POLITECNICO DI MILANO

DEPARTMENT OF ELECTRONICS, INFORMATION AND BIOINGINEERING

M.Sc. Course in

Wireless Internet

Project on:

Counting how many people are present in a classroom via traffic sniffing

Professor: Dr. ALESSANDRO ENRICO CESARE REDONDI

Students:

Hiva Amiri

A.Y 2019-2020

Contents

1 Introduction

Counting the number of visitors or devices on a WiFi network is a straightforward metric that organizations are tracking at physical locations. There are several reasons why organizations need to track this. Sometimes it’s to ensure a positive customer experience or to help in the sales process. Sometimes it’s because there is a regulation or KPI that needs these numbers to be reported. Finally, sometimes it’s tracked for employee productivity or energy savings.

Whatever the reason for needing to count the number of wireless users, from a solution standpoint, it usually comes down to one basic question, how can we do it?

From a technical point of view, there are a lot of technics to do it. One of the easies way is counting the number of users associated to the wifi network. It can be measured easily by monitoring associated users to the access point. On the other hand, we can use more specific process and monitor wifi packets sending around.

This method is somehow more complex and requires more fund to invest since it needs a device to monitor and process captured data. This project goes through one of the technics in this regard by sniffing prob request from users.

2 IEEE 802.11 Probe Requests

Since this project is done based on Probe Requests, let’s have a summary about it.

Most of the packets flying around which govern Access-Point-to-Client connectivity do indeed originate from the Access Points. However, the 802.11 specification (...WiFi) also allows personal devices to send out 'Probe Requests' blindly around them in the hope that a nearby familiar access point will realise "Hey, that's me! I better Beacon again myself so that device can see me and then elect to connect to me".

But what is a 'familiar access point'? Well, every time you add a new WiFi network to your phone, you essentially add it to your 'favorites'. Most people then forget this and build up a massive list of 'Saved Networks'. Take a look now on your Android or iOS phone, the list is usually in the WiFi settings somewhere.

When your device has WiFi enabled but is not connected to an Access Point it sends out probe requests, sometimes Broadcasts (to 'anyone' who is an access point to respond to), but other times it will explicitly list the name of the network it wishes to connect to. That is what we are interested in.

Since acquiring this data is ridiculously easy, it can be used to count the number of people present in a specific place, in our project, the number of students in a class.

3 Technical requirement

To see the packets we are interested in, we need to put our device wireless adaptor into Monitor Mode. It may some devices doesn't support Monitor Mode. In this case, we are using a HP laptop with Intel wireless adaptor and the codes run on an Ubuntu OS.

To put your machine into Monitor mode, these codes needs to be executed:

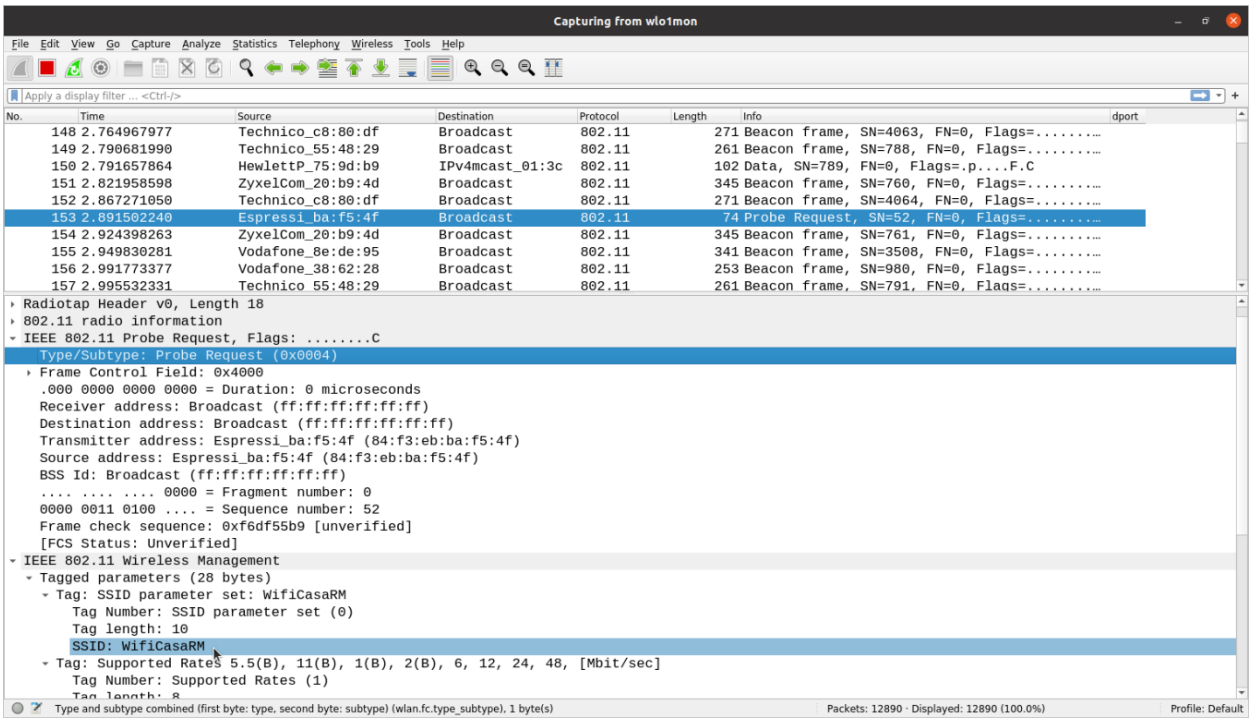
Set OS to universe repo	`sudo add-apt-repository universe`
Install a tool to w-adaptor into MM	`sudo apt install aircrack-ng`
Finding the name of w-adapter interface	`ip a`
Call the app to put w-adaptor to MM	`sudo airmon-ng start wlo1`

