POLITECNICO
MILANO 1863

WIRELESS INTERNET

PROJECT REPORT

AY 2021/2022

Vitobello Andrea
personal code XXXXXXXX, matriculation number XXXXXX

Index

1. Abstract

The objective of this project is to produce a set of probe requests captures generated by one or more mobile devices under different conditions, such as Wi-Fi on/off, power save mode on/off and active/inactive screen, to analyze the behavior of probe request generation and MAC randomization. The output of this project, additionally to the report at hand, is a set of pcap files for each capture and a python colab page required to filter and analyze the capture.

In the project zip directory you can find two different folders containing a set of pcap files each: in the "Raw Captures" folder are provided the captures right out from Wireshark, while in "Filtered Captures" are the pcap generated through the python code. Further information on the filtering phase is provided in the following paragraphs.

This report first explains the basic necessary concepts so that even unknowledgeable readers can understand the project, then it deep dives into the details of the practical experiment, describing both the means and procedures that were followed to complete it, as well as a brief analysis of the results.

2. Introduction

2.1 Active scanning and Probe requests

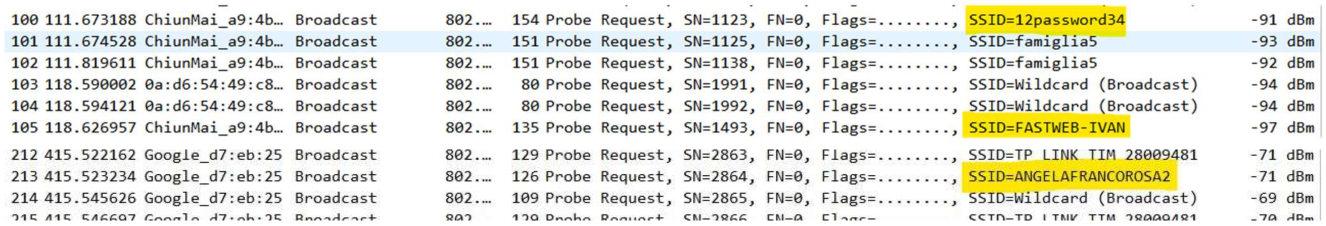
The WI-FI technology can only work if devices associate to each other to establish connections; for this purpose, it is necessary to accomplish the management procedure called scanning, which is performed by stations (STA) that are trying to connect to a basic service set (BSS) through an access point (AP). After this procedure the STA will produce a scanning report that lists all the BSSs it interacted with, as well as their main characteristics, such as BSSID, frequency of beacons and physical layer information: the STA can then automatically select what access point associate with or let the user manually choose [1], [2].

The scanning procedure can either be passive or active. In passive scanning the STA listens to the available channels through their periodic beacons, and thanks to such beacons it creates the scanning report. In active scanning, the STA proactively sends probe requests to each available channel, either with unicast or broadcast messages, to solicit the transmission of beacons. Active mode allows to perform the scanning process more efficiently, but it also carries more risks due to the proactive role of the STA and the rich information that can be found in active probe requests.

2.2 Privacy Issues and MAC randomization

Wi-Fi active probe requests contain the MAC address of the sending device, called source, and without any MAC randomization they leak relevant private information: they allow to track the device owner through indoor localization, similarly to what has been done with extreme ease with passive probe requests [3], [4], [5], but also in a broader setting such as whole cities [6]. Leaks also include past access point identifiers such as a list of preferred SSIDs, names or usernames that would allow profiling device users [7].

