIDs, and does so in 98% of its probes (other devices did not broadcast a single SSID in our experiments). We observe that Android reduced the broadcast of SSIDs to a bare minimum with its latest release (i.e., SSIDs are only broadcasted for hidden networks). This positive development for privacy might come as a result of previous criticism [24]. Nonetheless, according to recent Android developers statistics [1], about 40% of Android devices are on Kitkat 4.4.2 (and many more on older Android distributions), and thus continue to broadcast SSIDs.

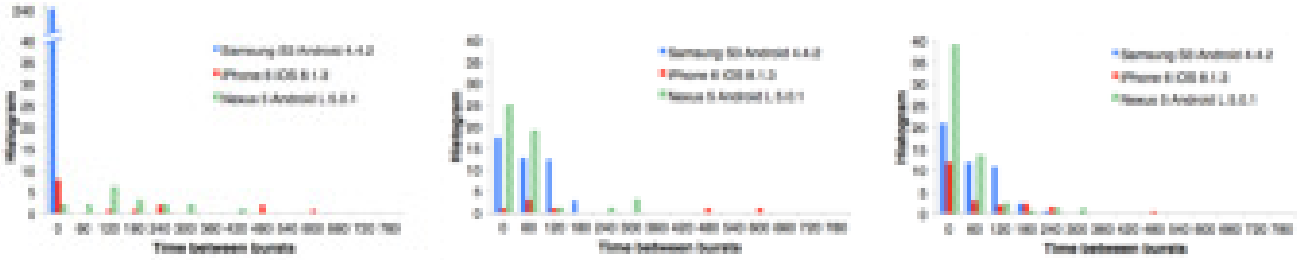
Probing bursts. In Fig. 5, we show the histogram of probing bursts for 0, 4, and 10 known SSIDs. Probing bursts are important as they help measure the ability of an adversary to track devices. With 0 known SSIDs, we observe that Android 4.4.2 is a clear outlier and broadcasts messages with high intensity (as often as every 15 seconds). Second, we observe that as the number of known SSIDs increases, the distribution of probe requests over time changes and approaches an exponential distribution: Probes tend to be sent over short periods of time, and with decreasing probability. Comparatively, the iPhone only shows an exponential property with a high number of known SSIDs.

We measure the mean number of bursts per hour and the mean inter-arrival time of probing bursts. For 4 known SSIDs, we find that Android L 5.0.1 broadcasts 55 bursts per hour on average, one burst every 66 seconds. Android 4.4.2 broadcasts about 45 bursts per hour on average, one burst every 72 seconds. iOS 8.1.3 broadcasts about 11 bursts per hour on average, one burst every 330 seconds. In other words, Android devices have more intense series of bursts compared to iOS 8.1.3. The number of bursts per hour remains relatively constant with different number of known SSIDs.

Another interesting insight is that if we focus on the inter-arrival times of probes, a majority of probing events repeatedly happen within the same second: In our experiments, the distribution of inter-arrival times for Android devices has about 3 bits of entropy, indicating that the timing of bursts is highly predictable. In fact, devices seem to slowly increase the inter-arrival time of bursts in a predictable manner, and reset the inter-arrival time when the OS goes out of sleep. On Android, Google Services are an important factor in generating OS activity, and reseting the probing timer, thus leading to a higher bursting frequency.

3.4 Experiment 3: Device Configurations

We consider a series different mobile device configurations and measure how it affects the number of probes.



(a) 0 Known SSIDs (b) 4 Known SSIDs (c) 20 Known SSIDs

Figure 5: Illustration of the frequency of probe requests given different numbers of known SSIDs.

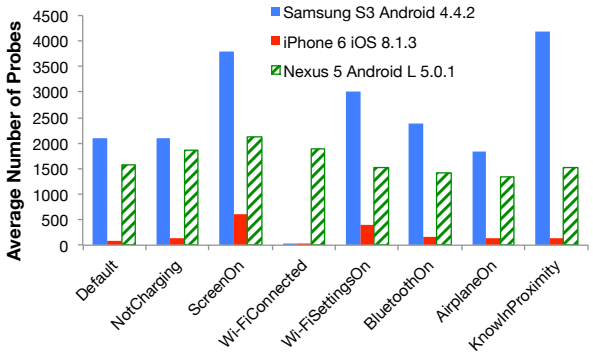


Figure 6: Effect of device configuration on average number of probes for four mobile devices.

Description. We consider nine different device configurations, each a variation of the default configuration: **1) Default:** Bluetooth is off, the device is charging, locked, not connected to a Wi-Fi network, and with 4 known SSIDs; **2) NotCharging:** The device is disconnected from the charger; **3) ScreenOn:** The device is unlocked every 5 minutes and locked again; **4) Wi-FiConnected:** The device is connected to a Wi-Fi network; **5) Wi-FiSettingsOn:** The device is left untouched with the Wi-Fi settings app on; **6) BluetoothOn:** Bluetooth is turned on; **7) AirplaneOn:** Airplane mode is turned on (i.e., no 3G/LTE connectivity), and Wi-Fi is left on; **8) KnownInProximity:** One known SSID appears in proximity of mobile devices every 5 minutes for 10 seconds.