Now, a new name should be assigned to the Monitor Mode wireless adapter, it usually has the same name as wireless adaptor interface+mon which means monitor mode.

Here is a response of above commands:

```
ubuntu@ubuntu:~/scapy$ sudo airmon-ng start wlo1

Found 4 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

    PID Name
   1487 avahi-daemon
   1491 NetworkManager
   1517 wpa_supplicant
   1535 avahi-daemon

PHY    Interface    Driver       Chipset

phy0   wlo1         iwlwifi            Intel Corporation Centrino Advanced-N 6205
[Taylor Peak] (rev 34)

              (mac80211 monitor mode vif enabled for [phy0]wlo1 on [phy0]wlo1mon)
              (mac80211 station mode vif disabled for [phy0]wlo1)
```

To have better understanding about what we are looking for, a sniff was captured using Wireshark and the field we are looking for is highlighted in the picture of the next page.



From the capture, the Probe Request can seen and we are interested to filter this packets and see how many of them there are, so we can parse the MAC address of the source of the packet which it turn each MAC address is uniqe so the number of devices can be counted.

3.1 Scapy library

One of most powerfull libraries in this regard is Scapy which easily can handle all of your needs in computer networks. We take advantage of this library by importing it into our Python code to filter Probe Requests, MAC addresses and singnal strengh of devices.

The instalation method of Scapy can be find on next pages but, for now we need to find the function that we need to parse information from packets incideing out wireless adaptor. By running Scapy from command line using `ubuntu@ubuntu:~$ scapy`, you will enter the Scapy CLI, after that you may type `ls()` to have almost all of the functions of Scapy.

```
ubuntu@ubuntu:~$ scapy
INFO: Can't import matplotlib. Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
INFO: Can't import python-cryptography v1.7+. Disabled WEP
decryption/encryption. (Dot11)
INFO: Can't import python-cryptography v1.7+. Disabled IPsec
encryption/authentication.
WARNING: IPython not available. Using standard Python shell instead.
AutoCompletion, History are disabled.
```

```
                     aSPY//YASa
             apyyyyCY//////////YCa       |
            sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.4.3.dev653
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | What is dead may never die!
       scccccp///pSP///p          p//Y   |                      -- Python 2
      sY/////////y  caa           S//P   |
       cayCyayP//Ya              pY/Ya
        sY/PsY////YCc           aC//Yp
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs

>>> ls()
```

In our case we are looking for functions related to IEEE 802.11. these functions are listed under the name `Dot11xxx` which xxx is the part you may look for.

```
Dot11      : 802.11
Dot11ATIM  : 802.11 ATIM
Dot11Ack   : 802.11 Ack packet
Dot11AssoReq : 802.11 Association Request
Dot11AssoResp : 802.11 Association Response
Dot11Auth  : 802.11 Authentication
Dot11Beacon : 802.11 Beacon
Dot11CCMP  : 802.11 TKIP packet
Dot11Deauth : 802.11 Deauthentication
Dot11Disas : 802.11 Disassociation
Dot11Elt   : 802.11 Information Element
Dot11EltCountry : 802.11 Country
Dot11EltCountryConstraintTriplet : 802.11 Country Constraint Triplet
Dot11EltMicrosoftWPA : 802.11 Microsoft WPA
Dot11EltRSN : 802.11 RSN information
Dot11EltRates : 802.11 Rates
Dot11EltVendorSpecific : 802.11 Vendor Specific
Dot11Encrypted : 802.11 Encrypted (unknown algorithm)
Dot11FCS   : 802.11-FCS
Dot11ProbeReq : 802.11 Probe Request
Dot11ProbeResp : 802.11 Probe Response
Dot11QoS   : 802.11 QoS
Dot11ReassoReq : 802.11 Reassociation Request
Dot11ReassoResp : 802.11 Reassociation Response
Dot11TKIP  : 802.11 TKIP packet
Dot11WEP   : 802.11 WEP packet
```

