KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Characterizing Wi-Fi Network Discovery

Yun-Chih Joy Chou

Thesis submitted for the degree of
Master of Science in
Electrical Engineering, option
Embedded Systems and Multimedia

Thesis supervisor:
Prof. dr. Claudia Diaz

Assessors:
Prof. dr. ir. Bart Preneel
Prof. dr. ir. Sofie Pollin

Mentors:
Dr. Gunes Acar
Rafael Gálvez Vizcaíno

Academic year 2016 – 2017

Preface

"When the going gets tough, the tough get going."

It's lots of tears and joy studying abroad in Leuven, Belgium. I will forever cherish the life here. So many thanks to the daily supervisors Gunes and Rafa for the tremendous help on my thesis, which is more than I can ask for. Thanks to Professor Claudia Diaz for giving me the opportunity to work on this thesis as well, even though it has already been booked. Thanks to the secretary Anne Ons for reserving the thesis for me and arranging the defense dates. And thanks to the assessors Professor Bart Preneel and Professor Sofie Pollin for investigating into my thesis besides their busy work. Thanks to the coordinator Professor Marc Moonen for adding my intermediate and final defenses in the tight schedule. And thanks to Mathy, Frederik, Edu, and others that gave helpful comments. Finally many thanks to my dear Yamu for supporting and accompanying me always. Let's fulfill God's perfect will together.

Yun-Chih Joy Chou

Contents

Abstract

In the Wi-Fi network discovery phase, the probe request sent by a mobile device is not encrypted. Therefore, the MAC address contained in the probe request frame can be exploited to track users without their knowledge and consent. To counter this problem, modern operating systems implement MAC address randomization. However, studies showed that the spoofed MAC address can still be reversed. In addition, studies showed that some devices send the Service Set Identifiers (SSIDs) of the previously connected networks. Such SSID list reveals sensitive personal information such as the locations one has been to. In order to help understand how much the probe requests reveal the behavior or traces of a user, this thesis characterizes the scanning behavior of the mobile device during network discovery. The different scenarios we tested include different numbers of the saved networks, different connection status, and mobile configurations. The characterized parameters include the channel sequence order, the number of probe request frames sent per channel, the interprobe delay time, and the minimum and maximum channel time a device spent on each channel (`MinChannelTime`, `MaxChannelTime`). We test five different devices and find that all of them send one or two probe requests per channel. All devices except the Windows PC set the same value for the minimum and maximum channel time, which ranged from 20 ms to 105 ms. Our experiments show that MacBook prioritizes the channels of the saved networks, which reduces the connection time. Our results show that devices exhibit different scanning behavior in different settings. We found that three of the five devices tested used randomized MAC addresses, and almost none of them sent the SSIDs of the previously connected networks. We also observed that, some devices with MAC randomization reveal their true MAC addresses once connected to a network.

Keywords: *Wi-Fi characterization, Wi-Fi scanning, probe request, SSID, MAC randomization, MinChannelTime, MaxChannelTime*

List of Figures and Tables

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

AP	Access Point
BSS	Basic Service Set
BSSID	Basic Service Set Identifier
DFS	Dynamic Frequency Selection
ESS	Extended Service Set
ESSID	Extended Service Set Identifier
IE	Information Elements
MAC	Medium Access Control
MaxCT	Maximum channel time, the `MaxChannelTime` in the protocol
MinCT	Minimum channel time, the `MinChannelTime` in the protocol
MLME	MAC Layer Management Entity
OS	Operating System
OUI	Organizationally Unique Identifier
PC	Personal Computer
PHY	Physical Layer
PRF	Probe Request Frame
SN	Sequence Number
SRD	Short Range Devices
SS	Service Set
SSID	Service Set IDentifier
STA	Station
TPC	Transmit Power Control
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Netowrk
WPA	Wi-Fi Protected Access

Symbols

ms	Millisecond
GHz	Gigahertz

Chapter 1

Introduction

1.1 The problem

Nowadays many people use Wi-Fi to connect to the Internet. It is one of the most important standards deployed in mobile devices as well as laptops. When a device starts to connect to the access point (AP) for the Internet, it sends the wireless packets in different phases. The phases include network discovery, authentication, association, distribution etc. In the network discovery phase, the device scans in different channels for the available APs. The packet sent by the device is the probe request. The purpose of the probe request is to ask the APs to reply with the probe response, so that the device can have a list of available networks and choose one to authenticate and associate with it. The probe request is not encrypted, therefore anyone who receives the probe request frame can interpret it. This is where the problem comes from.

In a probe request, there are some contents that may reveal the private information. The address of the Medium Access Control (MAC) layer is sent in every probe request frame. It means that the address can be used as a method to track the person who uses the device (Cunche et al. [16]). Also, there are shops that can tell if a customer visits for the first time by checking the MAC address (Soltani [27]). Because of this, randomized MAC addresses have been introduced and implemented, but most of the devices can still be identified by combining other information in the probe request (Martin et al. [23]).

Another parameter in the probe request is the Service Set IDentifier (SSID). The SSIDs are the available network names shown in the scan result on the Wi-Fi setting page, like 'MacDonald free WiFi'. The device can send a broadcast probe request that has no specific SSID, or contains a name from an AP that it connected to. This may be a problem when the device is probing with the history SSIDs. By these names it is possible to infer the social relationships of the people showing up together (Cunche et al. [15]).

There are a few papers that studied the parameters and the number of probe requests sent by the devices (Arcia et al. [10], Castignani et al. [13]). They suggests some possible schemes to optimize the scanning process. But there has not been any

paper that focuses on characterizing the probe request of the devices.

1.2 Why is the problem important?

The more probe requests are sent, the more MAC addresses are revealed. Therefore it is important to reduce the number of probe requests in every scan. Also it is important to see which devices are sending the SSIDs of the previously connected networks to suggest any improvement. How do the different probe requests in a scan relate to each other? Is the way of scanning unique in every device? Answering these questions helps us to understand how serious the probe requests leak personal information.

1.3 Objectives

The objective of the thesis is to characterize the scanning behavior of the mobile devices, namely the non-AP stations (STA). Besides focusing on the MAC randomization and whether the device sends the history SSIDs, we would like to measure the following parameters: The sequence order of the scanned channels, the number of probe requests per channel, the interprobe delay, and the channel time parameters described in the protocol (`MinChannelTime` and `MaxChannelTime` etc). The tests in different scenarios include the default mode (when not connected, and no saved network), different connection status (connected, not connected, or known in proximity), different numbers of the saved networks, and different configurations of the mobile.

1.4 The contribution

The contributions of the thesis are:

- We have characterized the scanning of devices from different brands (MacBook, Windows PC, Linux, Samsung mobile, and iPhone). The characteristics of probe requests from different devices are different, so they can be used to categorize the devices. The probe request sequence usually starts from channel 1 and moves to the next channel after 20-100 ms. One exception is MacBook. It does priority scanning starting from different channels according to the channels of the saved networks.

- Among the devices we tested, almost all devices don't send the SSIDs of the saved networks. Only MacBook seems to send the newly-established SSIDs of the newly-established networks within an amount of time.

- Some devices use the randomized MAC addresses when not connected to the networks. We found that MacBook and iPhone scan using the global MAC address when they are connected (in the Wi-Fi setting page), or when they are about to connect to the known network.

- The characterization of the scanning behavior helps make identifying the devices easier. There is yet room to strike a balance between making the scanning efficient as well as secure.

1.5 Overview

The thesis is structured as follows:

Chapter 2 introduces the background of the research, mainly the IEEE 802.11 protocol that describes the scanning behavior, and the parameters contained in a probe request and in the MAC Layer Management Entity.

Chapter 3 describes the related work that is most relevant to this study.

Chapter 4 describes the scenarios of different tests, the setup, recording, and the post processing of the data.

Chapter 5 presents the result from the measurement, and Chapter 6 concludes the thesis and mentions the future work.

Chapter 2

Background

This chapter introduces Wi-Fi, the scanning procedures and the parameters mentioned in the IEEE 802.11 protocol during the Wi-Fi network discovery phase. In addition, it presents a probe request frame sample from a mobile device.

2.1 Wi-Fi and Network Discovery

There are several ways to connect to the Internet without a cable. Wi-Fi is one of them. The Wireless Fidelity (Wi-Fi) is the most common type of the Wireless Local Area Network (WLAN). It is described by the IEEE 802.11 Protocol. The protocol describes the specifications in the MAC and physical (PHY) layers of what every device needs to comply when communicating with each other using the WLAN. The protocol itself evolves over time to include more advanced technologies. There are several versions of the protocol from 1999 to 2016 [5][6][7][8]. The most common frequencies used in the Wi-Fi are 2.4 GHz and 5 GHz.

When a mobile device (i.e. a station or STA) connects to the Wi-Fi, it can be in different networks. There are infrastructure, ad hoc, and mesh networks. In our thesis we focus on the infrastructure network. In the infrastructure network, there is an AP that transfers the packets transmitted through the Internet cable into the wireless packets. Every device connects to the AP for Wi-Fi. They form a basic service set (BSS). The AP constantly sends the beacon frame signals for the mobile device to synchronize with it. Also the beacon frame is constantly sent for the mobile devices to discover such network. The beacon frame include the timestamp and the SSID of the AP. When a mobile device wants to connect to any external Wi-Fi network, it does the interworking and launches the network discovery process.

2.1.1 Active and Passive scanning

There are two types of network discovery processes: Active and passive scanning. In the active scanning, a mobile device can choose to send a broadcast **probe request** to search for available APs in a channel. If any AP receives the probe request and replies, a probe response is sent to the mobile device. In this case the mobile device

can start to authenticate and associate with the AP. The active scanning requires the mobile device to take the initiative to send a probe first.

In the passive scanning scenario, the mobile device doesn't scan at all but passively wait for beacons in a channel. If the mobile device receives the beacon frame from the AP, it can start to authenticate and association messages to that AP. This connection requires the AP to send beacons first, and is therefore considered more time-consuming and thus more power-consuming than the active scanning [29]. Nevertheless, such passive scanning has its advantages that it does not broadcast its MAC address which can be tracked without users' awareness.

In the 5 GHz band, not all channels can be actively probed. There are channels that the mobile device can scan only when there is no other radar operating in the channel. To avoid interference with the radars, the mobile device or AP has to implement the total power control (TPC) mechanism to do the dynamic frequency selection (DFS).

2.2 Scanning Parameters

2.2.1 MLME-SCAN.request

The table 2.1 lists the primitive parameters contained in the scan request of the MAC Layer Management Entity (MLME) of the mobile device. This primitive requests a survey of potential basic service set (BSS, in this case the external networks) that the mobile device can later choose to join. The numbers of the parameters enlarge throughout different versions of the protocol.

These parameters are stored in the mobile device, and not all parameters are contained in the probe request sent over the air. Below describes the most important parameters contained in MLME.SCAN-request that affect how the mobile device scans and the content in the probe request.

ProbeDelay, MinChannelTime, MaxChannelTime

Among the list of parameters contained in MLME.SCAN-request, the following three parameters determines how much time the mobile device waits in a channel: `ProbeDelay`, `MinChannelTime`, and `MaxChannelTime`. The `ProbeDelay` is the time the mobile device waits before sending the probe request ([8] Page 266). `MinChannelTime` and `MaxChannelTime` are the minimum and maximum time the mobile device has to spend in a channel ([8] Page 266). If the mobile device receives no probe responses within the minimum channel time (MinCT), it will switch to another channel and start scanning in that channel. Otherwise, if there is at least a probe response within MinCT, the mobile device waits up to the maximum channel time (MaxCT) for more probe responses.

Figure 2.1 shows how the mobile device behaves in the active scanning network discovery phase. In the figure, the dwell time in the channel 1 is the MinCT, where there is no probe response received by the mobile device. In channel 2 the dwell

Table 2.1: Scan Parameters in the MLME-SCAN.request

Standard version	MLME-SCAN.request	Numbers of attributes	Newly added
1999(2003)	BSSType,BSSID, SSID,ScanType, ProbeDelay, ChannelList, MinChannelTime, MaxChannelTime	8	All
2007	BSSType,BSSID, SSID,ScanType, ProbeDelay, ChannelList, MinChannelTime, MaxChannelTime, VendorSpecificInfo	9	VendorSpecificInfo
2012	BSSType,BSSID, SSID,ScanType, ProbeDelay, ChannelList, MinChannelTime, MaxChannelTime, RequestInformation, SSID List, ChannelUsage, AccessNetworkType, HESSID, MeshID, VendorSpecificInfo	15	RequestInformation, SSID List, ChannelUsage, AccessNetworkType, HESSID, MeshID,
2016	BSSType,BSSID, SSID,ScanType, ProbeDelay, ChannelList, MinChannelTime, MaxChannelTime, RequestInformation, SSID List, ChannelUsage, AccessNetworkType, HESSID, MeshID, DiscoveryMode, VendorSpecificInfo	16	DiscoveryMode

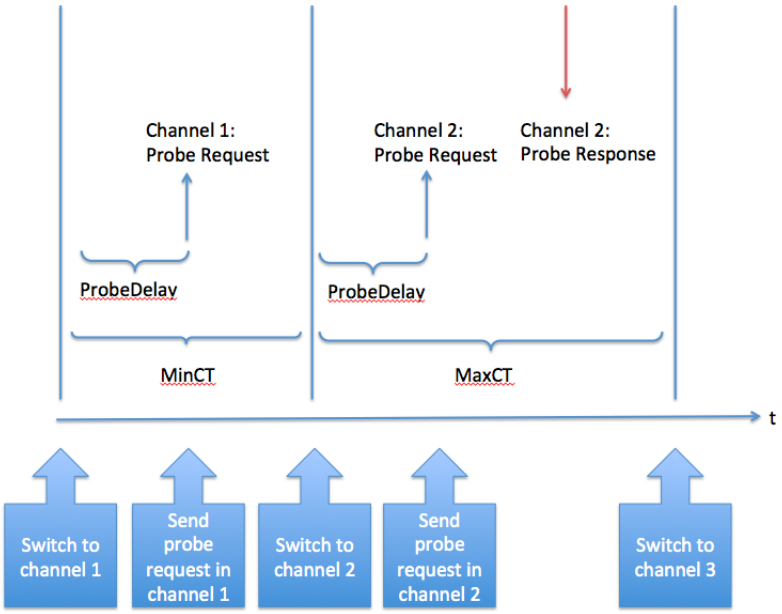


Figure 2.1: Overview of the ProbeDelay, MinCT, and MaxCT

time becomes MaxCT, because a probe response has been received during scanning[1]. In reality the dwell time can be larger than the MinCT or MaxCT, because it can also take some time for the mobile device to configure to the desired channel. Note that in the figure the mobile device sends only one probe per channel. This is a manufacturer factor and there can be more than just one.