Observations. In Fig. 6, we show the number of probes collected in each device configuration. We observe that **NotCharging** and **AirplaneOn** do not have much of an effect. **Wi-FiSettingsOn** increases the number of broadcasted probes for Android 4.4.2 and iOS 8.1.3, but not for Android L 5.0.1. In contrast, device unlocking (**ScreenOn**) dramatically increases the number of probes for Android 4.4.2 and iOS 8.1.3, and to a lesser extent, for Android L 5.0.1. Once connected to Wi-Fi (**Wi-FiConnected**), most devices stop transmitting probes, except Android L which surprisingly continues broadcasting. This might be used by Android L for faster hand-over between access points, but seems inefficient as probes should only be transmitted when the signal strength of an associated AP weakens.

Finally, we notice that with **KnownInProximity**, Android 4.4.2 has the highest increase in number of probes broadcasted. This is surprising as a known network in proximity should cause an association request, and not an increase in probes. In fact, other devices do not exhibit a significant probing increase. A malicious entity [28] can exploit this behavior by sending forged Wi-Fi beacons, thus pushing Android 4.4.2 devices to reveal their presence.

Figure 7: Illustration of randomized iOS 8.1.3 MAC addresses.

3.5 Experiment 4: Privacy Protection

We test the efficacy of an existing privacy mechanism.

Description. The best privacy mechanism consists in not broadcasting probes requests unless the network is hidden (as done by BlackBerry). To the best of our knowledge, the only deployed alternative to actively protect privacy randomizes MAC addresses.* Researchers in previous work [19] found that randomized frames are used when an iPhone is locked and location capabilities are disabled. We review the iOS 8.1.3 randomization mechanism by monitoring packets sent after randomization.

Re-identifying randomized MACs. In Fig. 7, we show the Wireshark capture of randomized MACs from iOS 8.1.3 devices. First, we find that the MAC address is altered while the device is in sleep mode, and the real MAC address is used when the device is in use. iOS 8.1.3 broadcasts a different random MAC address whenever the device goes into sleep mode. Location services do not need to be turned off for random MAC addresses to be used. Turning off location services makes it more probable that a device enters sleep mode, and uses a random MAC address.

Second, we find that it is easy to detect random MAC addresses. Organizations developing products using Wi-Fi (ISO/IEC 8802 standards) must register to IEEE MAC Address Block Large [14]. Random MAC addresses are thus easy to recognize since they correspond to unassigned MAC addresses. In Fig. 7, we see that Wireshark automatically labels MAC addresses from Apple devices, and does not label random MAC addresses.

Third, careful analysis of sequence numbers (SEQ) in probe requests shows that it is possible to link packets sent by the same device using different MAC addresses. For example, in Fig. 7, device 5a:e3:24:ea:35:4a broadcasts a probe with SEQ=1039 and shortly after device Apple_51:2d:db broadcasts a probe with SEQ=1040. In other words, the incremental SEQ increase indicates that both packets might originate from the same device. It is also possible to use vendor specific information included in probes to link different packets to a device, or to identify a device's brand. For example, 802.11n defines an aggregation process to group packets together rather than transmitting them separately (Aggregated MAC Protocol Data Unit (A-MPDU)). This is advertised in probe requests and tends to be different across devices from different brands.

*Bluetooth 4.2 uses random MAC addresses and passive network discovery.

We checked all MAC addresses in our collected data against the published list of assigned MACs [14] to identify potentially anonymized MAC addresses, and then used the sequence number information as well as timing information to re-identify randomized MAC addresses. Specifically, if a MAC address is detected as random and appears more than three times with a sequence number close to a real MAC address, it is flagged as a randomized MAC corresponding to the real MAC address. By doing so, we were able to automatically link randomized probes to real probes in several instances. In the series of experiments with iOS 8.1.3 in default configuration, we captured on average 121 probes with true MAC address, and could re-identify 16 randomized probes on average.

4. CONCLUSION AND FUTURE WORK

We quantify the threat to user privacy posed by Wi-Fi probe requests. Concretely, we consider several mobile devices under different configurations and measure the number of broadcasted probe requests. We reach a number of worrying conclusions. Our measurements show that third parties can monitor the whereabouts of certain devices with high precision. In our experiments, Android devices are particularly at risk as they broadcast probes at high frequency (approximately every minute). Among our surprising results, we find that it is possible to re-identify iOS 8.1.3 randomized MAC addresses using sequence numbers in probe requests. We also find that Android L 5.0.1 devices broadcast probe requests while connected to an access point, and that the number of probes broadcasted by Android 4.4.2 devices linearly depends on the number of SSIDs it has in memory. On a positive note, SSIDs are not systematically broadcasted in latest OS versions, and BlackBerry smartphones rarely broadcast probes.

While carrying out our work, we notice that the probing behavior of a given device significantly depends on the OS version, e.g., an iPhone 6 with iOS 8.3 probes more frequently than with iOS 8.1.3. This means that our results are indicative of a spectrum of probing behaviors. It also means that mobile device manufacturers have the technical means to change probing behavior, and improve user privacy. In the meantime, privacy-conscious users might be wise to turn off their Wi-Fi interface when not in use.

