ProbeQuest Documentation

Release 0.7.2

Paul-Emmanuel Raoul

Sep 25, 2020

TABLE OF CONTENTS

ProbeQuest is a toolkit allowing to sniff and display the Wi-Fi probe requests passing nearby your wireless interface.

This project has been inspired by this paper.

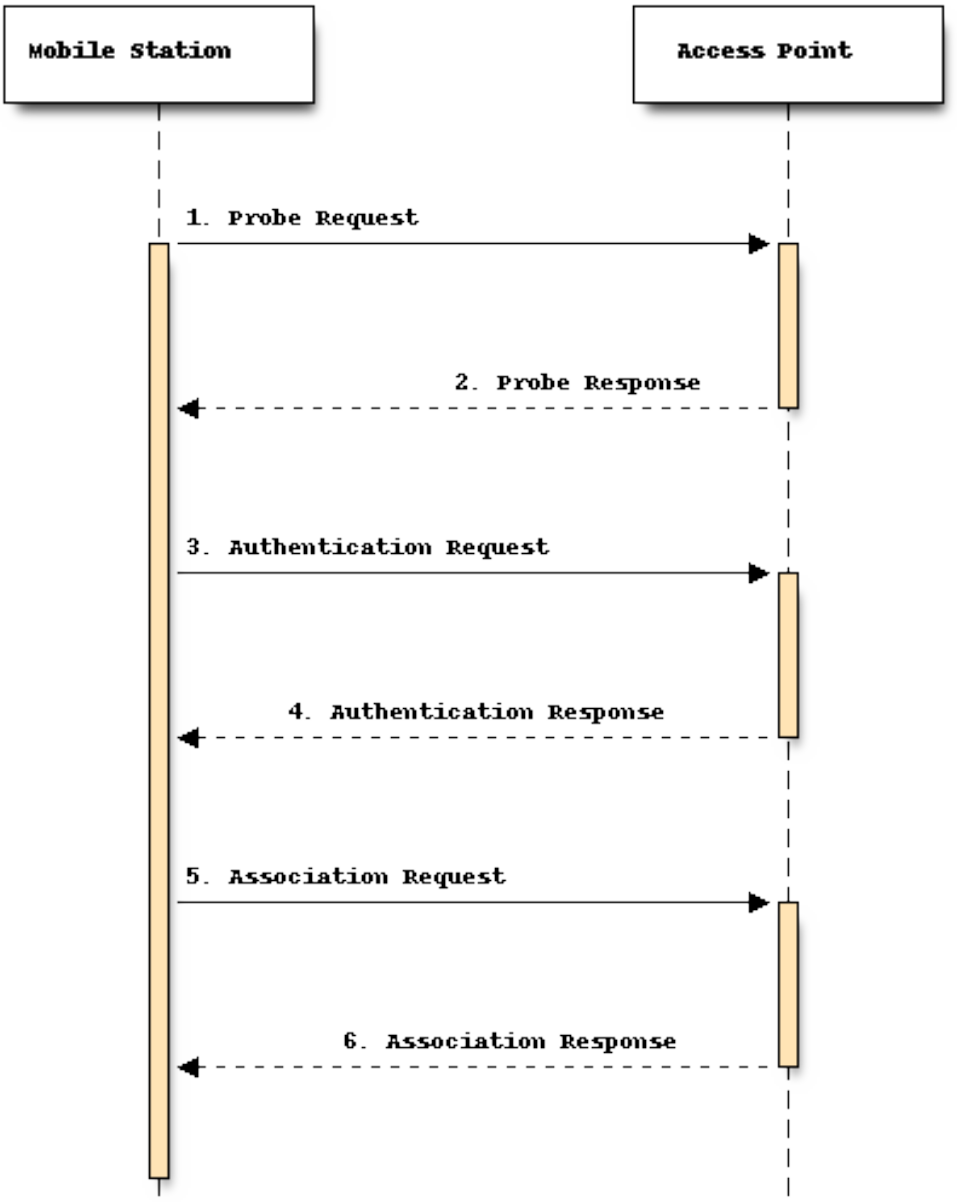
CHAPTER

ONE

WHAT ARE WI-FI PROBE REQUESTS?

Probe requests are sent by a station to elicit information about access points, in particular to determine if an access point is present or not in the nearby environment. Some devices (mostly smartphones and tablets) use these requests to determine if one of the networks they have previously been connected to is in range, leaking their preferred network list (PNL) and, therefore, your personal information.