SSID

The Service Set Identifier (SSID) is the identifier of the Service Set (SS). In an infrastructure network, the SS can be a Basic SS (BSS) or Extended SS (ESS). The identifier of the BSS (BSSID) is the name of the AP. For ESS, the identifier of the ESS (ESSID) is the common name of the APs. The identifier of the AP is often a readable name. For example there is an ESSID called `eduroam`.

In many mobile devices, a list of SSIDs is stored to allow auto-connection to the previously connected networks. By reading the history SSIDs, one can tell the places the user has been to, the attended conferences, or even the owner name of a mobile device that shares the Internet connection. Therefore the SSID list keeps the personal information that one doesn't want to distribute publicly.

[1] There can be two interpretations whether the `MinChannelTime` includes the `ProbeDelay`. In the revised standard published in 2012, the relationship "`MinChannelTime >= ProbeDelay`" has been removed, which implies the more general description that both the interpretations can be valid. In this thesis, we assume that `MinChannelTime` includes the `ProbeDelay`.

VendorSpecificInfo

The `VendorSpecificInfo` describes the manufacturer of the mobile device. It is also contained in the probe request sent by the mobile device.

2.2.2 Probe Request Frame

The probe request frame (PRF) is the packet sent by the mobile device in the active scanning. Some parameters are different from that in the `MLME-SCAN.request`. The parameters included in the probe request are listed in table 2.2. Note that not all parameters are present in a probe request frame. Most of them are optionally present under some conditions. For example, the SSID List which contains all the history saved networks will only be included in the probe request frame when `dot11SSIDListActivated` is true.

As for how to switch between the channels, the scanning sequence is not described in the protocol. This part will be discussed in the next chapter (related work).

2.2.3 MAC Address and Sequence Number

In every MAC frame transmitted, the MAC address of the mobile device is included. The MAC address is like an identifier of the mobile device. There are universal and locally administered MAC addresses. When the MAC address is universal (or global), there is only one device using the address. When the address is locally assigned, there can be more mobile devices that share the same address. The MAC address is a 6 octet string such as aa:bb:cc:dd:ee:ff. The first three octets is the Organizationally Unique Identifier (OUI). The OUI indicates the manufacturer of the device. From the first three octets of the MAC address, we know from which company the mobile device belongs to. This is not the case for the randomized MAC address deployed in some mobile devices. Some manufacturers like Apple use the spoofed MAC addresses over the entire MAC addresses. Therefore it is not possible to tell which company the mobile device belongs to. The randomized MAC has been introduced because it becomes sensitive to send the identifier of a mobile device in every frame it sends. This may reveal some privacy information like the location of the user.

The Sequence Number (SN) is a compulsory parameter added in every transmitted frame of a mobile device. It acts as a counter. The value is from 0 to 4095, increasing one in every frame. Adopting this counter makes it easy to group the frames of a mobile device among different frames by different mobile devices.

2.3 Sample probe request frame

This section shows the captured probe request sent by an iPhone 6S Plus (iOS 10.2.1) to illustrate how the mobile devices implement the protocol. The packet is interpreted using the free and open source network protocol analyzer called Wireshark [4].

Table 2.2: A sample Probe Request Frame from iPhone

Order	Information	Value
1	SSID	broadcast
2	Supported Rates and BSS Membership Selectors	1,2,5.5,11 (Mbit/s)
3	Request	
4	Extended Supported Rates and BSS Membership Selectors	6,9,12,18,24,36,48,54 (Mbit/s)
5	DSSS Parameter Set	Current channel: 6
6	Supported Operating Classes	
7	HT Capabilities	802.11n D1.10
8	20/40 BSS Coexistence	
9	Extended Capabilities	8 octets
10	SSID List	
11	Channel Usage	
12	Interworking	Access Network type: Wildcard HESSID: Broadcast
13	Mesh ID	
14	Multi-band	
15	DMG Capabilities	
16	Multiple MAC Sublayers	
17	VHT Capabilities	IEEE Std 802.11ac/D3.1
18	Estimated Service Parameters	
19	Extended Request	
20	Vendor Specific	Apple, Microsof [*sic*], Broadcom

Table 2.2 lists the parameters contained in the probe requests from iPhone, corresponding to the parameters mentioned in the protocol. About half of the parameters are listed, according to different setting in the mobile device.

Chapter 3

Related Work

3.1 Probe Request Application and Possible Mischievous Use

In [16] the author Cunche mentioned that when a device turned on the Wi-Fi, it acts like a wireless beacon that is periodically advertising its identifier, i.e. MAC address. This is the starting point to track or exploit the device. In [14] Cheng et al. began to use the probe requests as the means for Wi-Fi positioning, with an accuracy of about 15 meter in urban areas. Musa et al. [25] tracked the devices by capturing the Wi-Fi frames sent by the devices that passed by. To stimulate more probe requests, they set an AP with two popular SSIDs that many devices have associated with. In their captured data, they detected mostly iPhones and characterize the scanning frequency in different configurations. Sharabani, the author of [2] pointed out the possible man-in-the-middle attack by creating a rogue AP with a common SSID. The devices that automatically connect to the saved networks will fall into victim. The author further found out that iPhone had a stored SSID list before the user ever saved any in it. This means that the attack can have a higher chance of success if using the saved names in all iPhones.

Tracking using the Wi-Fi probe requests has been exploited as a means of marketing. The Euclid analytics (euclidanalytics.com) company has been selling and designing the analytical tools to the shops as well as organizations. Soltani in [27] gave a few examples of how the opt-in and opt-out can make a difference in the commercial aspect.

In [17] Cunche et al. described a model to infer the social relationship between two persons by examining the probe requests sent from their mobiles. They assumed that the more socially linked two persons are, the more similar their history SSIDs will be. This research nevertheless did not indicate how successful the prediction was after applying on the dataset. It seemed that for most of the case, the intersection of two devices is just one SSID.

In [19] Gentry et al. demonstrated how the information received in the probe request and responses can be used to identify the model and manufacturer of a device. The signature post-processed after receiving the probe requests contains all

the parameters in the probe request frame. The number of parameters increase in later versions of the protocol. This makes the signatures of all the devices more diverse and in many cases distinguishable.

3.2 MAC Randomization Reversal

MAC randomization is a way to protect the devices from sending the real MAC addresses and thus being tracked. The MAC address is randomized and it is more difficult for the attacker to find the real MAC address and link the device. However, in [28] Vanhoef et al. found that it is possible to reverse the randomized MAC and fingerprint the device. By combining the Information Elements (IE) and the SN of the probe request frame, it is easy to group the packets of the device even though they use different MAC addresses. Further more, they found that the devices change to real MAC when they are connected to an AP. So they set up a fake AP and allow the devices to connect to them. In this way they get the real MAC of the devices. Also they found that Linux and Windows send the Access Network Query Protocol (ANQP) using the real MAC.

In [24] Matte et al. found that the signature based on Inter-Frame Arrival Time (IFAT) from the probe requests can be used to identify the devices, even though the devices have randomized MAC. They could identify the packets sent from a device in 75% of the packets.

In [23] Martin et al. did a thorough study to reverse the randomized MAC addresses from different devices. The corpus they used was recorded in 2 years, and contains 2.6 million MAC addresses. They pointed out that locally assigned MAC randomization of some devices do not possess the one-to-one relationship so they focus on the devices that have the global MAC addresses. After detailed testing, they confirmed that there are at least three kinds of methods to retrieve the global randomized MAC addresses.

Robyns et al. in [26] gives the latest update about fingerprinting devices using the IE in the probe request frames or other packets. They collected some bits in the probe request frame and used them to identify the devices. They also discussed in details about the useful parameters in the protocol that can stimulate the devices to send more frames to help tracking.

3.3 Scanning Characterization

Freudiger et al.'s work [18] is the most relevant study to this thesis. They measured the number of probe requests from several mobiles. First, they counted the number of probes under different capture configurations. The configuration of three antennas set at 3 non-overlapping channels captured the most probes than other configurations. Then they compared the number of probe requests from different numbers of the saved networks. They found that the Android phones sent the most probes as the number of saved SSIDs increased. They also measured the probes of different device configurations like charging or Bluetooth on. The configuration that had

the most probes is when there is a known AP in proximity. Note this is not really the configuration from the mobile, but the different kind of environment scenario when the device is put to test. They found that Android 4.4.2 would sent a lot of probes in the `KnownInProximity` mode. This paper also mentioned that it was easy to identify the randomized-MAC device.

Gupta et al's work [21] is the early study on characterizing the Wi-Fi scanning, thus is also very related to our study. They listed the scanning parameters that was not described in the 802.11 protocol:

- Channel on which the 1st PRF is sent.
- Number of PRFs transmitted per channel (Burstiness).
- Delays between PRFs on the same channel.
- Frequency of channels probed (Number of Probes sent on a channel).
- Order of channels probed.

Along with the `ProbeDelay`, MinCT, and MaxCT, these parameters can only be known by analyzing the recorded probe requests. From the six wireless network interface cards they measured, they observed that most of the first PRFs are in channel 1,7,9, instead of the more commonly used non-overlapping 1,6,11 channels. The recorded time spent in each channel was referring to the dwell time, instead of `ProbeDelay`, MinCT, or MaxCT, probably due to the difficulty of breaking down the measured time into these parameters.

In [9] Arcia et al. measured the response delay of Linux. The response delay is the time difference of the probe request and the probe response. They implemented a different scanning algorithm in the Linux kernel. The result showed that the response delay changes for different APs.

3.4 Scanning Optimization

In [30] Wu et al. found that in a mobile, the scanning before the association is the main power consumption in the power saving mode. Thus they proposed not to scan if the mobile does not move beyond 10-20 meters. They implemented the cellular assisted movement detection in Windows phones and showed that 90% of the power has been saved.

In [31] Xia et al. proposed to consider adding the location of the device and only turn on the Wi-Fi when the user is close to an AP. The locations of available APs can be retrieved by a few means. They showed that this reduced the energy of the unnecessary scans.

In [22] Kim et al. proposed a location-aided mechanism that aims to reduce the probe requests sent by the device. By comparing the current location of the device and the locations of the history saved networks, the device will send the SSID with the saved network name only when the current location is in the proximity of

that saved network. They implemented the mechanism in the `wpa_supplicant` of Android 4.2 and reduced the number of SSIDs sent by the device.

In [12][13] [11] Castignani et al. discussed how the values of MinCT and MaxCT play a role in the network discovery process. The two values are fixed among all devices. They proposed to set the values adaptively, based on the availability of the APs found in that channel. They simulated, tested and confirmed that the adaptive timer has the better performance than the fixed timer, which can strike a good balance between the full scanning latency and the full scanning failure.

In [10] Arcia et al. found that a device can find only partial APs in a busy environment in one scan, so it requires a few scans to have a complete list of available APs. They listed in detail how they setup the equipment, change the channel time, and tested Android and Linux devices. They proposed that it saves the scanning time if the devices can share discovered AP information in a database.

In [20] Goovearts et al. modified the `wpa_supplicant` source code to change the scanning parameters of a Android phone. They modified the probe delay and channel time, and change the channel order by scanning the popular channels first, which are the non-overlapping channels 1,6, and 11. The available networks showed up in the list a lot more quickly.

Chapter 4

Data Collection

4.1 Mobile Device Selection

We chose a few common phones and PCs as our testing mobile devices. The cellphones are iPhone, Android (Samsung). The PCs are Windows, Linux, and MacBook. The OS and model version of each device is in table 4.1.

4.2 The focus and test scenarios

We aim to characterize how each mobile device scans under different scenarios. There are attributes which may affect the scanning of the device. First, the device configuration (active or sleep, charging or not, connected to APs or not etc.). Second, the saved networks (SSID history) in the device. Third, the environment that has known or unknown APs (known in proximity or not).

For the scanning, there are some parameters to be tested (the list is mostly from Gupta et al.'s paper [21]):

- Channel sequence
- Channel of the first probe request frame (PRF)
- Number of PRF per channel

Table 4.1: Tested Devices

Type	Brand	Model	OS
Mobile	Samsung	S3 Mini	Android 4.1.2
Mobile	Apple	6S Plus	iOS 10.2.1
PC	Dell	Latitude E7470	Windows 10
PC	Dell	Latitude E7470	Ubuntu 16.04
PC	Dell	Latitude E5430	Ubuntu 14.04
PC	Apple	MacBook Air	MacOS 10.12.4

Table 4.2: List of the captured channels from the tuned channel

Tuned channel	Covered channel(s)
2	1,2,3
5	4,5,6
8	7,8,9
11	10,11,12
13	12,13

- Interprobe delay in each channel (when there are more than 2 probes per channel)
- MinCT
- MaxCT

The first test scenario is how the devices scan under the default mode. The default mode for the cellphone is:

- screen on at the Wi-Fi setting page / not charging / not connected to any AP / no saved SSID

For PCs, they have the same default mode as the cellphones, expect that they are not at the Wi-Fi setting page, but that the Wi-Fi has been turned on and off manually for every few seconds to stimulate more bursts (a set of probe requests scanned continously). The second test scenario is to see whether different configurations of the same device will change the scanning behavior. The default setting is that the device has no saved SSID. The third test scenario is to check the scanning behaviors when the device is connected to the network, known in proximity (not connected), or not connected to the network. Note the known-in-proximity mode is considered a kind of the mobile configurations in [18]. In this thesis we consider it as a connection type and compare with other connection status. The fourth scenario is to check if the numbers of the saved networks affect the scanning when the device is not connected to any network. The last scenario is to check if the APs affect the device.

4.3 Measurement

4.3.1 General Setup

For 2.4 GHz band, We captured the 13 channels at once by using the 5 antennas tuned to 5 different channels. The reason is that each antenna captures not only the tuned channel, but also the adjacent channels. So while tuning them to channel 2,5,8,11,13, each antenna can capture at least 3 channels. The mapping of each channel is listed in the table 4.2.

The probes from the adjacent channels are captured in most of the devices we tested except in Windows, where the data rate is 6 Mb/s or 12 Mb/s, higher than

Figure 4.1: Measurement setup for the 2.4 GHz band

others (Linux is 1 Mb/s), so it is more difficult for the antenna to pick up the adjacent channels in this setting. Hence, for the case of Windows we record the information in three parts: channel 1-5, channel 6-10, and channel 9-13.

For the measurement in the 2.4 GHz band, our setup consists of five antennas, a Linux PC, and the USB hub to connect all the antennas to the PC (please see Figure 4.1.) For the measurement of 2.4 GHz with the 5GHz band, we add an extra 5 GHz antenna to capture the 5GHz signal.

We used a few Linux command line tools to record the signals: First, we used `airmon-ng` to put the antennae to the monitor mode. Second, the `iw` command set the antennas to different channels. Then, we used the `tcpdump` command to capture only the beacon frames and the probe requests sent or received from the device to be tested. After three minutes we stopped the recording and merge all .pcap files into one file for the parameter calculation. We write all the commands into one bash file to run them with one command line (please see the appendix B for the shell script).