The reader is now invited to sneak a peek at the capture *Raw Captures\1_Probes_Active.pcap*, where among the SSIDs can be observed many names and possibly even passwords.



```
100 111.673188 ChiunMai_a9:4b… Broadcast   802.…  154 Probe Request, SN=1123, FN=0, Flags=........, SSID=12password34          -91 dBm
101 111.674528 ChiunMai_a9:4b… Broadcast   802.…  151 Probe Request, SN=1125, FN=0, Flags=........, SSID=famiglia5             -93 dBm
102 111.819611 ChiunMai_a9:4b… Broadcast   802.…  151 Probe Request, SN=1138, FN=0, Flags=........, SSID=famiglia5             -92 dBm
103 118.590002 0a:d6:54:49:c8… Broadcast   802.…   80 Probe Request, SN=1991, FN=0, Flags=........, SSID=Wildcard (Broadcast)  -94 dBm
104 118.594121 0a:d6:54:49:c8… Broadcast   802.…   80 Probe Request, SN=1992, FN=0, Flags=........, SSID=Wildcard (Broadcast)  -94 dBm
105 118.626957 ChiunMai_a9:4b… Broadcast   802.…  135 Probe Request, SN=1493, FN=0, Flags=........, SSID=FASTWEB-IVAN          -97 dBm
212 415.522162 Google_d7:eb:25 Broadcast   802.…  129 Probe Request, SN=2863, FN=0, Flags=........, SSID=TP LINK TIM 28009481  -71 dBm
213 415.523234 Google_d7:eb:25 Broadcast   802.…  126 Probe Request, SN=2864, FN=0, Flags=........, SSID=ANGELAFRANCOROSA2     -71 dBm
214 415.545626 Google_d7:eb:25 Broadcast   802.…  109 Probe Request, SN=2865, FN=0, Flags=........, SSID=Wildcard (Broadcast)  -69 dBm
215 415.546697 Google_d7:eb:25 Broadcast   802    129 Probe Request   SN=2866  FN=0  Flags=          SSID=TP LINK TIM 28009481  -70 dBm
```

Several countermeasures have been proposed [7] such as completely removing link layer identifiers, which would effectively break the Wi-Fi technology, or changing MAC address after a period of time is elapsed, however the adopted solution was MAC randomization, introduced by Apple Inc. to enhance the privacy offered to its users [8]. After Apple, many electronic devices and software producers started implementing such technique, as demonstrated by the project carried out by Pintor and Aztori [9], including Linux [10] as well as android devices [11].

Through MAC randomization techniques it is possible to send probe requests, as the name suggests, with dummy and randomized MAC addresses instead of the factory MAC address provided by the Network Interface Controller, NIC, or the mobile device equivalent, so that the sniffing attacks presented above are *mitigated*. As a matter of fact, the Wi-Fi communication is still far from secure as it is possible to track devices and users in other ways, such as using different data contained in probe requests or by bypassing MAC randomization [12],[13],[14].

It is worth mentioning that MAC randomization techniques vary depending on the device producer or even on the device model [9].

2.3 Sidenote: Received Signal Strength Indicator, RSSI

The RSSI is a measure that expresses how well a device can hear signals from another device; to rephrase it, the RSSI represents the quality of the transmission between two devices. In this project the RSSI was expressed in decibels, through a logarithmic scale. Since we are working with decibels it means that the closer the value is to 0 the better, and the higher the absolute value the worse. Additionally, values are negative only, and they can reach a maximum absolute value of -110 dbm.

As an example: a signal with RSSI equal to -20 dbm was better received than another signal with RSSI equal to -60 dbm.

3. Project execution

3.1 Set-up

Three devices were used to carry out the experiment, and out of these were a laptop and two mobile devices.

- ➔ Samsung SM-T535 with Android 5.0.2, tablet, Subject A
- ➔ Xiaomi MI A2 lite with Android 10, mobile phone, Subject B
- ➔ DELL latitude e6410, laptop

The first two devices were used to provide the necessary probe requests, while the third one was used to capture the frames. Since the laptop runs on windows OS, it was necessary to boot it with a kali live distribution from a USB drive so that enabling monitor mode could be easily achieved. The specific kali distribution was *kali-linux-2022.2-live-amd64*. Once superuser privileges were obtained, monitor mode could be easily activated through the following commands

```
ifconfig wlan0 down
iwconfig wlan0 mode monitor
```

3.2 Wireshark captures

The experiment could not be carried out in an isolated room, however it was still possible to limit unwanted Wi-Fi activities to some degree, so that pruning random frames and removing unwanted probe requests from third parties would be easier. At each capture, the subject device was placed right next to the laptop capturing packets so that the RSSI could be as good as possible.