Below is a typical Wi-Fi authentication process between a mobile station (for example, your smartphone) and an access point (AP):



Step 1 is optional (and therefore, step 2) since the access points announce their presence by broadcasting their name (ESSID) using beacon frames. Consequently, it is not necessary to rely on probe requests to get the list of the access points available. It is a design choice that, although it speeds up the discovery process, causes privacy and security issues.

ProbeQuest can be used to leverage this leak of information to conduct diverse social engineering and network attacks.

CHAPTER

TWO

INSTALLATION

2.1 Using pip (recommended)

```
sudo pip3 install --upgrade probequest
```

2.2 From sources

```
git clone https://github.com/SkypLabs/probequest.git
cd probequest
sudo pip3 install --upgrade .
```

CHAPTER

THREE

USAGE

3.1 Enabling the monitor mode

To be able to sniff the probe requests, your Wi-Fi network interface must be set to monitor mode.

3.1.1 With *ifconfig* and *iwconfig*

```
sudo ifconfig <wireless interface> down
sudo iwconfig <wireless interface> mode monitor
sudo ifconfig <wireless interface> up
```

For example:

```
sudo ifconfig wlan0 down
sudo iwconfig wlan0 mode monitor
sudo ifconfig wlan0 up
```

3.1.2 With *airmon-ng* from aircrack-ng

To kill all the interfering processes:

```
sudo airmon-ng check kill
```

To enable the monitor mode:

```
sudo airmon-ng start <wireless interface>
```

For example:

```
sudo airmon-ng start wlan0
```

3.2 Command line arguments

Toolkit for Playing with Wi-Fi Probe Requests

```
usage: probequest [-h] [--debug] [--fake] -i INTERFACE [--ignore-case]
                  [--mode {raw,pnl}] [-o OUTPUT_FILE] [--version]
                  [-e ESSID_FILTERS [ESSID_FILTERS ...] | -r ESSID_REGEX]
                  [--exclude MAC_EXCLUSIONS [MAC_EXCLUSIONS ...] | -s
                  MAC_FILTERS [MAC_FILTERS ...]]
```

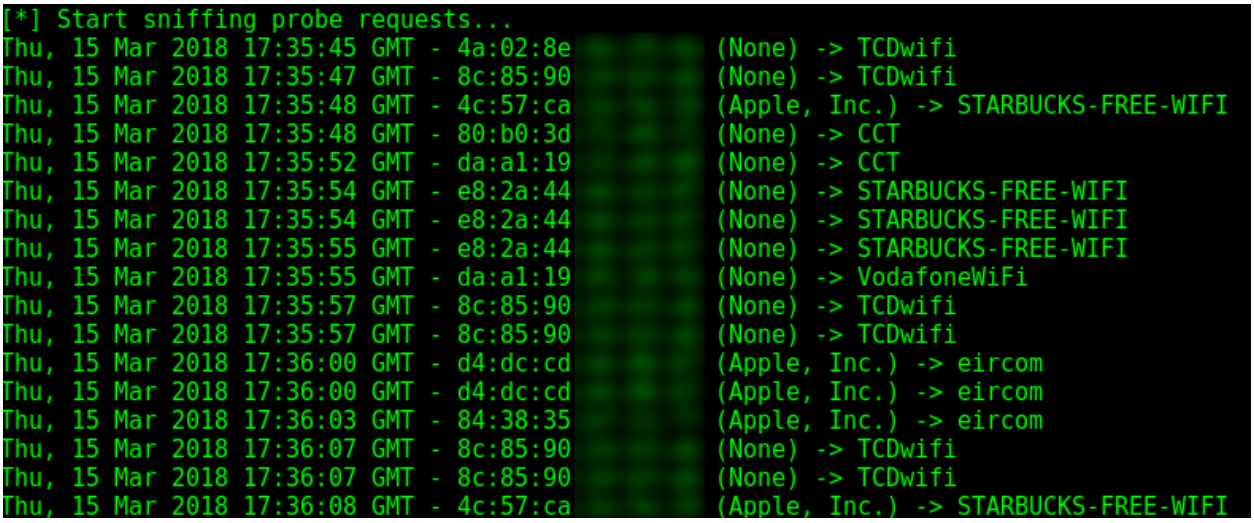
3.2.1 Named Arguments

--debug	debug mode
	Default: False
--fake	display only fake ESSIDs
	Default: False
-i, --interface	wireless interface to use (must be in monitor mode)
--ignore-case	ignore case distinctions in the regex pattern (default: false)
	Default: False
--mode	Possible choices: raw, pnl
	set the mode to use
	Default: raw
-o, --output	output file to save the captured data (CSV format)
--version	show program's version number and exit
-e, --essid	ESSID of the APs to filter (space-separated list)
-r, --regex	regex to filter the ESSIDs
--exclude	MAC addresses of the stations to exclude (space-separated list)
-s, --station	MAC addresses of the stations to filter (space-separated list)