4.3.2 Data Collection

In the measurement, we recorded only the broadcast beacons from the APs or the frames sent or received from the device we tested. However, in some devices (ex. MacBook) we noticed that there is MAC randomization so we couldn't capture the probes from the global MAC address. In this way we first monitored the device using the Wireshark software, found out the randomized MAC by filtering out the probes while turning the Wi-Fi of the device on and off, and start recording using the temporary MAC address. Normally the randomized MAC is constant throughout our consecutive recordings of a few minutes in each experiment.

The experiment location throughout all test scenarios is fixed, so that we can compare different devices from the same basis.

For collecting up to 10 SSIDs outside the testing environment, we later setup the AP of an assigned channel using the Linux built-in network connection tool. To allow the devices to automatically connect to the setup AP, we need to have an automatic network connection of the host PC first.

Note that it is possible that the antennas didn't capture all the probes received by the device in test, so we need to check whether the categorization of MinCT and MaxCT is correct, because the time difference of two probes if no probe response is recorded (where in fact there was one) will be categorized as a MinCT value rather than the MaxCT value.

When we do the recording, the device in test is within 1 meter, for the purpose of catching sufficient signal power.

In the environment that a device has not yet connected to an AP, we stimulate the device for sending more probe requests by manually turning the Wi-Fi of the device on and off before the connected is established.

The location of the experiment is in Europe, where the 2.4 GHz channel band is from channel 1 to 13, and the 5 GHz band is channel 36,40,44,48 for indoor band, channel 52,56,60,64 for indoors/DFS/TPC band. There are also other available channels in DFS/TPC and SRD(25mW) bands. They are channel 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140 for DFS/TPC, and channel 149, 151, 153, 155, 157, 159, 161, 165 for SRD (25mW) bands[3].

4.4 Post Processing and Parameter Calculation

After the measurement, we manually checked the recorded pcap file and post-process the ones who don't have too many missing packets. The pcap file created from the `tcpdump` command may contain redundant probes. They are the same probe request frame captured by different antennas. We remove the redundant probes by comparing the time difference, the channel the probe request frame is in, and the Sequence Number (SN). If the channel and the SN are identical in the two probe requests, and the time difference is smaller than 0.5 ms, we consider them as the redundant probes and remove the second one.

We listed each burst and calculated the interprobe delay, MinCT, and MaxC. We discarded the parameters calculated using the missing probe requests. This helps us to reduce the error and eliminate the standard variation of the calculated values.

The python code for the post processing is in the appendix C.

Chapter 5

Results

5.1 Scenario 1: Default mode

Table 5.1 gives an overview of the scanning from different devices in the default mode.

It showed that most devices start from channel 1 and scan until the last channel (in Europe it is channel 13). Only the Linux 14.04 scans until channel 11, and skips channel 12 and 13. It could be due to an error. The number of probe requests sent in each channel varies from brand to brand. It is either 1 or 2. With 2 probe requests in a channel, we can compute the delay that is the time difference of this probe to the next in the same channel. Note that this delay is different from the `ProbeDelay` mentioned in the protocol, which means the time before a mobile device starts to scan in a new channel.

Note that for the devices that send two probe requests in a channel, the delay value is about half of the MinCT and MaxCT. This means that these devices separate the time in each channel to the time before sending the first probe request (delay), and the time after sending the first probe request (MinCT or MaxCT). Table 5.1 shows the situation where the interprobe delay is about half of the MinCT (or MaxCT) value. It means that the probe requests are evenly spaced within the time in the channel:

One interesting issue is that in most devices the MinCT and MaxCT are about

Table 5.1: Characterization of different devices in the default mode

Devices	Version	Scan Sequence	1st PRF	PRF /channel	Interprobe Delay (ms)	MinCT (ms)	MaxCT (ms)
Linux	14.04	1,2,3...11	1	2	18.6 ± 1.2	40.5 ± 1.5	39.9 ± 1
Linux	16.04	1,2,3...13	1	1	n	24.2 ± 0.6	24.3 ± 0.6
Windows	10	1,2,3...13	1	1	n	24.2 ± 0.4	105.6 ± 0.4
MacBook	10.12.4	1,2,3...13	1	2	21.2 ± 0.4	47.4 ± 2.8	53.1 ± 7.5
iPhone	10.2.1	1,2,3...13	1	1	n	43.6 ± 0.5	43.6 ± 0.5
Samsung	4.1.2	1,2,3...13	1	2	41.6 ± 0.7	84.8 ± 1.0	84.9 ± 1.1

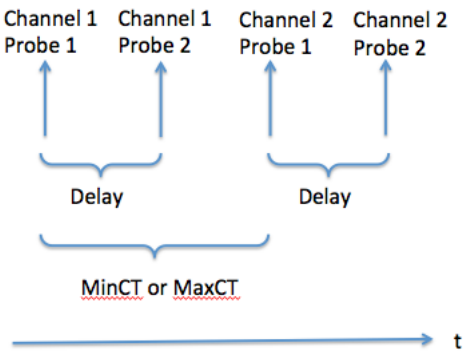


Figure 5.1: Illustration of a mobile device that sends two probe requests in a channel

the same. That means that these devices don't wait longer even if they receive the probe response. We checked the source code of the Android phone and confirmed that the wpa_supplicant which determines the scanning behavior set only one parameter for MinCT and MaxCT. It is called Channel_Time, and the value is 33 ms [1].

Figure 5.2 is the box plot of MinCT and MaxCT values from different devices. Note that in Figure 5.2, the MinCT and MaxCT values vary a lot for Windows. It seems that among the devices tested, Windows is the only brand that has different MinCT and MaxCT values. It has the highest MaxCT value (around 0.1 seconds).

For MacBook the MinCT and MaxCT values jumped between a few values. Figure 5.3 shows that there are values around 47 ms and 62 ms, thus changing the average values in table 5.1. The higher value 62 ms appeared regardless of a probe response in the same probing channel. Note that there can be a mismatch issue. The probe responses received by a device may differ from the probe responses captured by the measurement PC. This is because the directional antennas from the measurement PC (Figure 4.1) are more sensitive than the built-in wireless Internet chipset from the tested device. Therefore it is possible that the measurement PC captured the probe responses which the device failed to capture. In this case the MaxCT may be in fact the MinCT if the device indeed received no probe responses. The experiment has been repeated a few times and we confirmed that there are indeed two values for MinCT and MaxCT. A possible explanation for this phenomenon is that for MacBook, the MinCT and MaxCT values are not fixed values. The channel time can be triggered by another event and resulted in a longer or shorter channel time. Due to time constraint we didn't analyze this in detail but will leave it for the future investigation.

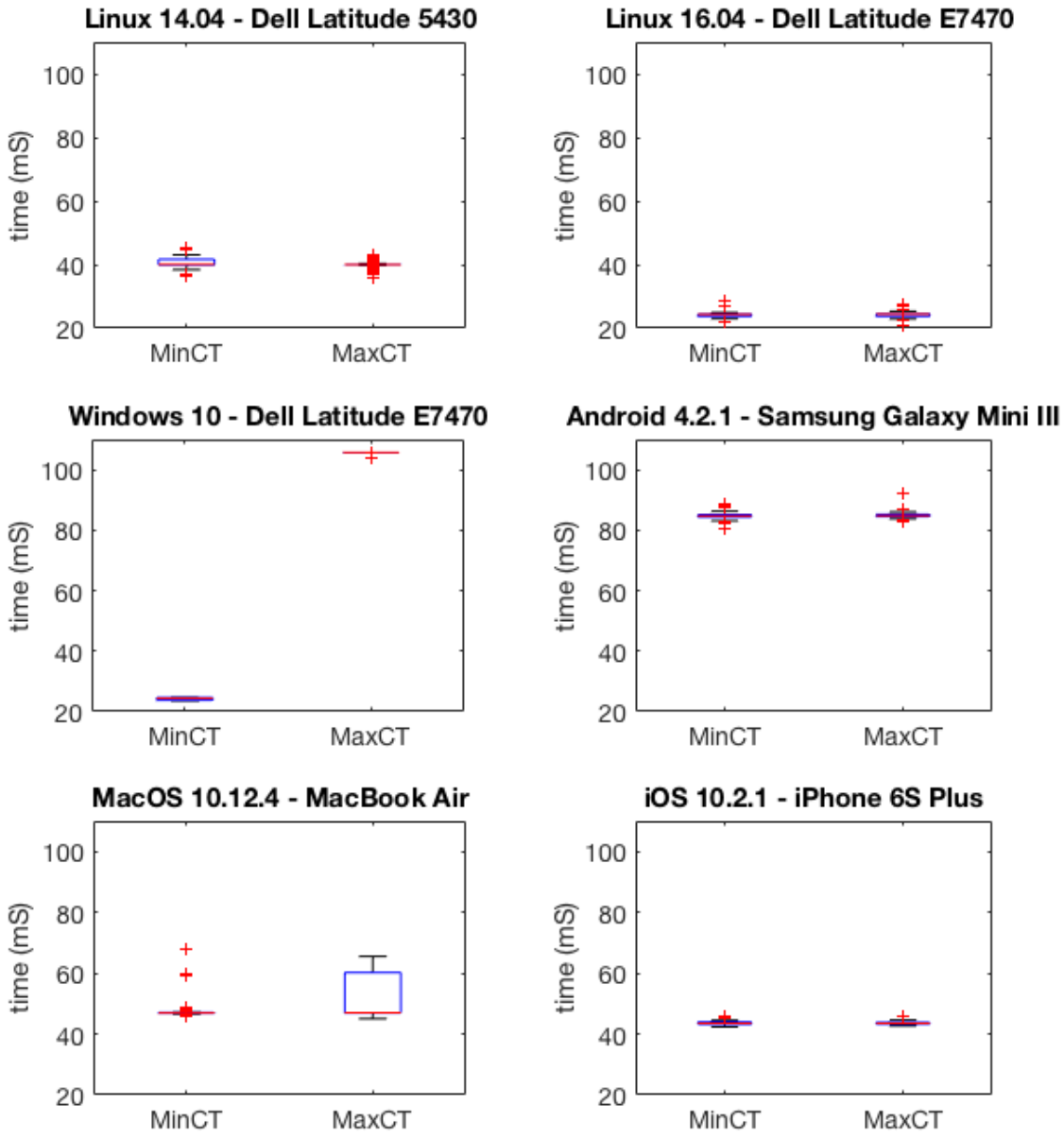


Figure 5.2: MinCT and MaxCT values of different devices in the default mode

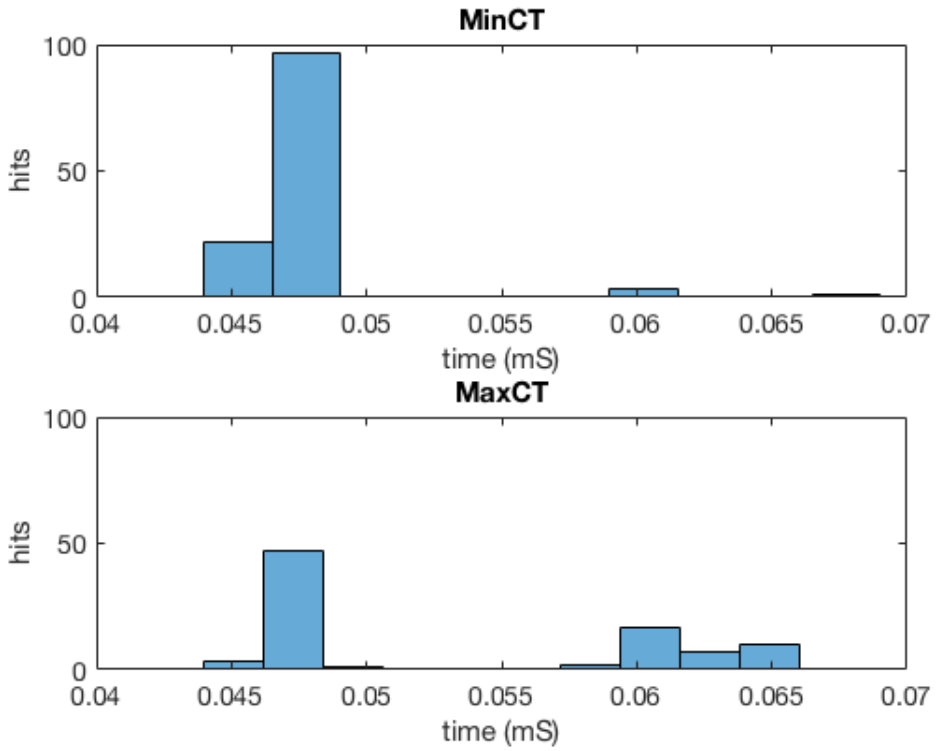


Figure 5.3: MacBook: Histogram of the channel time in the default mode

5.2 Scenario 2: Different Configurations of the Samsung mobile

The experiment result showed the device always did the normal broadcast scanning (the scanning in the default mode) for every 10 seconds, as long as it is on the Wi-Fi setting page. It is regardless of whether the mobile is charging or not, or has been connected to an AP or not. The mobile does the normal broadcast scanning at it default frequency even when it is low in battery (8 percent). The charging mode does not speed the scanning frequency either.

Table 5.2 tested the scan at different functions or pages when not charging. The result showed that turning on the browser which requires the Internet connection did not trigger the mobile to do another scan, nor did the main page play a role. The configuration that triggered the scanning was when the mobile changed from the screen off (sleep) mode to the screen saver mode (by pressing the main page button) while it had not been connected to an AP yet. The mobile did one normal scanning in this situation, but it did not scan when it had already been connected.

Table 5.2: Different test mode when Samsung is not charging

Which page or function?	AP Connected?	Scan
main page (home)	no	no
main page	yes	no
screen saver on from sleep	no	**yes, one default scanning**
screen saver on from sleep	yes	no
browser	no	no
unlock	yes	no
sleep	yes	no

5.3 Scenario 3: Connected, known in proximity (not connected), or not connected

In this test scenario we report the different scanning behaviors when the devices were connected, not yet connected, or not connected to the saved networks. The number of the connected networks also affected the scanning. We discuss the devices one by one.

5.3.1 MacBook

Connected

We observed two different kinds of scanning behaviors when the MacBook connects to the saved network.

First, we observed that MacBook is sending the broadcast probes as well as the probes with the SSID of the saved networks. Figure 5.4 is the snapshot from Wireshark which shows that the saved SSID has been sent over the air. In this test, we have 4 saved networks: 3 are around, 1 is not. The device is connected to one of the saved networks.

The result showed that the channel order of the scanning is the same as in the default scan, but in every channel there are 6 probe requests sent (2 broadcast, 2 to one saved network, 2 to another saved network), where the default scan sends only 2 broadcast probes per channel.