Due to the nature of the project and to avoid any possible confusion, the two mobile devices were separately analyzed. In both cases the inspected operational modes were 9:

1) WIFI off
2) WIFI on + Active screen → 1_Probes_Active.pcap
3) WIFI on + Inactive screen → 2_Probes_Inactive.pcap
4) WIFI on + Active screen + Power Save mode ON → 3_Probes_Active+PS.pcap
5) WIFI on + Inactive screen + Power Save mode ON → 4_Probes_Inactive+PS.pcap
6) WIFI on + Active screen + Plane mode ON → 5_Probes_Active+Plane.pcap
7) WIFI on + Inactive screen + Plane mode ON → 6_Probes_Inactive+Plane.pcap
8) WIFI on + Active screen + Bluetooth ON → 7_Probes_Active+BT.pcap
9) WIFI on + Inactive screen + Bluetooth ON → 8_Probes_Inactive+BT.pcap

Of course, when the Wi-Fi option was off no probe request was sent and thus the corresponding captures are not reported in the "Raw Captures" folder, while in all other cases the process was effective.

Each capture was executed in a 15 minute time slot so that the probe requests generated could be numerous enough for the subsequent analysis. The Wireshark captures recorded not only probe requests but all kinds of Wi-Fi packets flowing through the medium at the time: to lighten the capture files the following reasonable Wireshark display filter was applied:

```
wlan.fc.type_subtype == 0x0004
```

This specific filter allows to select only packets whose type and subtype match 0x04, which corresponds to active probe requests management frame, type 0b0000 and subtype 0b0100[1]. The result of this first step is the "Raw Captures" folder, that contains a zip file for each relevant capture.

3.3 Supplementary python code

Thanks to the Wireshark display filter it was possible to trim all packets recorded in the 15 minutes slots to only active probe requests. The result of this filtering operation is included in the zip as "Raw Capture" folder. As the name of the folder suggests, however, such filtering operation was still insufficient to produce the expected pcap output, as well as provide with some useful analytics: here is where python comes to lend a helping hand. Inside the project zip folder is included a colab page with all the python code used in this project as well as a step-by-step commentary. The used python code is an elaboration of the examples showed by prof. Alessandro E.C. Redondi [1].

The code is divided into four main sections:

I. *Setting the environment*: here libraries are installed and imported.

II. *Check: probe requests are all valid*: in this section it is possible to use as input one of the captures in the "Raw Captures" folders by importing it on colab and modifying the name of the variable *captureName*. All the subsequent computation will produce results about such capture. The code then checks that all packets included in the capture are indeed valid and can be used to produce useful data.

III. *Filtering probe requests and checking randomicity*: this section carries out the second filtering operation; due to the nature of the experiment it is impossible to discriminate based on the source MAC address, so the filter uses the radio signal strength indicator, RSSI, as a discriminator. The RSSI is a characteristic of the capture and can be checked upon both on Wireshark and python. It was indeed noticed that most unwanted activities came from the (-90dbm,-55dbm) range, thus the best option was to consider as "valid" only captures from source addresses that had recorded at least a capture with RSSI better than -50.

    ```
    wlan_radio.signal_dbm > -50
    ```

    Thanks to this filter it was possible to approximately determine how many source addresses are *valid* out of all the recorded ones. After checking that all valid source addresses are in fact randomized, this section creates a new filtered capture .pcap file that includes only packets coming from valid addresses. It is in this step that the capture files included in the "Filtered

[1] 0x04 expresses the number in hexadecimal notation, while 0b0000 and 0b0100 use the binary notation. The conversion of 0x04, first digit for the type and second for the subtype, to the binary format is exactly 0b00000100.

Captures” folder are produced. Notice that the output required by the project is the “Filtered Captures” folder, while the “Raw Captures” folder is provided for transparency and so that the reader can reproduce the experiment if s/he so desires.

IV. *Average RSSI and IAT*: this last section aims at producing some useful data. First all relevant information is displayed after grouping it by source address, then mean RSSI and IAT are calculated for each source address. The IAT, Interval Arrival Time, expresses the elapsed time between two probe request arrivals from the same source address. Lastly, the average[2] and the RDP[3] are determined for both RSSI and IAT.