Our Python scripts are available upon request for others to run their own experiments and expand the set of covered devices and configurations (e.g., app in use, user mobility [13], laptops...). In the future, we will consider active attacks that stimulate the broadcast of probes (similar to **KnownInProximity**). We will also further examine the ability to re-identify probes with randomized MACs, exempli gratia, taking into account user mobility patterns.

Acknowledgements. The author wishes to thank Shantanu Rane, Alejandro Brito, Emiliano De Cristofaro, and anonymous reviewers for their useful comments.

References

[1] Android. Developer Dashboards. https://developer.android.com/about/dashboards/index.html, 2015.

[2] M. V. Barbera, A. Epasto, A. Mei, V. C. Perta, and J. Stefa. Signals from the crowd: uncovering social relationships through smartphone probes. In *IMC*, 2013.

[3] A. R. Beresford and F. Stajano. Mix zones: User privacy in location-aware services. In *Pervasive Computing and Communications Workshops*, 2004.

[4] B. Bonné, A. Barzan, P. Quax, and W. Lamotte. WiFiPi: Involuntary tracking of visitors at mass events. In *WoWMoM*, 2013.

[5] N. Cheng, X. Wang, W. Cheng, P. Mohapatra, and A. Seneviratne. Characterizing privacy leakage of public WiFi networks for users on travel. In *INFOCOM*, 2013.

[6] M. Cunche, M. A. Kaafar, and R. Boreli. I know who you will meet this evening! linking wireless devices using Wi-Fi probe requests. In *WoWMoM*, 2012.

[7] M. Cunche, M. A. Kaafar, and R. Boreli. Linking wireless devices using information contained in Wi-Fi probe requests. *Pervasive and Mobile Computing*, 2014.

[8] Euclid Analytics. Answers and insights for physical locations. http://euclidanalytics.com/, 2015.

[9] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. V. Randwyk, and D. Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *Usenix Security*, 2006.

[10] J. Freudiger, M. Raya, M. Félegyházi, P. Papadimitratos, and J.-P. Hubaux. Mix-zones for location privacy in vehicular networks. In *Workshop on wireless networking for intelligent transportation systems (Win-ITS)*, 2007.

[11] J. Freudiger, R. Shokri, and J.-P. Hubaux. Evaluating the privacy risk of location-based services. In *FC*. 2012.

[12] B. Greenstein, D. McCoy, J. Pang, T. Kohno, S. Seshan, and D. Wetherall. Improving wireless privacy with an identifier-free link layer protocol. In *MobiSys*, 2008.

[13] M. Humbert, M. H. Manshaei, J. Freudiger, and J.-P. Hubaux. Tracking games in mobile networks. In *GameSec*. 2010.

[14] IEEE. MA-L PUBLIC LISTING. http://standards.ieee.org/develop/regauth/oui/public.html, 2015. ISO/IEC 8802 standards.

[15] IEEE Standard Association. IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY). 2012.

[16] Y. S. Kim, Y. Tian, L. T. Nguyen, and P. Tague. LAPWiN: Location-Aided Probing for Protecting User Privacy in Wi-Fi Networks. *S&P poster*, 2013.

[17] B. Konings, C. Bachmaier, F. Schaub, and M. Weber. Device names in the wild: Investigating privacy risks of zero configuration networking. In *Mobile Data Management (MDM)*, volume 2, 2013.

[18] J. Lindqvist, T. Aura, G. Danezis, T. Koponen, A. Myllyniemi, J. Mäki, and M. Roe. Privacy-preserving 802.11 access-point discovery. In *WiSec*, 2009.

[19] B. Misra. iOS8 MAC randomization - Analyzed! http://blog.airtightnetworks.com/ios8-mac-randomization-analyzed, 2014. Airtight Networks.

[20] Motorola. Analysis of iOS 8 MAC Randomization on Locationing. *White Paper*, 2014.

[21] A. Musa and J. Eriksson. Tracking unmodified smartphones using Wi-Fi monitors. In *SenSys*, 2012.

[22] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *S&P*, 2008.

[23] Path Intelligence. Revolutionary technology for detailed data insights. http://www.pathintelligence.com/technology/, 2015.

[24] Peter Eckersley and Jeremy Gillula. Is Your Android Device Telling the World Where You've Been? http://goo.gl/3XezqR, 2014. EFF, 2014.

[25] Sensepost. Snoopy: Distributed Tracking and Profiling Framework. http://research.sensepost.com/conferences/2012/distributed_tracking_and_profiling_framework, 2012.

[26] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux. Quantifying location privacy. In *S&P*, 2011.

[27] K. Thilakarathna, H. Petander, J. Mestre, and A. Seneviratne. MobiTribe: Cost Efficient Distributed User Generated Content Sharing on Smartphones. *IEEE Transactions on Mobile Computing*, 2013.

[28] WiFi Pineapple. Mark V Standard. https://hakshop.myshopify.com/products/wifi-pineapple, 2015.