At the end of the list we can find `Dot11ProbeReq` function that we need to work with. We use this function to filter the packets of Probe Request.

The packets can be filtered using a simple code as follow,

```
if pkt.haslayer(Dot11ProbeReq):
```

Then we use addr2 function to parse the MAC address, signal strength and SSID of the packet using the following code,

```
MAC_Addr = pkt.addr2
dBm = pkt.dBm_AntSignal
SSID = pkt.info
```

It is usual that a same device broadcast multiple Probe Request with different SSIDs, to avoid counting same device over and over again, the mac address is saved into a List and new MAC addresses is checked whether is in on the list or not. By doing this we can have a unique MAC addresses correspond to a specific device in the vicinity.

To run the code, some requirement should be satisfied by running and installing the following commands,

	`sudo add-apt-repository universe`
Install Python 2.7	`sudo apt install python2.7`
Install git	`sudo apt install git`
Clone Scapy for Python 2.7	`git clone https://github.com/secdev/scapy.git`
Install setuptools	`sudo apt-get install python-setuptools`
	`cd scapy/`
Install cloned repo of Scapy using setuptools	`sudo python2.7 setup.py install`

4 Result

After successful installation, the Python code can be run on the device while its wireless adaptor is on Monitor Mode, the name of the Monitor Mode interface should be as a parameter,

```
ubuntu@ubuntu:/media/ubuntu/Hiwa 2/OneDrive/OneDrive - Politecnico di
Milano/@polimi_private/WI$ sudo python2.7 tprob.py wlo1mon
Ready to monitoring
>>>>>>>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<<<


New User Found  >>> e4:f0:42:17:45:9e  Vodafone-A61171252

=====================> Probes in List <======================
```

```
0 |   Time    |   MAC Adress       |  dBm |    Serached for SSID
1 |  22:30:35 |  6c:ad:f8:6f:f7:5a |  -87 |    FRITZ!Box WLAN 3270
2 |  22:30:37 |  f4:f5:e8:3f:50:46 |  -87 |    Home&Life SuperWiFi-EE09
3 |  22:30:45 |  84:f3:eb:ba:f5:4f |  -79 |    WifiCasaRM
4 |  22:30:50 |  20:df:b9:87:7d:21 |  -84 |    Asimov
5 |  22:30:51 |  ac:5f:3e:d4:a6:e3 |  -68 |    AndroidShare_6420
6 |  22:30:52 |  f4:f5:d8:8e:2d:84 |  -90 |    Vodafone-A42233611
7 |  22:31:07 |  a8:54:b2:0e:09:00 |  -90 |    Alice Gatta
8 |  22:31:12 |  dc:fb:48:a8:a9:d5 |  -92 |    FASTWEB-B5C2AB
9 |  22:31:54 |  e4:f0:42:17:45:9e |  -88 |    Vodafone-A61171252

================> Devices around:9 Device(s) <================


>> Press Ctrl+Z to terminate monitoring
>> Monitoring result will be saved on source folder as a .CSV file.
```

In this run we just considered unique MAC addresses around but, there may be user away from the place we don’t want to count. In this regard we can have another filter on signal strength. Since close devices have a better signal power, the dBm field can be filtered as well.

At the above result, the user number 5 is one of my devices at the living room which is closer to the laptop and consequently it has higher signal power.

Since counting MAC addresses in this way is not precise to have the actual number of present people, we use Machine Learning approaches to have a better estimation on present user in the area.

Sources

[1] Last year projects on Wireless Internet course on : http://www.antlab.polimi.it/ale-teaching/wireless-internet

[2] Lecture slides of Wireless Internet course on: http://www.antlab.polimi.it/ale-teaching/wireless-internet

[3] Tutorials on: https://www.pentesteracademy.com/course?id=14

[4] Scapy documentation on: https://scapy.readthedocs.io/en/latest/