March Challenge

Naavin Ravinthran

Abstract

This document aims to assist you in completing the March Challenge.

1 Introduction

2 The Idea

MAC Addresses are unique.[1] It is not exactly kept secret either, when probing new networks, the address is given out. If we have a large data collection of different places and time, we can target a specific device using its MAC Address and build a profile on it, for example, noting patterns in where the device is at different times of the day or week. Devices won't continuously send this information out, but for the purposes of the challenge let's assume it does.

3 PCAP File Format

This is a binary file format. You can open it up in Wireshark and search manually, but it would be easier to quickly write a script to look through it.

There is a global header found at the very beginning of the file with the following contents.

The magic number is always set to $0xa1b2c3d4$, the major number is 2, the minor number is 4 (as of time of writing), thiszone indicates the timezone (In practice this is always 0 for GMT/UTC), sigfigs indicates the significant figures for the time stamp, which is usually 0, snapen is the "snapshot length",

Name	Type
magic_number	uint32
version_major	uint16
version_minor	uint16
thiszone	int32
sigfigs	uint32
snaplen	uint32
network	uint32

Figure 1: Source: `https://gitlab.com/wireshark/wireshark/-/wikis/Development/LibpcapFileFormat#record-packet-header`

and network indicates the link-layer header type, which will tell you what type it is and how to interpret the data, eg: Ethernet, Bluetooth, etc. A full list is found here: `http://www.tcpdump.org/linktypes.html`.

And for each packet there is a packet (Record) header:-

Name	Type
ts_sec	uint32
ts_usec	uint32
incl_len	uint32
orig_len	uint32

Figure 2: Source: `https://gitlab.com/wireshark/wireshark/-/wikis/Development/LibpcapFileFormat#record-packet-header`

ts_sec is particularly useful to us, because it gives us the unix epoch time of the packet arrival, which gives us the date and time it was captured. One would simply need to convert it to a more human-readable format.

ts_usec is the microseconds when this packet was captured (as an offset to the second).

incl_len is the number of bytes in the packet data

[1] I may be slightly misleading here, on some devices, MAC Randomisation is a thing, where the MAC Addresses are randomised when it probes a new network. Android 10 and above does this, `https://source.android.com/devices/tech/connect/wifi-mac-randomization` but Windows 10 does not `https://support.microsoft.com/en-us/windows/how-to-use-random-hardware-addresses-ac58de34-35fc-31ff-c650-823fc48eb1bc`.

in the file. Don't worry about orig_len for now.

Once you have that, the data after that would depend on the *l*ink-layer header (See above.). Usually this means that you'll have to comb through PDFs of Core specifications of different protocols. But as a tip, you can open in an existing packet capturing software like Wireshark, and open up the pcap in a hex editor and compare. [2]. A hex editor simply opens a binary file and shows in hex representation what each byte is. Using this method, you can find where a MAC Address [3] is located in the packet data, and use that to parse it.

4 Analysis

Basically, what you want to do is find any abnormalities. One way of doing this would be by grouping the MAC Addresses together (across all the areas), and seeing the time they are at a particular area. (?, ?)

Perhaps you can construct a table like this:-

Area	Date/Time
1	2020-01-01 11:15am
2	2020-01-01 5:00pm
1	2020-01-02 11:16am
3	2020-01-02 2:18pm
4	2020-01-02 2:19pm
...	...

Figure 3: Behaviour of a specific Mac Address X

Across the different files see if there are any points that don't match up. For example, if somebody was in Area 3 at 2:18pm, and Area 4 in 2:19pm, and it is known that it would be unfeasible for a regular person to go from Area 3 to Area 4 in a minute as they are too far away, we would know that would be the speedster. [4]

After finding the abnormality, the MAC Address doesn't tell us their identity. We can however, use it to detect patterns in their behaviour, where they go at what times. Once we have that, it should be easy to just go to the area and observe who that person is (e.g: If you see the MAC Address also regularly visits Area Z on Thursdays at 3pm, go there at that time on multiple weeks and find the person who is always there). In the figure above, perhaps you can surmise that the person goes to Area 1 every morning at roughly 11:15am.

[2]As for hex editors, For linux-based distros, I like Okteta. For Windows, HxD is good

[3]In Bluetooth this is called BD_ADDR

[4]I suppose it is possible that there are multiple devices spoofing the MAC of other devices, but why would anyone do that?

4.1 Unix Epoch Time

A date was chosen, 00:00:00 UTC on 1 Jan 1970, and it was decided that a format would be created where a 32-bit integer would represent the number of seconds *a*fter that time, and this integer would be used for storing the date and time (It's a signed integer, so yes it includes negative numbers and dates/times prior to 1970 works). As a side-note, this would unfortunately only work until Tuesday 2038-01-19, because after that, the 32-bit integer would overflow. Some protocols have already migrated to other standards of representing points of time, but the pcap file hasn't [5]

Because of difficult-to-account-for factors like leap-years, change in timezones, daylight savings time, etc. I'd recommend either using a library for conversion to a human readable format, or simply using maths to get the time duration between different time points, bypassing the need to get the date at all. (e.g: instead of finding 25 Feb 2019 12:00am and 25 Feb 2019 12:05am and finding the difference, simply subtract the later date with the earlier date to get 300 seconds, which you can easily convert to 6 minutes)

References

Cunche, M. (2013, 11). I know your mac address: Targeted tracking of individual using wi-fi. *Journal of Computer Virology and Hacking Techniques*, *10*, 219–227. doi: 10.1007/s11416-013-0196-1

Wikipedia. (2021). *Unix time — Wikipedia, the free encyclopedia.* `http://en.wikipedia.org/w/index.php?title=Unix%20time&oldid=1010057781`. ([Online; accessed 05-March-2021])

[5]Technically more modern version variants of pcap like pcap-ng has.