This seems like a serious privacy leakage issue that the MacBook is sending the saved SSID list, so we repeated the experiment again.

This time we have the normal scan as in the default mode, and the scan that contains the SSID of the connected network. The other four saved networks are not sent over the air.

To explain such different outcomes, we speculate that there is a timeout mechanism in MacBook that transmits the probe requests containing only the saved networks established a few minutes ago (which was the case in the first experiment). Due to time constraint we did not experiment further to prove such ideas.

When connected, MacBook uses either the randomized MAC or the global MAC address.

Time	Source	Destination	Length	Current Channel	SSID	Info
19:52:31.895874	Apple_	Broadcast	128	11	test1	Probe Request, SN=3341, FN=0, Flags=........C, SSID=test1
19:52:31.896835	Apple_	Broadcast	128	11	test2	Probe Request, SN=3342, FN=0, Flags=........C, SSID=test2
19:52:31.897766	Apple_	Broadcast	123	11		Probe Request, SN=3343, FN=0, Flags=........C, SSID=Broadcast
19:52:31.907064	Apple_	Broadcast	128	11	test1	Probe Request, SN=3344, FN=0, Flags=........C, SSID=test1
19:52:31.908087	Apple_	Broadcast	128	11	test2	Probe Request, SN=3345, FN=0, Flags=........C, SSID=test2
19:52:31.909017	Apple_	Broadcast	123	11		Probe Request, SN=3346, FN=0, Flags=........C, SSID=Broadcast

Figure 5.4: MacBook broadcasts the SSID from the saved network when connected

Known in proximity (not connected)

In this experiment we have 4 saved networks, and one of them is available for connection. The available AP is in channel 6. MacBook begins by sending two probes in channel 6 containing the SSID of the available AP, and then does the default scanning. Apparently MacBook received the beacon frame from the previously connected network before scanning. Thus it changed the scanning behavior according to the environment. The MAC address of MacBook is the global MAC when know in proximity.

Not connected

When not connected, MacBook prioritizes the channels according to the saved networks. The scanning starts from the channels of the previously saved networks, and then the rest channels. For example when the saved networks are in channel 6 and 11, MacBook sends the probes in channel 6 and 11 first, and then channel 1, 2, 3, 4, 5, 7, 8, 9, 10, 12, 13. We call this behavior **priority scanning**, and investigate this in more details in the scenario 4.

When not connected, MacBook uses the randomized MAC address.

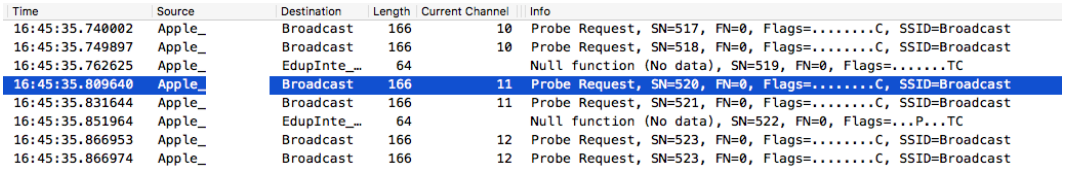
5.3.2 iPhone

Connected

We tested with three saved networks, and connect to one AP in channel 11. iPhone sends two probes in every channel, where the default mode sends only one probe request per channel. Table 5.3 lists the different interprobe delay, MinCT, and MaxCT values with different connection status. The interprobe delay doubles in the connected channel compared to the other channels. Compared to the scan when unconnected, the MinCT and MaxCT values decrease from 43 ms to 23 ms for the connected status, whereas the values in the connected channel increase from 43 ms to 69 ms. A possible explanation to the change can be that when connected, the scanning is in lower priority compared to other tasks, so the MinCT and MaxCT in the unconnected channels are reduced, in order to reduce the total scanning time. As for the increase of MinCT and MaxCT when in the connected channel, the Figure 5.5 (the snapshot of the probe requests in the connected channel (channel 11)) shows that the mobile device sends the null function packet to the connected AP between the probe requests. Such extra tasks increase the MinCT and MaxCT values in the connected channel.

Table 5.3: Scanning characterization table of iPhone 6S plus

status	Scan Sequence	1st PRF	PRF/ channel	Interprobe Delay (mS)	MinCT (mS)	MaxCT(mS)
not connected	1,2...13	1	1	-	43.6 ± 0.5	43.6 ± 0.5
connected	1,2...13 or connected channel and other channels	1 or connected channel	2	10.2 ± 0.5 (in connected channel 21.9 ± 1.1)	23.6 ± 0.7 (in connected channel 69.1 ± 1.1)	23.1 ± 0.4 (in connected channel 68.6 ± 0.5)



Time	Source	Destination	Length	Current Channel	Info
16:45:35.740002	Apple_	Broadcast	166	10	Probe Request, SN=517, FN=0, Flags=........C, SSID=Broadcast
16:45:35.749897	Apple_	Broadcast	166	10	Probe Request, SN=518, FN=0, Flags=........C, SSID=Broadcast
16:45:35.762625	Apple_	EdupInte_…	64		Null function (No data), SN=519, FN=0, Flags=.......TC
16:45:35.809640	Apple_	Broadcast	166	11	Probe Request, SN=520, FN=0, Flags=........C, SSID=Broadcast
16:45:35.831644	Apple_	Broadcast	166	11	Probe Request, SN=521, FN=0, Flags=........C, SSID=Broadcast
16:45:35.851964	Apple_	EdupInte_…	64		Null function (No data), SN=522, FN=0, Flags=...P...TC
16:45:35.866953	Apple_	Broadcast	166	12	Probe Request, SN=523, FN=0, Flags=........C, SSID=Broadcast
16:45:35.866974	Apple_	Broadcast	166	12	Probe Request, SN=523, FN=0, Flags=........C, SSID=Broadcast

Figure 5.5: Wireshark snapshot of the probe requests from iPhone, when connected to an AP in channel 11

When iPhone is connected to an AP, it will do the broadcast scan from channel 1 to 13, and also the scan containing the SSID of the connected network, starting from the connected channel first, and then other channels. For example when the connected channel is 11, it will send 4 probe requests containing the SSID of the saved network in channel 11 first, and then 2 probes per channel in channel 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13.

When connected, iPhone scans using the global MAC address.

Known in proximity (not connected)

When there is an available AP which iPhone connected before, the device will send the whole burst of probe requests containing the SSID of the available AP. The scanning is also 2 probe requests per channel, starting from the channel of the available AP, and then roll over the channels. For example when the available AP is in channel 6, iPhone starts by sending 2 probe requests in channel 6, and then 7, 8, 9, 10, 11, and then 1, 2, 3, 4, 5 all containing the SSID of the available network. The broadcast probe requests are also sent starting from channel 1 first.

When there is known AP in proximity, iPhone scans using the real MAC address.

Not connected

When not connected to any saved networks, iPhone sometimes does the broadcast scanning from channel 1 to 13, or changes order as that in the known-in-proximity case. The reason for triggering such change of order is unknown, because the user cannot see the whole list of the saved networks from the Wi-Fi setting page. Unlike

that in MacBook, it is not possible to manipulate all the channels of the saved network unless reinstalling iPhone. More details of this mode is left as future study.

When not connected, iPhone used the randomized MAC address.

5.3.3 Windows

Windows scans as that in the default mode, whether it is connected to the saved networks or not. The only difference is that when there is a saved network around, it will send an extra probe request containing the SSID of the AP in the channel where AP stays.

Another interesting finding is that Windows seems to lower the data rate speed when it connects to an AP with lower data rate. Original it is 6 Mb/s, as mentioned in the setup part of the data collection chapter, now it becomes 1 Mb/s, so we could measure probes requests from channel 1 to 13 altogether using 5 antennas. This helps us to identify the channel of the probe request more easily.

5.3.4 Linux

Linux also scans as that in the default mode, regardless of connecting to a network or not. When it is connected to an AP, it will send the probe requests containing the SSID to the AP in the channel where the AP stays.

We didn't manage to measure the case of known-in-proximity to the saved networks, because the Wireless Internet card has some problems connecting to APs. After trying all means we still couldn't establish the saved networks inside the Linux PC. As it does the normal scanning when connected, we guess it will probably also do the normal scanning in this case. This is left to be confirmed.

5.3.5 Samsung

The Samsung mobile also does the normal broadcast scanning when connected or not connected to the saved networks. For the known-in-proximity case, it sends the probe request containing the SSID of the saved network, and then does the normal broadcast scanning.

Table 5.4 summarizes the result for scenario 3.

5.4 Scenario 4: The effect of saved networks when not connected

In this section we focus on MacBook as this device has been found doing the priority scanning when there are saved networks. We also checked the scanning behavior of other devices as well.

Table 5.5 lists the change of the channel sequence according to the channels of the saved network in MacBook. Here we see that the device always scans from the channel where the most saved networks are.

Table 5.4: Scenario 3: Characterization of the different connection status when there are saved networks in the devices.

Devices	Not connected	Connected	Known in proximity
MacBook (10.12.4)	PBS	NBS + BSASS(rarely)	PRCCCS + NBS
Windows 10	NBS	NBS	PRCCCS + NBS
iPhone (10.2.1)	NBS (2 PRF/chan.) + BSICS	BSCCS (CC+C1) + NBS (2 PRF/chan.)	BSCCS (CC+Roll) + NBS (2 PRF/chan.)
Samsung (4.1.2)	NBS	NBS	PRCCCS + NBS
linux (16.04)	NBS	NBS + PRCA	-

Abbreviation: **PBS**: priority broadcast scanning; **NBS**: normal broadcast scanning; **BSICS**: broadcast scanning with irregular channel sequence; **BSASS**: broadcast scanning containing all saved SSIDs; **BSCCS (CC+C1)**: broadcast scanning containing the connected SSID (connected channel + other channels from channel 1); **BSCCS (CC+Roll)**: broadcast scanning containing the connected SSID (channel sequence rolling from the connected channel); **PRCCCS**: probe request in the connected channel with the connected SSID; **PRCA**: probe request to the connected AP

Table 5.5: MacBook: Priority Scanning

No. of SSID	Channels of the saved network	First scanned channels	Other scanned channels
10	6 in C6, 2 in C11, 1 in C10 and C1	6,11,10,1	2,3,4,5,7,8,9,12,13
5	2 in C11, 1 in C10, C6, and C1	11,6,10,1	2,3,4,5,7,8,9,12,13
2	C10 and C6	6,10	1,2,3,4,5,7,8,9,11,12,13
1	C10	10	1,2,3,4,5,6,7,8,9,11,12,13
0	N	N	normal scan

For Windows and Linux we added the saved networks with known channels and tested the scanning behavior. They do the normal scanning as in the default mode. For Samsung the saved networks doesn't affect the channel order in the scan, either. As mentioned earlier, it is difficult to remove all the saved networks in iPhone, therefore we didn't test if the device priorities the scanned channels.

5.5 Scenario 5: AP and 5 GHz band

In this section we discuss whether APs affect the scanning of the mobile device. From previous experiment in scenario 3, we know that some devices behave differently when they receive the beacon frames from the saved network (known-in-proximity).

5.5.1 Non-AP environment

In order to simulate a setting where there are no APs accessible by the scanning device, we made a Faraday Cage. Inside the cage we put the measurement PC and antennas along with the devices to test, and shielded with 3 layers of the aluminum foil, including the one attached on the surface of the box. The AP signals and other sources outside the box are successfully blocked, so the measurement tool captured only the signal from the device. We did the experiment with only the mobiles iPhone and Samsung, due to limited space of the box. There was no saved networks in the

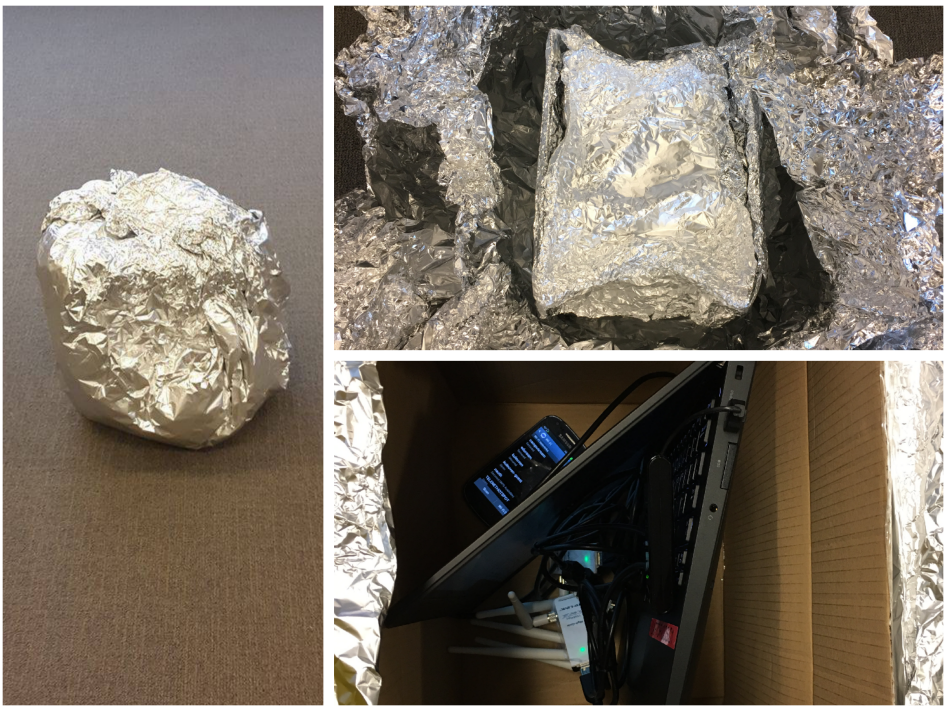


Figure 5.6: The Faraday Cage that creates the non-AP environment

Samsung mobile. Figure 5.6 is the setup of the Faraday cage. The result showed that iPhone and Samsung scan as in the default mode.

5.5.2 5 GHz band

From the previous scenarios, we observed that the 5 GHz bands are also scanned following the 2.4 GHz band. Among all the devices we tested, they scan the 5 GHz channels after the 2.4 GHz channels in the same burst. Unlike the 2.4 GHz that the adjacent channels can be picked up by one antenna, we can only capture the probe request from one channel at a time. Table 5.6 is the preliminary result of the 5 GHz channels scanned by the devices. These measurements are done at the same location, and the there are beacon frames sent in channel 36, 44, 52. We observed that only the first few channels in 5 GHz are scanned by these devices, and the devices scan with the channel spacing of 20 MHz or 40 MHz. Channel 52 is the DFS channel, where the usage is monitored for other radar systems. Therefore we speculate that scanning in this channel may be due to at least one AP sending the beacons in this channel. Or that the mobile device measures the channel usage itself and confirmed that there is no other radar using the channel. To verify such idea, we need to setup an AP in other DFS channels and test the devices again. This is left for future investigation.