3.4 Results presentation

Subject A did not show any randomization. As a matter of fact, the android version of such device is antecedent to the first software version that introduced MAC randomization [11]. All captures used the factory MAC, and the only notable behavior was that probe requests were sent only when the screen was active. Captures from subject A were discarded as they are essentially useless for the purpose of the experiment. The discussion regarding subject A, however, was still included in the report as it is an interesting example of how old and unsupported technologies can be a source of vulnerabilities and privacy leakages.

Subject B showed MAC randomization in all captures where it sent active probe requests. Here are the results derived from each capture file through the python code.

1_Probes_Active frame length 108 bytes total packets: 542 valid packets: 158 total SA: 70 valid SA: 6 Average RSSI: -51.8288; RSSI RDP: 6.4332% Max RSSI: -50.4783; Min RSSI: -53.8125 Average IAT: 0.0124; IAT RDP: 21.5667% Max IAT: 0.0135; Min IAT: 0.0109	**2_Probes_Inactive** frame length 68 bytes total packets: 503 valid packets: 50 total SA: 190 valid SA: 35 Average RSSI: -34.4143; RSSI RDP: 61.0212% Max RSSI: -25.0; Min RSSI: -46.0 Average IAT: 0.025; IAT RDP: 1.7956% Max IAT: 0.0252; Min IAT: 0.0247
3_Probes_Active+PS frame length 108 bytes total packets: 350 valid packets: 136 total SA: 60 valid SA: 6 Average RSSI: -51.4417; RSSI RDP: 7.4032% Max RSSI: -49.4; Min RSSI: -53.2083 Average IAT: 0.012; IAT RDP: 19.7713% Max IAT: 0.0133; Min IAT: 0.0109	**4_Probes_Inactive+PS** frame length 68 bytes total packets: 431 valid packets: 45 total SA: 146 valid SA: 32 Average RSSI: -33.7812; RSSI RDP: 63.6448% Max RSSI: -23.5; Min RSSI: -45.0 Average IAT: 0.0251; IAT RDP: 1.6421% Max IAT: 0.0254; Min IAT: 0.025
5_Probes_Active+Plane frame length 108 bytes total packets: 532 valid packets: 143 total SA: 69	**6_Probes_Inactive+Plane** frame length 68 bytes total packets: 425 valid packers: 43 total SA: 189

[2] Notice that the mean RSSI and IAT are calculated for each source address and thus produce two values for each source address, while the average RSSI and IAT are calculated from all source addresses and produce a single value for the whole capture

[3] The RDP is the Relative Percentage Difference, and it expresses how much the values of mean RSSI and IAT vary in the space of a whole capture among different source addresses

```
valid SA: 6                                        valid SA: 33
Average RSSI: -50.3365;      RSSI RDP: 14.0463%   Average RSSI: -33.0455;      RSSI RDP: 127.097%
Max RSSI: -46.1905;          Min RSSI: -53.2609   Max RSSI: -14.0;             Min RSSI: -56.0
Average IAT: 0.0128;         IAT RDP: 16.8576%    Average IAT: 0.0251;         IAT RDP: 1.4022%
Max IAT: 0.0138;             Min IAT: 0.0116      Max IAT: 0.0253;             Min IAT: 0.0249
```

```
7_Probes_Active+BT                                 8_Probes_Inactive+BT
frame length 108 bytes                             frame length 68 bytes
total packets: 387                                 total packets: 310
valid packers: 151                                 valid packets: 42
total SA: 44                                       total SA: 145
valid SA: 7                                        valid SA: 31
Average RSSI: -50.0361;      RSSI RDP: 11.2962%   Average RSSI: -35.2903;      RSSI RDP: 51.0055%
Max RSSI: -46.6522;          Min RSSI: -52.3043   Max RSSI: -27.0;             Min RSSI: -45.0
Average IAT: 0.0131;         IAT RDP: 27.5084%    Average IAT: 0.025;          IAT RDP: 1.6172%
Max IAT: 0.0152;             Min IAT: 0.0116      Max IAT: 0.0252;             Min IAT: 0.0248
```

Total packets: total packets of the capture

Valid packets: packets that come from "valid" source addresses, and that are saved in the filtered captures

Total SA: total number of source addresses analyzed in the capture

Valid SA: source addresses that were considered as coming from the device under study

3.5 Brief analysis

The data above was the result of subject B captures.