Table 5.6: Preliminary result of the 5 GHz scanning from different devices

Devices	Scan	Scanned channels	Channel Spacing (MHz)
Samsung	consecutively after the 2.4 GHz band	36,40,44,48	20
Windows	consecutively after the 2.4 GHz band	36,44,52	40
Linux	consecutively after the 2.4 GHz band	36,44,52	40
Macbook	consecutively after the 2.4 GHz band	36,40,44,48,52	20

Table 5.7: MAC randomization of the devices

Devices	MAC randomization	Note
MacBook	yes	Can still identify the device. Global MAC is used when connected or known in proximity.
Windows	yes	Can be manually switched off.
Linux	no	
Samsung	no	
iPhone	yes	Can still identify the device. Global MAC is used when connected or known in proximity.

5.6 Other findings

5.6.1 MAC randomization

For MacBook and iPhone, the MAC address is randomized, while in Windows the MAC randomization can be manually turned off. Table 5.7 gives the MAC randomization of the devices we tested.

We observed that iPhone sends the global MAC when connected, or when known in proximity. The randomized MAC is only used when it is not connected to any AP. Considering that most users use iPhone when connected, this is surely something for the manufacturer to improve. For MacBook, the randomized MAC is used when unconnected. When connected, we observed it uses either the global or randomized MAC. For the known in proximity it also uses the real MAC. We do not know when the MacBook switches between the global and the randomized MAC. Note that iPhone and Macbook are recorded when they are in the Wi-Fi setting page.

5.6.2 Same OS, different hardware

We tested the Linux OS in different PCs to show whether the firmware or the hardware determines the scanning behavior. The result showed that the scanning sequence of Linux 16.04 is the same, no matter on the old PC (Dell Latitude 5430) or the new PC (Dell Latitude E7470). However, the MinCT and MaxCT values are a lot larger in the old PC than in the new PC. This shows that the hardware plays a role in the channel time too. Please refer to Table 5.8 and the MinCT and MaxCT graph 5.7.

Table 5.8: The characterization result of different Linux OS on different PCs

Devices	Scan Sequence	1st PRF	PRF/ channel	Delay (ms)	MinCT (ms)	MaxCT (ms)
Linux 14.04 in old pc 1	1,2...11	1	2	18.6 ± 1.2	40.5 ± 1.5	39.9 ± 1
Linux 16.04 in new PC	1,2...13	1	1	n	24.2 ± 0.6	24.3 ± 0.6
Linux 16.04 in old pc 2	1,2...11	1	1	n	92.6 ± 2.3	91.9 ± 1.3

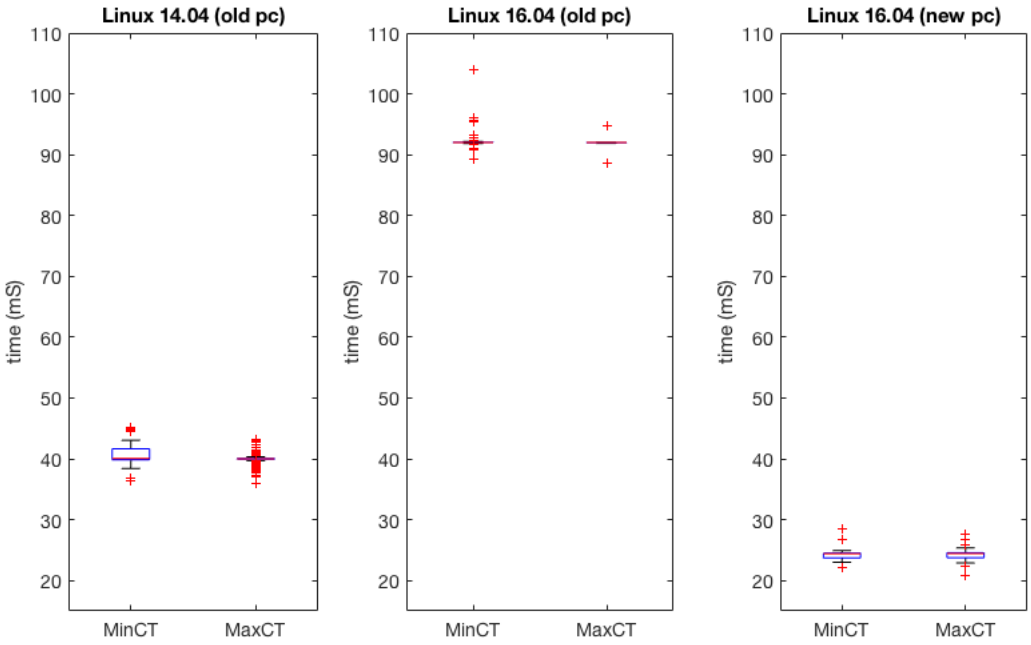


Figure 5.7: MinCT and MaxCT of different Linux versions on different PCs

5.7 Conclusion

The scanning of the five devices has been characterized depending on different connection status, saved networks, and configurations. When the devices are unconnected to any network, and there is no saved network, they scan from channel 1 to the last channel. One or two probe requests per channel are sent, and the `MinChannelTime` and `MaxChannelTime` vary. For the Samsung mobile, the Wi-Fi setting page always triggers the device to scan frequently, and a specific configuration triggers the scanning too. When there are saved networks in the devices, the MacBook does the priority scanning which switches the channel order according to the channels in the saved network. MacBook sends the probe request containing the SSIDs of the saved network, but this case seems to happen to the newly-established saved networks. When the saved network is around, all the devices send the probe request containing the SSID of the available saved network. Finally we have some preliminary result on the 5 GHz band, observing that the devices scan a few 5 GHz channels after scanning the 2.4 GHz channels. We also observed that besides the firmware, the hardware also plays a role in the MinCT and MaxCT values.

Among all devices, iPhone has the most irregular channel sequence order. When not connected and there are saved networks, it switches the channel order, but unlike MacBook, it does not scan the channels of the saved networks first, but some channels based on other mechanism. Such behavior seems to support the information from the Skycure webiste [2] that iPhone has the built-in SSID list. Since the saved network list is not available in the Wi-Fi setting page of iPhone, it is possible that iPhone may be doing the priority scanning. iPhone prioritizes the channels based on the channels of the manually added saved networks, along with the built-in networks. Also, iPhone is found sending double probe requests in each channel after being connected. The channel time in the connected channel also increased. These variations makes identifying probe requests from iPhones easier.

Chapter 6

Conclusion and Future Work

The thesis characterizes the Wi-Fi network discovery by measuring the probe requests sent in the active scanning. We measured the MinCT and MaxCT. Their values range from 20 ms to 105 ms, and only Windows set different values for MinCT and MaxCT. One or two probe requests per channel are sent among the five devices tested. In the default mode, the scanning sequence is from channel 1 to channel 13, and then to a few channels in 5 GHz. These parameters may change under different connection status or saved networks. For example, iPhone doubles the number of probe requests when connected, and MacBook prioritizes the channel sequence when there are saved networks.

Throughout the measurement we found that MAC randomization is implemented in MacBook, iPhone, and Windows. In Linux and Samsung mobile we didn't observe MAC randomization. Although the Samsung mobile has an older OS version (4.1.2), the result agreed with the finding in Martin et al.'s study [23] which observed none of the Samsung mobile using MAC randomization. To our surprise almost none of the devices we tested sent the history SSIDs. In this case it is not possible to refer to the social relationship of two persons as mentioned in Cunche et al.'s work [17]. Only MacBook is found sending the history SSIDs from the networks set up a few minutes ago, not sending the SSID from the networks set up months ago. We found that when connected or there is known AP in proximity, iPhone and MacBook send the probe requests using the real MAC addresses. We also found that MacBook uses the priority scanning mechanism when unconnected. A device doing the priority scanning based on the saved networks makes the probe requests sequence more unique and thus easier to be identified.

To conclude, among the devices we tested, they scan differently. Therefore it is possible to identify the probe requests from different devices. It can be combined with the MAC addresses and the IE mentioned by Vanhoef et al. [28] to help fingerprint the devices. Second, the result showed that it is possible to monitor a person using the probe request. How a device scans may reveal how a person uses the device. For example, the probe requests characteristics change when the user picks up the device which wakes up from the sleep mode. Note that the monitoring only makes sense when the user is not observable. For example when a person is in a building, we

can speculate what he or she is doing by checking the probe requests of the devices being used. This implies, if a person lives in a house full of Internet of Things linked together by Wi-Fi, it is possible for another person from the other side of the wall to know a lot more about him or her through the probe requests characteristics.

We suggest improvement to the probe request setting:

- Remove the Sequence Number in every probe request. Also use different randomized MAC addresses in every probe request to avoid being categorized easily.
- Set different values for `MinChannelTime` and `MaxChannelTime` to help speed up the scanning time. Now all devices expect the Windows PC set the MinCT and MaxCT values identical. The MinCT values can be reduced.
- (For MacBook and iPhone) Avoid using the global MAC addresses to scan when connected or known in proximity.

There can be several aspects to expand the current work. We need to upgrade the measurement setup to catch all the probe requests sent in 5 GHz band. This can help characterize the complete probe requests in a scan. Also we need to configure an AP to send the beacons in different DFS channels to check if it changes the channel scan sequence of a device. We also need to measure the frequency of a scan in different configurations. For example how frequent does a device scan when in sleep mode? This helps us to understand how many probe requests are sent and can help identify the devices easier. We also need to test more devices, especially new versions of the Android phones.

Appendices

Characterizing Wi-Fi Network Discovery

Yun-Chih Joy Chou*, Gunes Acar*, Rafael Galvez*, and Claudia Diaz*
* COSIC, ESAT, KU Leuven
Kasteelpark Arenberg 10 bus 2452,
3001 Heverlee, Belgium

***Abstract*—In the Wi-Fi network discovery phase, the probe request sent by a mobile device is not encrypted. Therefore, the MAC address contained in the probe request frame can be exploited to track users without their knowledge and consent. In order to help understand how much the probe requests reveal the behavior or traces of a user, this paper characterizes the scanning behavior of the mobile device during network discovery. We test five different devices and find that all of them send one or two probe requests per channel. All devices except the Windows PC set the same value for the minimum and maximum channel time, which ranged from 20 ms to 105 ms. Our experiments show that MacBook prioritizes the channels of the saved networks, which reduces the connection time. Our results show that devices exhibit different scanning behavior in different settings. We found that three of the five devices tested used randomized MAC addresses, and almost none of them sent the SSIDs of the previously connected networks. We also observed that, some devices with MAC randomization reveal their true MAC addresses once connected to a network.**

I. Introduction

Nowadays many people use Wi-Fi to connect to the Internet. It is one of the most important standards deployed in the mobile devices as well as laptops. In the network discovery phase, the device scans by sending the probe request to ask the APs to reply with the probe response. The probe request is not encrypted, therefore anyone who receives the probe request frame can interpret it. This is where the problem comes from.

In a probe request, the MAC address and the Service Set IDentifier (SSID) may reveal the private information. Because of this, the randomized MAC address has been introduced and implemented, but most of the devices can still be identified by combining other information in the probe request (Martin et al. [1]). Therefore it is important to reduce the number of probe requests in every scan. Also it is important to see which devices are sending the SSIDs of the previously connected networks to suggest any improvement.

The contributions of the paper are:

- The characteristics of probe requests from different devices under different scenarios are different, so they can be used to identify the devices. The probe request sequence usually start from channel 1 and move to the next channel after 20-100 ms. One exception is MacBook. It does the priority scanning starting from different channels according to the channels of the saved networks.
- Among the devices we tested, almost all devices don't send the SSIDs of the saved networks. Only MacBook seems to send the SSIDs of the newly-established networks within an amount of time.
- Some devices use the randomized MAC addresses when not connected to the network. We found that MacBook and iPhone scan using the global MAC address when they are connected (in the Wi-Fi setting page), or when they are about to connect to the known network.

II. Background

A. Wi-Fi and Network Discovery

The Wireless Fidelity (Wi-Fi) is the most common type of the Wireless Local Area Network (WLAN). It is described by the IEEE 802.11 Protocol [2] that describes the specification in the MAC and physical (PHY) layers of what every device needs to comply when communicating with each other using the WLAN. The most common frequencies used in the Wi-Fi are 2.4 GHz and 5 GHz. There are different types of the networks. Here we focus on the infrastructure network. In the infrastructure network, there is an AP that every device connects to the AP for Wi-Fi. The AP constantly sends the beacon frame signals for the mobile device to synchronize with it, also allowing the external mobile devices to discover it. When a mobile device wants to connect to any external Wi-Fi network, it starts the interworking and launches the network discovery process.

B. Active and Passive scanning

There are two types of network discovery processes: Active and passive scanning. In the active scanning, a mobile device can choose to send a broadcast **probe request** to search for available APs in a channel. Once receiving the probe request, an AP replies with a probe response. In the passive scanning, the mobile device doesn't send probe requests but passively wait for beacons in a channel. This connection requires the AP to send beacons first, and is therefore considered more time-consuming and thus more power-consuming than the active scanning [3].

C. Scanning Parameters

The primitive parameters are in the scan request of the MAC Layer Management Entity (MLME) of the mobile device named `MLME-SCAN.request`. This primitive requests a survey of potential basic service set (BSS, in this case the external networks) that the mobile device can later choose to join. These parameters are stored in the mobile device, and not all parameters are contained in the probe request sent over the air.

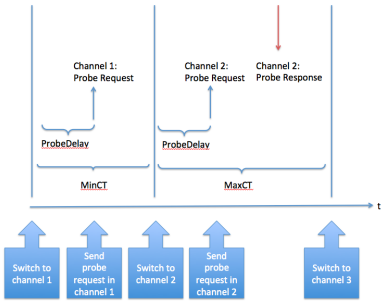


Fig. 1. Overview of the ProbeDelay, MinCT, and MaxCT

1) ProbeDelay, MinChannelTime, MaxChannelTime: The `ProbeDelay` is the time the mobile device waits before sending the probe request ([2] p.266). `MinChannelTime` and `MaxChannelTime` are the minimum and maximum time the mobile device has to spend in a channel ([2] p.266). If the mobile device receives no probe responses within the minimum channel time (MinCT), it will switch to another channel and start scanning in that channel. Otherwise, if there is at least a probe response within MinCT, the mobile device waits up to the maximum channel time (MaxCT) for more probe responses. Figure 1 shows how the mobile device behaves in the active scanning network discovery phase. In reality the dwell time can be larger than the MinCT or MaxCT, because it can also take some time for the mobile device to switch to the desired channel.

2) SSID: The Service Set Identifier (SSID) is the identifier of the Service Set (SS). In an infrastructure network, the SS can be a BSS or Extended SS (ESS). The identifier of the AP is often a readable name. For example there is an SSID of ESS is called `eduroam`. In many mobile devices, a list of SSIDs is stored to allow auto-connection to the previously connected networks.

3) Other parameters: The `VendorSpecificInfo` describes the manufacturer of the mobile device. It is also contained in the probe request sent by the mobile device.

D. Probe Request Frame

Unlike `MLME-SCAN.request`, the probe request frame (PRF) is the data sent over the air. The parameters mentioned in the protocol are optionally present in a probe request frame. In every probe request frame, the MAC address of the mobile device is included. The MAC address is like an identifier of the mobile device. There are universal and locally administered MAC addresses. When the MAC address is universal (or global), there is only one device using the address. When the address is locally assigned, there can be more mobile devices that share the same address. The Sequence Number (SN) is a compulsory parameter added in every transmitted frame of a mobile device. It acts as a counter.

III. Related Work

A. Probe Request Application and Possible Mischievous Use

Cheng et al. [4] used the probe requests for Wi-Fi positioning, with an accuracy of about 15 meter in urban areas. Musa et al. [5] tracked the devices by capturing the Wi-Fi frames sent by the devices that passed by. Tracking using the Wi-Fi probe requests has been exploited for marketing. The Euclid analytics (euclidanalytics.com) is one example. In [6] Cunche et al. described a model to infer the social relationship between two persons by examining the probe requests sent from their mobiles.

B. MAC Randomization Reversal

In [7] Vanhoef et al. found that it is possible to reverse the randomized MAC and fingerprint the device. By combining the Information Elements (IE) in the probe requests and the SN, it is easy to group the packets sent from a device even though randomized MAC addresses are used. In [8] Matte et al. found that the signature based on Inter-Frame Arrival Time (IFAT) from the probe requests can be used to identify the devices, even though the devices have randomized MAC. In [1] Martin et al. did a thorough study to reverse the randomized MAC addresses from different devices that have the global MAC addresses. After detailed testing, they confirmed that there are at least three kinds of methods to retrieve the global randomized MAC addresses. Robyns et al. in [9] gives the latest update about fingerprinting devices using the IE in the probe request frames or other packets. They collected some bits in the probe request frame and used them to identify the devices.

C. Scanning Characterization

In [10] Freudiger et al. measured the number of probe requests from several mobiles. They tested different devices under different scenarios, including the capture configurations, the numbers of the saved networks, and the device configurations. Gupta et al's work [11] is the early study on characterizing the Wi-Fi scanning. They listed the scanning parameters that was not described in the 802.11 protocol such as the channel on which the 1st PRF is sent, or the number of PRFs transmitted per channel.

D. Scanning Optimization

In [12] Wu et al. proposed not to scan if the mobile does not move beyond 10-20 meters. They implemented the cellular assisted movement detection in Windows phones and showed that 90 percent of the power has been saved. In [13] Kim et al. proposed a location-aided mechanism that sends the SSID of the saved network only when the current location is in the proximity of that saved network. In [14] Castignani et al. proposed to set the MinCT and MaxCT adaptively, based on the availability of the APs found in that channel. In [15] Goovearts et al. modified the `wpa_supplicant` source code to scan the non-overlapping channels (1,6, 11) first. The the available networks showed up in the list a lot more quickly.

TABLE I
TESTED DEVICES

Type	Brand	Model	OS
Mobile	Samsung	S3 Mini	Android 4.1.2
Mobile	Apple	6S Plus	iOS 10.2.1
PC	Dell	Latitude E7470	Windows 10
PC	Dell	Latitude E7470	Ubuntu 16.04
PC	Dell	Latitude E5430	Ubuntu 14.04
PC	Apple	MacBook Air	MacOS 10.12.4

IV. DATA COLLECTION

A. Mobile Devices Selection

We chose a few common phones and PCs as our testing mobile devices. Table I lists the OS and model version of each device.

B. The focus and test scenarios

We aim to characterize how each mobile device scans under different scenarios. There are attributes which may affect the scanning of the device. First, the device configuration (active or sleep, charging or not, connected to APs or not etc.). Second, the saved networks (SSID history) in the device. Third, the environment that has known or unknown APs (known in proximity or not).

To characterize the scanning, there are some parameters to be tested (the list is mostly from Gupta et al.'s paper [11]): Channel sequence, channel of the first probe request frame (PRF), number of PRF per channel, interprobe delay in each channel (when there are more than 2 probes per channel), MinCT and MaxCT.

The first test scenario is how the devices scan under the default mode. The default mode for the cellphone is when the screen is on at the Wi-Fi setting page, not charging, not connected to any AP, and there is no saved SSID. For PCs, the Wi-Fi has been turned on and off manually for every few seconds to stimulate more bursts (a set of probe requests scanned continously). The second test scenario is to see whether different configurations of the same device will change the scanning behavior. The default setting is that the device has no saved SSID. The third test scenario is to check the scanning behaviors when the device is connected to the network, known in proximity (not connected), or not connected to the network. The fourth scenario is to check if the numbers of the saved networks affect the scanning when the device is not connected to any network.

C. Measurement

1) General Setup: For 2.4 GHz band, We captured the 13 channels at once by using the 5 antennas tuned to 5 different channels. The reason is that each antenna captures not only the tuned channel, but also the adjacent channels. So while tuning them to channel 2,5,8,11,13, each antenna can capture at least 3 channels. The exception is Windows PC, where the data rate is 6 Mb/s or 12 Mb/s, higher than others (Linux is 1 Mb/s), so it is more difficult for the antenna to pick up the adjacent channels in this setting. Hence, for the case of Windows we record the information in three parts to cover all the channels.

For the measurement in the 2.4 GHz band, our setup consists of five antennas, a Linux PC, and the USB hub to connect all the antennas to the PC. For the measurement of 2.4 GHz with the 5GHz band, we add an extra 5 GHz antenna to capture the 5GHz signal. We used a few Linux command line tools to record the signals: First, we used `airmon-ng` to put the antennae to the monitor mode. Second, the `iw` command set the antennas to different channels. Then, we used the `tcpdump` command to capture only the beacon frames and the probe requests sent or received from the device to be tested. After three minutes we stopped the recording and merge all .pcap files into one file for the parameter calculation.

2) Data Collection: In the measurement, we recorded only the broadcast beacons from the APs or the frames sent or received from the device we tested. However, in some devices (ex. MacBook) we noticed that there is MAC randomization so we couldn't capture the probes from the global MAC address. In this way we first monitored the device using the Wireshark software, found out the randomized MAC by filtering out the probes while turning the Wi-Fi of the device on and off, and start recording using the temporary MAC address. Normally the randomized MAC is constant throughout our consecutive recordings of a few minutes in each experiment.The experiment location throughout all test scenarios is fixed, so that we can compare different devices from the same basis.

For collecting up to 10 SSIDs outside the testing environment, we later setup the AP of an assigned channel using the Linux built-in network connection tool. While recording, the device in test is within 1 meter, for the purpose of catching sufficient signal power. In the environment that a device has not yet connected to an AP, we stimulate the device for sending more probe requests by manually turning the Wi-Fi of the device on and off before the connected is established.

The location of the experiment is in Europe, where the 2.4 GHz channel band is from channel 1 to 13, and the 5 GHz band is channel 36,40,44,48 for indoor band, channel 52,56,60,64 for indoors/DFS/TPC band. There are also other available channels in DFS/TPC and SRD(25mW) bands.

D. Post Processing and Parameter Calculation

After the measurement, we remove the redundant probes by comparing the time difference, the channel the probe request frame is in, and the SN. If the channel and the SN are identical in the two probe requests, and the time difference is smaller than 0.5 ms, we consider them as the redundant probes and remove the second one.

V. RESULTS

A. Scenario 1: Default mode

Table II gives an overview of the scanning from different devices in the default mode.

It showed that most devices start from channel 1 and scan until the last channel (in Europe it is channel 13). Only the Linux 14.04 scans until channel 11, which skipped channel

TABLE II
CHARACTERIZATION OF DIFFERENT DEVICES IN THE DEFAULT MODE

Devices	Version	Scan Sequence	1st PRF	PRF /channel	Interprobe Delay (ms)	MinCT (ms)	MaxCT (ms)
Linux	14.04	1,2,3...11	1	2	18.6 ± 1.2	40.5 ± 1.5	39.9 ± 1
Linux	16.04	1,2,3...13	1	1	n	24.2 ± 0.6	24.3 ± 0.6
Windows	10	1,2,3...13	1	1	n	24.2 ± 0.4	105.6 ± 0.4
MacBook	10.12.4	1,2,3...13	1	2	21.2 ± 0.4	47.4 ± 2.8	53.1 ± 7.5
iPhone	10.2.1	1,2,3...13	1	1	n	43.6 ± 0.5	43.6 ± 0.5
Samsung	4.1.2	1,2,3...13	1	2	41.6 ± 0.7	84.8 ± 1.0	84.9 ± 1.1

12 and 13. It could be due to an error. The number of probe requests sent in each channel varies from brand to brand. It is either 1 or 2. With 2 probe requests in a channel, we can compute the delay that is the time difference of this probe to the next in the same channel. For the devices that send two probe requests in a channel, the delay value is about half of the MinCT and MaxCT. This means that these devices basically separate the time in each channel to the time before sending the first probe request (delay), and the time after sending the first probe request (MinCT or MaxCT). One interesting issue is that in most devices the MinCT and MaxCT are about the same. That means that these devices don't wait longer even if they receive the probe response. We checked the source code of the Android phone and confirmed that the wpa_supplicant which determines the scanning behavior set only one parameter for MinCT and MaxCT. It is called Channel_Time, and the value is 33 ms [16].

Figure 2 is the box plot of MinCT and MaxCT values from different devices. Note that in Figure 2, the MinCT and MaxCT values vary a lot for Windows. It seems that among the devices tested, Windows is the only brand that has different MinCT and MaxCT values. It has the highest MaxCT value (around 0.1 seconds).

B. Scenario 2: Different Configurations of the Samsung mobile

The experiment result showed the device will always do the normal broadcast scanning (the scanning in the default mode) for every 10 seconds, as long as it is on the Wi-Fi setting page. It is regardless of whether the mobile is charging or not, or has been connected to an AP or not. The mobile does the normal broadcast scanning at it default frequency even when it is low in battery (8 percent). The charging mode does not speed the scanning frequency either. Also, the configuration that triggered the scanning was when the mobile changed from the screen off (sleep) mode to the screen saver mode (by pressing the main page button) while it had not been connected to an AP yet. The mobile did one normal scanning in this situation, but it did not scan when it had already been connected.

C. Scenario 3: Connected, known in proximity (not connected), or not connected

1) MacBook: The results for MacBook:

- Connected:

We observed two different kinds of scanning behaviors when the MacBook is connected to the network. MacBook

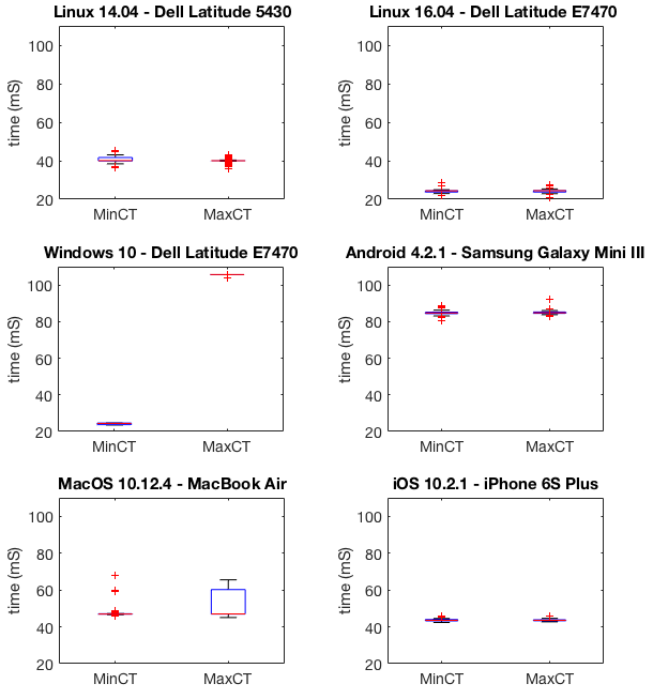


Fig. 2. MinCT and MaxCT values of different devices in the default mode

is either doing the normal broadcast scan as that in the default mode, or the broadcast scan with the SSIDs of the newly-established saved networks. To explain such different outcomes, we speculate that there is a timeout mechanism in MacBook that transmits the probe requests containing only the saved networks established a few minutes ago (which was the case in the first experiment).

- Known in proximity (not connected):

In this experiment we have 4 saved networks, and one of them is available for connection. The available AP is in channel 6. MacBook begins by sending two probes in channel 6 containing the SSID of the available AP, and then does the default scanning. Apparently MacBook received the beacon frame from the previously connected network before scanning. Thus it changed the scanning behavior according to the environment.

- Not Connected:

When not connected, MacBook prioritizes the channels according to the saved networks. The scanning starts from the channels of the previously saved networks, and then the rest channels. For example when the saved networks are in channel 6 and 11, MacBook sends the probes in channel 6 and 11 first, and then channel 1, 2, 3, 4, 5, 7, 8, 9, 10, 12, 13. We call this behavior **priority scanning**, and investigate this in more details in the scenario 4.

2) iPhone: The results for iPhone:

- Connected:

iPhone sends two probes in every channel, where the default mode sends only one probe request per channel. Table III lists the different interprobe delay, MinCT, and MaxCT values

TABLE III
SCANNING CHARACTERIZATION TABLE OF IPHONE 6S PLUS

status	Scan Sequence	1st PRF	PRF/ channel	Interprobe Delay (mS)	MinCT (mS)	MaxCT(mS)
not connected	1,2...13	1	1	-	43.6 ± 0.5	43.6 ± 0.5
connected	1,2...13 or connected channel and other channels	1 or connected channel	2	10.2 ± 0.5 (in connected channel 21.9 ± 1.1)	23.6 ± 0.7 (in connected channel 69.1 ± 1.1)	23.1 ± 0.4 (in connected channel 68.6 ± 0.5)

with different connection status. The interprobe delay doubles in the connected channel compared to the other channels. Compared to the scan when unconnected, the MinCT and MaxCT values decrease from 43 ms to 23 ms for the connected status, whereas the values in the connected channel increase from 43 ms to 69 ms. A possible explanation to the change can be that when connected, the scanning is in lower priority compared to other tasks, so the MinCT and MaxCT in the unconnected channels are reduced, in order to reduce the total scanning time. As for the increase of MinCT and MaxCT in the connected channel, we found that the mobile device sends the null function packet to the connected AP between the probe requests. Such extra tasks increase the MinCT and MaxCT values in the connected channel.