At first only captures 1 to 4 were executed, with the addition of the Wi-Fi off capture that is not included because of the lack of probe requests. After a brief analysis, though, it seemed quite probable that the activation of Power Saving Mode had no effect at all on how the device sends probe requests; for this reason, two additional interesting settings were checked: Plane Mode and Bluetooth, both with active and inactive screen. It is clear from the results that the only relevant difference in the specific case of device Xiaomi MI A2 lite with Android 10 is whether the screen is active or not.

- Active screen: the device sends bursts of probe requests at different RSSI levels and from the same, randomized source address, waits for a few minutes and then, after changing source address, repeats the process. This pattern is repeated about 6 times in 15 minutes. Since the number of valid packets is so high even though the valid addresses are so few it means that for each randomization the device has sent a relevant quantity of probes. The RSSI RDP is generally quite low, which means that each burst of probe requests roughly shows the same behavior from this point of view. The IAT RDP, however, is considerably high, meaning that each burst carries some level of erraticism regarding the inter arrival time.
- Inactive screen: the device now behaves far more erratically. The number of valid source addresses is much higher while the number of valid packets is drastically lower, and in particular the corresponding filtered captures show that each randomized source MAC address only sends about 1 or 2 probe requests before randomizing again. Additionally, the total number of source addresses is disproportionately large in comparison to active screen scenarios, thus by also considering the highly variable RSSI it is possible to conclude that the device sent many probe requests from randomized MAC with RSSI $<$ -50 that were discarded. Lastly, the IAT values recorded in these captures can be deceptive to some degree: from the python colab section III we can see that, in most cases, a randomized MAC

only sent a single packet, so the corresponding IAT is null and is thus not considered; this means that the pool of valid IAT is scarcer than in the active screen captures and mainly considers burst of just 2 requests, which seem to be regularly spaced.

4. Conclusion

This project allowed to analyze how different devices send probe requests and perform MAC randomization. The first device, subject A, did not show any MAC randomization as it was too old and used an android version that does not support it. This means that old devices can be a source of privacy leakage and places the emphasis on how device updates and support should be extended, at the very least, for security related patches.

The second device, subject B, does indeed implement MAC randomization, however the only discriminator is whether the screen is active or not. As it has been explained, though, no unique and widespread MAC randomization algorithm exists so the results might vary further if other devices are analyzed.

Bibliography

[1] Alessandro E.C. Redondi, “Wireless Internet” lectures, labs and Slides

[2] James F. Kurose, Keith W. Ross, “Computer Networking”

[3] Alessandro E.C. Redondi, Matteo Cesana, “Building up knowledge through passive WiFi probes”

[4] Suining He, S.-H. Gary Chan, “Wi-Fi Fingerprint-Based Indoor Positioning: Recent Advances and Comparisons”

[5] Faheem Zafari, Athanasios Gkelias, Kin K. Leung, “A Survey of Indoor Localization Systems and Technologies”

[6] https://www.wired.com/2014/08/edward-snowden/

[7] Julien Freudiger, “How talkative is your mobile device? An experimental study of Wi-Fi probe requests”

[8] https://support.apple.com/guide/security/wi-fi-privacy-secb9cb3140c/web

[9] Lucia Pintor, Luigi Atzori, “dataset of labelled device Wi-Fi probe requests for MAC address de-randomization”

[10]https://github.com/torvalds/linux/commit/effd05ac479b80641835f9126bbe93146686c2b8

[11]https://developer.android.com/about/versions/marshmallow/android-6.0-changes

[12] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S. Cardoso, Frank Piessens, iMinds-Distrinet, KU Leuven, Univ Lyon, INSA Lyon, Inria, CITI, France, “Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms”

[13] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C. Rye, and Dane Brown, “A Study of MAC Address Randomization in Mobile Devices and When it Fails”

[14] Célestin Matte, Mathieu Cunche, Franck Rousseau, Mathy Vanhoefq, “Defeating MAC Address Randomization Through Timing Attacks”