When iPhone is connected to an AP, it will do the broadcast scan from channel 1 to 13, and also the scan containing the SSID of the connected network, starting from the connected channel first, and then other channels.

- Known in proximity (not connected):

When there is an available AP around iPhone, it will send the whole burst of probe requests containing the SSID of the available AP. The scanning is also 2 probe requests per channel, starting from the channel of the available AP, and then roll over the channels. The broadcast probe requests are also sent starting from channel 1 first.

- Not connected:

When not connected to any saved networks, iPhone sometimes does the broadcast scanning from channel 1 to 13, or changes order as that in the known-in-proximity case. The reason for triggering such change of order is unknown, because the user cannot see the whole list of the saved networks from the Wi-Fi setting page.

3) Windows: Windows scans as that in the default mode, whether it is connected to the saved networks or not. The only difference is that when there is a saved network around, it will send an extra probe request containing the SSID of the AP in the channel where AP stays.

4) Linux: Linux also scans as in the default mode, regardless of connecting to a network or not. When it is connected to an AP, it will send the probe requests containing the SSID to the AP in the channel where the AP stays.

5) Samsung: The Samsung mobile also does the normal broadcast scanning when connected or not connected to the saved networks. For the known-in-proximity case, it sends the probe request containing the SSID of the saved network, and then does the normal broadcast scanning.

TABLE IV
SCENARIO 3: CHARACTERIZATION OF THE DIFFERENT CONNECTION STATUS WHEN THERE ARE SAVED NETWORKS IN THE DEVICES.

Devices	Not connected	Connected	Known in proximity
MacBook (10.12.4)	PBS	NBS + BSASS(rarely)	PRCCCS + NBS
Windows 10	NBS	NBS	PRCCCS + NBS
iPhone (10.2.1)	NBS (2 PRF/chan.) + BSICS	BSCCS (CC+C1) + NBS (2 PRF/chan.)	BSCCS (CC+Roll) + NBS (2 PRF/chan.)
Samsung (4.1.2)	NBS	NBS	PRCCCS + NBS
linux (16.04)	NBS	NBS + PRCA	-

Abbreviation: **PBS**: priority broadcast scanning; **NBS**: normal broadcast scanning; **BSICS**: broadcast scanning with irregular channel sequence; **BSASS**: broadcast scanning containing all saved SSIDs; **BSCCS (CC+C1)**: broadcast scanning containing the connected SSID (connected channel + other channels from channel 1); **BSCCS (CC+Roll)**: broadcast scanning containing the connected SSID (channel sequence rolling from the connected channel); **PRCCCS**: probe request in the connected channel with the connected SSID; **PRCA**: probe request to the connected AP

Table IV summarizes the result for scenario 3.

D. Scenario 4: The effect of saved networks when not connected

We found that MacBook always scans the channel where the most saved networks are, then other channels in the saved networks, and then the rest channels. For Windows, Linux, and Samsung we added the saved networks with known channels and tested the scanning behavior. They do the normal scanning as in the default mode. As mentioned earlier, it is difficult to remove all the saved networks in iPhone, therefore we didn't test if the device priorities the scanned channels.

E. Scenario 5: AP and 5 GHz band

1) Non-AP environment: In order to simulate a setting where there are no APs accessible by the scanning device, we made a Faraday Cage. Inside the cage we put the measurement PC and antennas along with the devices to test, and shielded with 3 layers of the aluminum foil, including the one attached on the surface of the box. The AP signals and other sources outside the box are successfully blocked, so the measurement tool captured only the signal from the device. We did the experiment with only the mobiles iPhone and Samsung, due to limited space of the box. The result showed that iPhone and Samsung scan as in the default mode.

2) 5 GHz band: From the previous scenarios, we observed that the 5 GHz bands are also scanned following the 2.4 GHz band. Among all the devices we tested, they scan the 5 GHz channels after the 2.4 GHz channels in the same burst. Unlike the 2.4 GHz that the adjacent channels can be picked up by one antenna, we can only capture the probe request from one channel at a time. These measurements are done at the same location, and the there are beacon frames sent in channel 36,44,52. We observed that only the first few channels in 5 GHz are scanned by these devices, and the devices scan with the channel spacing of 20 MHz or 40 MHz. Channel 52 is the DFS channel, where the usage is monitored for other radar systems. Therefore we speculate that scanning in this channel may be due to that there is at least an AP sending the beacons in this channel. Or that the mobile device measures the channel usage itself and confirmed that there is no other radar using the channel.

TABLE V
THE CHARACTERIZATION RESULT OF DIFFERENT LINUX OS ON DIFFERENT PCS

Devices	Scan Sequence	1st PRF	PRF/ channel	Delay (ms)	MinCT (ms)	MaxCT (ms)
Linux 14.04 in old pc 1	1,2...11	1	2	18.6 $\pm$ 1.2	40.5 $\pm$ 1.5	39.9 $\pm$ 1
Linux 16.04 in new PC	1,2...13	1	1	n	24.2 $\pm$ 0.6	24.3 $\pm$ 0.6
Linux 16.04 in old pc 2	1,2...11	1	1	n	92.6 $\pm$ 2.3	91.9 $\pm$ 1.3

F. Other findings

1) MAC randomization: For MacBook and iPhone, the MAC address is randomized, while in Windows the MAC randomization can be manually turned off. We observed that iPhone sends the global MAC when connected, or when known in proximity. The randomized MAC is only used when it is not connected to any AP. Considering that most users use iPhone when connected, this is surely something for the manufacturer to improve. For MacBook, the randomized MAC is used when unconnected. When connected, we observed it uses either the global or randomized MAC. For the known in proximity it also uses the real MAC. We do not know when the MacBook switches between the global and the randomized MAC. Note that iPhone and Macbook are recorded when they are in the Wi-Fi setting page.

2) Same OS, different hardware: The result showed that the scanning sequence of Linux 16.04 is the same, no matter on the old PC (Dell Latitude 5430) or the new PC(Dell Latitude E7470). However, the MinCT and MaxCT values are a lot larger in the old PC than in the new PC. This shows that the hardware plays a role in the channel time besides the effect of firmware. Please refer to table V.

VI. CONCLUSION

This paper characterizes the Wi-Fi network discovery by measuring the probe requests sent in the active scanning. We measured the MinCT and MaxCT. Their values range from 20 ms to 105 ms, and only Windows set different values for MinCT and MaxCT. One or two probe requests per channel are sent among the five devices tested. In the default mode, the scanning sequence is from channel 1 to channel 13, and then to a few channels in 5GHz. These parameters may change under different connection status or saved networks. For example, iPhone doubles the number of probe requests when connected, and MacBook prioritizes the channel sequence when there are saved networks.

Throughout the measurement we found that MAC randomization is implemented in MacBook, iPhone, and Windows. In Linux and Samsung mobile we didn't observe MAC randomization. Although the Samsung mobile has an older OS version (4.1.2), the result agreed with the finding in Martin et al.'s study [1] which observed none of the Samsung mobile using the MAC randomization. To our surprise almost none of the devices we tested sent the history SSIDs. In this case it is not possible to refer to the social relationship of two persons as mentioned in Cunche et al.'s work [6]. Only MacBook is found sending the history SSIDs from the networks set up a few minutes ago, not sending the SSID from the networks set up months ago. We found that when connected or there is known AP in proximity, iPhone and MacBook send the probe requests using the real MAC addresses. We also found that MacBook uses the priority scanning mechanism when unconnected. A device doing the priority scanning based on the saved networks makes the probe requests sequence more unique and thus easier to be identified.

ACKNOWLEDGMENT

The authors would like to thank Professor Bart Preneel, Professor Sofie Pollin for evaluating this paper. Also thanks to Mathy, Frederik, Edu, and others that gave helpful comments.

REFERENCES

[1] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, and D. Brown, "A Study of MAC Address Randomization in Mobile Devices and When it Fails," *arXiv preprint arXiv:1703.02874*, 2017.

[2] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification," Piscataway, NJ, 2016.

[3] H. Velayos and G. Karlsson, "Techniques to reduce the IEEE 802.11 b handoff time," in *Communications, 2004 IEEE International Conference on*, vol. 7. IEEE, 2004, pp. 3844–3848.

[4] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm, "Accuracy characterization for metropolitan-scale Wi-Fi localization," in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM, 2005, pp. 233–245.

[5] A. B. M. Musa and J. Eriksson, "Tracking unmodified smartphones using wi-fi monitors," in *Proceedings of the 10th ACM conference on embedded network sensor systems*. ACM, 2012, pp. 281–294.

[6] M. Cunche, M.-A. Kaafar, and R. Boreli, "Linking wireless devices using information contained in Wi-Fi probe requests," *Pervasive and Mobile Computing*, vol. 11, pp. 56–69, 2014.

[7] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. 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*. ACM, 2016, pp. 413–424.

[8] C. Matte, M. Cunche, F. Rousseau, and M. Vanhoef, "Defeating MAC Address Randomization Through Timing Attacks," in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2016, pp. 15–20.

[9] P. Robyns, B. Bonné, P. Quax, and W. Lamotte, "Noncooperative 802.11 MAC Layer Fingerprinting and Tracking of Mobile Devices," *Security and Communication Networks*, vol. 2017, 2017.

[10] J. Freudiger, "Short: 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*, ser. WiSec '15. ACM, 2015.

[11] V. Gupta, R. Beyah, and C. Corbett, "A characterization of wireless {NIC} active scanning algorithms," in *Wireless Communications and Networking Conference 2007. WCNC 2007.* IEEE, 2007, pp. 2385–2390.

[12] H. Wu, K. Tan, J. Liu, and Y. Zhang, "Footprint: cellular assisted Wi-Fi AP discovery on mobile phones for energy saving," in *Proceedings of the 4th ACM international workshop on Experimental evaluation and characterization*. ACM, 2009, pp. 67–76.

[13] Y. S. Kim, Y. Tian, L. T. Nguyen, and P. Tague, "LAPWiN: Location-aided probing for protecting user privacy in Wi-Fi networks," in *Communications and Network Security (CNS), 2014 IEEE Conference on*. IEEE, 2014, pp. 427–435.

[14] G. Castignani, A. Arcia, and N. Montavont, "A study of the discovery process in 802.11 networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 15, no. 1, pp. 25–36, 2011.

[15] F. Goovaerts, "Passive Wi-Fi Scanning: a viability study," Master's thesis, KU Leuven, 2017.

[16] "Android open source code: WPA_Supplicant mlme.c in Jelly Bean 4.1.2," http://androidxref.com/4.1.2/xref/external/wpa_supplicant_6/wpa_supplicant/mlme.c#2605.

Appendix B

Shell Script

This is the shell script to record the frames and create a .pcap file:

```
#!/bin/bash
# To Run the code: sudo bash test.sh

TIMEOUT=180

enable_monitor_mode () {
  sudo airmon-ng start $1
}

set_channel () {
        sudo iw dev $1 set channel $2
}

start_capture () {
# only the beacon frames or the probe request sent from the
    assigned MAC is captured
  tcpdump -i $1 wlan[0]==0x80 or wlan host aa:bb:cc:dd:ee:ff
      -w $1.pcap &
}

remove_mon_interfaces () {
  sudo airmon-ng stop mon0
  sudo airmon-ng stop mon1
  sudo airmon-ng stop mon2
  sudo airmon-ng stop mon3
  sudo airmon-ng stop mon4
}
```

```
# Note: IMPORTANT!! Check if error message appears when
    channels are set.

enable_monitor_mode wlan6
enable_monitor_mode wlan2
enable_monitor_mode wlan3
enable_monitor_mode wlan4
enable_monitor_mode wlan5

echo "5 antennas are put to monitor mode..."

# set channels of the 5 antennas to cover all 13 channels
set_channel wlan2 2
set_channel wlan3 5
set_channel wlan4 8
set_channel wlan5 11
set_channel wlan6 13

echo "Antennas are set into 5 different channels
    2,5,8,11,13..."

echo "Capture will begin..."

start_capture mon0
start_capture mon1
start_capture mon2
start_capture mon3
start_capture mon4

echo "All started capturing..."

sleep $TIMEOUT
sudo killall tcpdump

#for i in (seq 0 4)
sudo airmon-ng stop mon0
sudo airmon-ng stop mon1
sudo airmon-ng stop mon2
sudo airmon-ng stop mon3
sudo airmon-ng stop mon4

echo "All stopped capturing..."

# mergin all pcap files into combined.pcap
```

```
mergecap -w combined.pcap mon0.pcap mon1.pcap mon2.pcap mon3
    .pcap mon4.pcap
echo "Files are merged into combined.pcap. Finished."
```

Appendix C

Python Code

This the Python code to calculate the scanning parameters:

```
import dpkt
import struct
import binascii
import traceback
import numpy
from dpkt.dpkt import NeedData, UnpackError

SEQ_NO_INDEX = 22

def mac_addr(adrs):
    return ':'.join('%02x' % ord(b) for b in adrs)

def get_current_chan(wlan_ies):
    for wlan_ie in wlan_ies:
        if wlan_ie.id == 3:
            return wlan_ie.ch
    else:
        return ""

def get_seq_no(rawdata, radiotap_len):
    index = radiotap_len + SEQ_NO_INDEX
    i, = struct.unpack("<H", rawdata[index:index + 2])
    return i >> 4

def parse_probe_reqs(pcap_path):

    pkt_dicts = []
    pc = dpkt.pcap.Reader(open(pcap_path))
    if pc.datalink() != 127:  # Check if RadioTap
        return
```

```
probe_resp = [] # List of probe response timestamp

for timestamp, rawdata in pc:
    try:
        radiotap = dpkt.radiotap.Radiotap(rawdata)

        # radiotap_len field indicates the entire length
            of the radiotap data, including the radiotap
            header.
        radiotap_len = binascii.hexlify(rawdata[2:3])
        radiotap_len = int(radiotap_len, 16)
        wlan = dpkt.ieee80211.IEEE80211(rawdata[
           radiotap_len:])

        # Skip if not a probe request
        if wlan.type != 0 or wlan.subtype != 4:
            if wlan.type ==0 and wlan.subtype == 5: #
               probe response(duplicate not removed)
                probe_resp.append(timestamp)
            continue

        mac = binascii.hexlify(wlan.mgmt.src)
        # skip if not SSID from the MAC address aa:bb:cc
           :dd:ee:ff
        if mac != 'aabbccddeeff':
            continue


        try:
            ssid = wlan.ies[0].info
            ssid = ssid if len(ssid) else "Broadcast"
        except:
            ssid = "<SSID Decode error>"

        chan_no = get_current_chan(wlan.ies)
        seq_no = get_seq_no(rawdata, radiotap_len)

        if chan_no >=1 and chan_no <=13: # Skip 5GHz
           bands
            pkt_dict = {"chan_no": chan_no, "seq_no":
               seq_no, "timestamp": timestamp}
            pkt_dicts.append(pkt_dict)

    except (NeedData, UnpackError, KeyError):
```

```
            # probably due to malformed packets
            continue
        except Exception as exc:
         #   print "Exception", exc, traceback.format_exc()
            continue


        with open('probe_resps_ts_list.txt', 'w')as f:
            for line in probe_resp:
                f.write(str(line)+',')

    return (pkt_dicts,probe_resp)

def trim(probe_req_dicts):
    MIN_PROBE_REQ_TIMEDIFF = 0.0005
    trimmed_probe_reqs = []
    last_probe_req = None

    for probe_req_dict in probe_req_dicts:
        duplicate = False
        if last_probe_req:
            if probe_req_dict["seq_no"] == last_probe_req["
               seq_no"]:
                if probe_req_dict["chan_no"] ==
                   last_probe_req["chan_no"]:
                    if (probe_req_dict["timestamp"] -
                       last_probe_req["timestamp"]) <
                       MIN_PROBE_REQ_TIMEDIFF:
                        duplicate = True
        if not duplicate:
            trimmed_probe_reqs.append(probe_req_dict)

        last_probe_req = probe_req_dict

    return trimmed_probe_reqs

def param_calc(probe_req_dicts, probe_resps_ts_list):

    # order of channel probed, first PRF channel:
    with open('Channel_order.txt', 'w')as f1,open('
       First_PRF_channel.txt', 'w') as f2:
        f1.write('\nSequence order of probed channel:\n')
        f2.write('\nFirst PRF channel:')
        last_probe_req = None
        burst = 1
```

```
        for probe_req_dict in probe_req_dicts:
            if last_probe_req:
                if abs(probe_req_dict["chan_no"] -
                   last_probe_req["chan_no"]) > 5:
                    f1.write('\nNew Burst:\n')
                    burst += 1
                    f2.write(str(probe_req_dict["chan_no"])
                       +' ')
            f1.write(str(probe_req_dict["chan_no"])+' ')
            last_probe_req = probe_req_dict
        f1.write('\nTotal number of bursts:' + str(burst) +
           '\n')

    # probe delay:
    with open('Number_of_PRF_per_chanel_and_delay.txt', 'w')
       as f:
        f.write('\nNumber of PRF per channel:')
        last_probe_req = None
        hit_list = []
        hit = 1
        delay_list = []
        for probe_req_dict in probe_req_dicts:
            if last_probe_req:
                if last_probe_req["chan_no"] ==
                   probe_req_dict["chan_no"]:
                    hit += 1
                    delay_list.append(probe_req_dict["
                       timestamp"]-last_probe_req["timestamp
                       "])
                else:
                    hit_list.append(hit)
                    hit = 1
            last_probe_req = probe_req_dict
        f.write(str(hit_list)+'\n')

        # Most likely hit:
        counts = numpy.bincount(hit_list)
        most_likely_hit = numpy.argmax(counts)
        f.write("\nMost likely number of PRF per channel is:
           "+str(most_likely_hit)+'\n')

        # Delay and calculation:
        if most_likely_hit !=1:
            f.write('\nDelay: ' + str(delay_list) + '\n')
```

```
        f.write('\nThere are ' + str(len(delay_list)) +
            ' Delay values.\n')
        f.write('Mean: ' + str(numpy.mean(delay_list)) +
             ' Std: ' + str(numpy.std(delay_list)))
        f.write('\n')
    else:
        f.write("\nNo delay calculation as the most
            likely probe request per channel is 1.\n")

# MinCT and MaxCT:
# assume MinCT is the time difference betwwen different
    channels when there is NO probe response
# assume MaxCT is the time difference betwwen different
    channels when there is probe response
MinCT_list = []
MaxCT_list = []
last_probe_req = None
last_probe_req_2 = None
last_probe_req_3 = None

if most_likely_hit==1:
    for probe_req_dict in probe_req_dicts:
        if last_probe_req:
            if (probe_req_dict["chan_no"] -
                last_probe_req["chan_no"]) == 1:  #
                adjacent channel (no missing packets)
                 probe_resp_found = False
                 for ts in probe_resps_ts_list:
                     if ts>=last_probe_req["timestamp"]
                        and ts<=probe_req_dict["timestamp
                        "]: # Probe response: Case of
                        MaxCT
                         MaxCT_list.append(probe_req_dict
                             ["timestamp"]-last_probe_req
                             ["timestamp"])
                         probe_resp_found = True
                         break
                 if not probe_resp_found:
                     MinCT_list.append(probe_req_dict["
                        timestamp"] - last_probe_req["
                        timestamp"])
        last_probe_req = probe_req_dict

elif most_likely_hit==2:
    for probe_req_dict in probe_req_dicts:
```

```
            if last_probe_req:
                if last_probe_req_2:
                    if last_probe_req_3:
                        if probe_req_dict["chan_no"] ==
                           last_probe_req_3["chan_no"]:
                            if last_probe_req_2["chan_no"]
                               == last_probe_req["chan_no"]:
                                if last_probe_req_3["chan_no
                                   "]-last_probe_req["
                                   chan_no"]==1: # no
                                   missing packets: complete
                                    4 probes for two
                                   adjacent channels
                                    probe_resp_found = False
                                    for ts in
                                       probe_resps_ts_list:
                                        if ts>=
                                           last_probe_req["
                                           timestamp"] and
                                           ts<=
                                           last_probe_req_3
                                           ["timestamp"]: #
                                           Probe response:
                                           Case of MaxCT
                                            MaxCT_list.
                                               append(
                                               last_probe_req_3
                                               ["timestamp
                                               "]-
                                               last_probe_req
                                               ["timestamp
                                               "])
                                            probe_resp_found
                                                = True
                                            break
                                    if not probe_resp_found:
                                        MinCT_list.append(
                                           last_probe_req_3
                                           ["timestamp"] -
                                           last_probe_req["
                                           timestamp"])
                last_probe_req = last_probe_req_2
                last_probe_req_2 = last_probe_req_3
                last_probe_req_3 = probe_req_dict
    else:
```

```
        print "No result for MinCT/MaxCT because the most
            likely hit is neither 1 nor 2!"

    #calculate mean and standard derivation of MinCT and
        MaxCT
    with open('MinCT_MaxCT.txt','w')as f:
        f.write('\nMinCT: '+ str(MinCT_list) + '\n')
        f.write('\nThere are ' + str(len(MinCT_list)) + '
            MinCT values.\n')
        f.write('Mean: ' + str(numpy.mean(MinCT_list)) + '
            Std: ' + str(numpy.std(MinCT_list)) + '\n')
        f.write('\nMaxCT: ' + str(MaxCT_list) + '\n')
        f.write('There are ' + str(len(MaxCT_list)) + '
            MaxCT values.\n')
        f.write('Mean: ' + str(numpy.mean(MaxCT_list)) + '
            Std: ' + str(numpy.std(MaxCT_list)) + '\n')
        f.write('\n')

    # merge all files
    files = ['Channel_order.txt', 'First_PRF_channel.txt', '
        Number_of_PRF_per_chanel_and_delay.txt', 'MinCT_MaxCT
        .txt']
    with open('result_030517_old_linux_u16.txt', 'w') as f:
        for f1 in files:
            with open(f1) as f2:
                f.write(f2.read())
        f.write('\n')
    print "The result file was created."

    return


if __name__ == "__main__":

    (pkt_dicts, probe_resps_ts_list) = parse_probe_reqs('
        captured_packet.pcap')

    trimmed_probe_req_dicts = trim(pkt_dicts)

    param_calc(trimmed_probe_req_dicts,probe_resps_ts_list)
```

Bibliography

[1] Android open source code: WPA Supplicant mlme.c in Jelly Bean 4.1.2. http://androidxref.com/4.1.2/xref/external/wpa_supplicant_6/wpa_supplicant/mlme.c#2605. Accessed: 2017-06-07.

[2] WiFiGate - how mobile carriers expose us to Wi-Fi attacks. https://www.skycure.com/blog/wifigate-how-mobile-carriers-expose-us-to-wi-fi-attacks. Accessed: 2017-06-03.

[3] Wikipedia: List of WLAN channels. https://en.wikipedia.org/wiki/List_of_WLAN_channels. Accessed: 2017-05-11.

[4] Wireshark. https://www.wireshark.org. Accessed: 2017-05-19.

[5] Wireless LAN medium access control (MAC) and physical layer (PHY) specification, 2003.

[6] Wireless LAN medium access control (MAC) and physical layer (PHY) specification, 2007.

[7] Wireless LAN medium access control (MAC) and physical layer (PHY) specification, 2012.

[8] Wireless LAN medium access control (MAC) and physical layer (PHY) specification, 2016.

[9] A. Arcia-Moret, L. Molina, G. Castignani, and N. Montavont. Characterizing spontaneous ieee 802.11 network deployments. In *Network Games, Control and Optimization (NetGCooP), 2012 6th International Conference on*, pages 1–8. IEEE, 2012.

[10] A. Arcia-Moret, L. Molina, N. Montavont, G. Castignani, and A. Blanc. Access point discovery in 802.11 networks. In *Wireless Days (WD), 2014 IFIP*, pages 1–6. IEEE, 2014.

[11] G. Castignani, A. Arcia, and N. Montavont. A study of the discovery process in 802.11 networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 15(1):25–36, 2011.

[12] G. Castignani, N. Montavont, and A. Arcia-Moret. Analysis and evaluation of WiFi scanning strategies. *Proceeding of IV Cibelec*, pages 3–7, 2010.

[13] G. Castignani, N. Montavont, A. Arcia-Moret, M. Oularbi, and S. Houcke. Cross-layer adaptive scanning algorithms for IEEE 802.11 networks. In *2011 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 327–332. IEEE, 2011.

[14] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale Wi-Fi localization. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 233–245. ACM, 2005.

[15] M. Cunche. Wi-Fi told me everything about you. http://confiance-numerique.clermont-universite.fr/Slides/M-Cunche-2014.pdf. Accessed: 2017-05-09.

[16] M. Cunche. I know your MAC address: Targeted tracking of individual using wi-fi. *Journal of Computer Virology and Hacking Techniques*, 10(4):219–227, 2014.

[17] M. Cunche, M.-A. Kaafar, and R. Boreli. Linking wireless devices using information contained in Wi-Fi probe requests. *Pervasive and Mobile Computing*, 11:56–69, 2014.

[18] J. Freudiger. Short: 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*, WiSec '15. ACM, 2015.

[19] D. Gentry and A. Pennarun. Passive taxonomy of Wifi clients using MLME frame contents. *arXiv preprint arXiv:1608.01725*, 2016.

[20] F. Goovaerts. Passive Wi-Fi scanning: a viability study. Master's thesis, KU Leuven, 2017.

[21] V. Gupta, R. Beyah, and C. Corbett. A characterization of wireless NIC active scanning algorithms. In *Wireless Communications and Networking Conference 2007. WCNC 2007.*, pages 2385–2390. IEEE, 2007.

[22] Y. S. Kim, Y. Tian, L. T. Nguyen, and P. Tague. Lapwin: Location-aided probing for protecting user privacy in wi-fi networks. In *Communications and Network Security (CNS), 2014 IEEE Conference on*, pages 427–435. IEEE, 2014.

[23] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, and D. Brown. A study of MAC address randomization in mobile devices and when it fails. *arXiv preprint arXiv:1703.02874*, 2017.

[24] C. Matte, M. Cunche, F. Rousseau, and M. Vanhoef. Defeating MAC address randomization through timing attacks. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 15–20. ACM, 2016.

[25] A. Musa and J. Eriksson. Tracking unmodified smartphones using Wi-Fi monitors. In *Proceedings of the 10th ACM conference on embedded network sensor systems*, pages 281–294. ACM, 2012.

[26] P. Robyns, B. Bonné, P. Quax, and W. Lamotte. Noncooperative 802.11 MAC layer fingerprinting and tracking of mobile devices. *Security and Communication Networks*, 2017, 2017.

[27] A. Soltani. Mobile device tracking. https://ashkansoltani.files.wordpress.com/2014/02/soltani-ftc-mobile-device-tracking-2014-02-19.pdf. Accessed: 2017-05-09.

[28] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. 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*, pages 413–424. ACM, 2016.

[29] H. Velayos and G. Karlsson. Techniques to reduce the IEEE 802.11 b handoff time. In *Communications, 2004 IEEE International Conference on*, volume 7, pages 3844–3848. IEEE, 2004.

[30] H. Wu, K. Tan, J. Liu, and Y. Zhang. Footprint: cellular assisted Wi-Fi AP discovery on mobile phones for energy saving. In *Proceedings of the 4th ACM international workshop on Experimental evaluation and characterization*, pages 67–76. ACM, 2009.

[31] F. Xia, W. Zhang, F. Ding, and R. Hao. A-GPS assisted Wi-Fi access point discovery on mobile devices for energy saving. In *Global Information Infrastructure Symposium (GIIS), 2011*, pages 1–6. IEEE, 2011.

Master thesis filing card

Student: Yun-Chih Joy Chou

Title: Characterizing Wi-Fi Network Discovery

UDC: 621.3

Abstract:
In the Wi-Fi network discovery phase, the probe request sent by a mobile device is not encrypted. Therefore, the MAC address contained in the probe request frame can be exploited to track users without their knowledge and consent. To counter this problem, modern operating systems implement MAC address randomization. However, studies showed that the spoofed MAC address can still be reversed. In addition, studies showed that some devices send the Service Set Identifiers (SSIDs) of the previously connected networks. Such SSID list reveals sensitive personal information such as the locations one has been to. In order to help understand how much the probe requests reveal the behavior or traces of a user, this thesis characterizes the scanning behavior of the mobile device during network discovery. The different scenarios we tested include different numbers of the saved networks, different connection status, and mobile configurations. The characterized parameters include the channel sequence order, the number of probe request frames sent per channel, the interprobe delay time, and the minimum and maximum channel time a device spent on each channel (`MinChannelTime`, `MaxChannelTime`). We test five different devices and find that all of them send one or two probe requests per channel. All devices except the Windows PC set the same value for the minimum and maximum channel time, which ranged from 20 ms to 105 ms. Our experiments show that MacBook prioritizes the channels of the saved networks, which reduces the connection time. Our results show that devices exhibit different scanning behavior in different settings. We found that three of the five devices tested used randomized MAC addresses, and almost none of them sent the SSIDs of the previously connected networks. We also observed that, some devices with MAC randomization reveal their true MAC addresses once connected to a network.

Thesis submitted for the degree of Master of Science in Electrical Engineering, option Embedded Systems and Multimedia

Thesis supervisor: Prof. dr. Claudia Diaz

Assessors: Prof. dr. ir. Bart Preneel
Prof. dr. ir. Sofie Pollin

Mentors: Dr. Gunes Acar
Rafael Gálvez Vizcaíno