3.2.2 Example of use

```
sudo probequest -i wlan0
```

Here is a sample output:


[*] Start sniffing probe requests...
Thu, 15 Mar 2018 17:35:45 GMT - 4a:02:8e (None) -> TCDwifi
Thu, 15 Mar 2018 17:35:47 GMT - 8c:85:90 (None) -> TCDwifi
Thu, 15 Mar 2018 17:35:48 GMT - 4c:57:ca (Apple, Inc.) -> STARBUCKS-FREE-WIFI
Thu, 15 Mar 2018 17:35:48 GMT - 80:b0:3d (None) -> CCT
Thu, 15 Mar 2018 17:35:52 GMT - da:a1:19 (None) -> CCT
Thu, 15 Mar 2018 17:35:54 GMT - e8:2a:44 (None) -> STARBUCKS-FREE-WIFI
Thu, 15 Mar 2018 17:35:54 GMT - e8:2a:44 (None) -> STARBUCKS-FREE-WIFI
Thu, 15 Mar 2018 17:35:55 GMT - e8:2a:44 (None) -> STARBUCKS-FREE-WIFI
Thu, 15 Mar 2018 17:35:55 GMT - da:a1:19 (None) -> VodafoneWiFi
Thu, 15 Mar 2018 17:35:57 GMT - 8c:85:90 (None) -> TCDwifi
Thu, 15 Mar 2018 17:35:57 GMT - 8c:85:90 (None) -> TCDwifi
Thu, 15 Mar 2018 17:36:00 GMT - d4:dc:cd (Apple, Inc.) -> eircom
Thu, 15 Mar 2018 17:36:00 GMT - d4:dc:cd (Apple, Inc.) -> eircom
Thu, 15 Mar 2018 17:36:03 GMT - 84:38:35 (Apple, Inc.) -> eircom
Thu, 15 Mar 2018 17:36:07 GMT - 8c:85:90 (None) -> TCDwifi
Thu, 15 Mar 2018 17:36:07 GMT - 8c:85:90 (None) -> TCDwifi
Thu, 15 Mar 2018 17:36:08 GMT - 4c:57:ca (Apple, Inc.) -> STARBUCKS-FREE-WIFI

CHAPTER

FOUR

USE CASE

Let's consider the following simple scenario inspired from a real data collection (the data have been anonymised): a device tries to connect to *John's iPhone*, *CompanyX_staff*, *STARBUCKS-FREE-WIFI* and *VM21ECAB2*. Based on this information, several assumptions can be made:

- The device owner's name is John.
- The device is set in English and its owner speaks this language (otherwise it would have been *iPhone de John* in French, *iPhone von John* in German, etc).
- The device should be a laptop trying to connect to an iPhone in hotspot mode. The owner has consequently at least two devices and is nomad.
- The owner works for CompanyX.
- The owner frequents coffee shops, in particular StarBucks.
- The owner is used to connecting to open Wi-Fi access points.
- *VM21ECAB2* seems to be a home access point and is the only one in the device's PNL. It is likely the owner's place and, consequently, the device's owner is a customer of Virgin Media.

As you can see, the amount of data inferred from these four probe requests is already impressive, but we can go further. Relying on a database of Wi-Fi access points' location, such as WIGLE.net, it becomes possible to determine the places the device's owner has previously been to. VM21ECAB2 should be a unique name, easily localisable on a map. Same for CompanyX_staff. If this last one is not unique (because CompanyX has several offices), crossing the data we have can help us in our investigation. For example, if CompanyX is present in several countries, we can assume that the device's owner lives in a country where both CompanyX and Virgin Media are present. Once we have determined which office it is, we can suppose that the device's owner is used to stopping in StarBucks located on their way from home to their office.

Profiling a person is the first step to conduct a social engineering attack. The more we know about our target, the better chance the attack has to succeed. Also, because we know which Wi-Fi access points our target's devices will try to connect to, an evil twin attack is conceivable.

CHAPTER

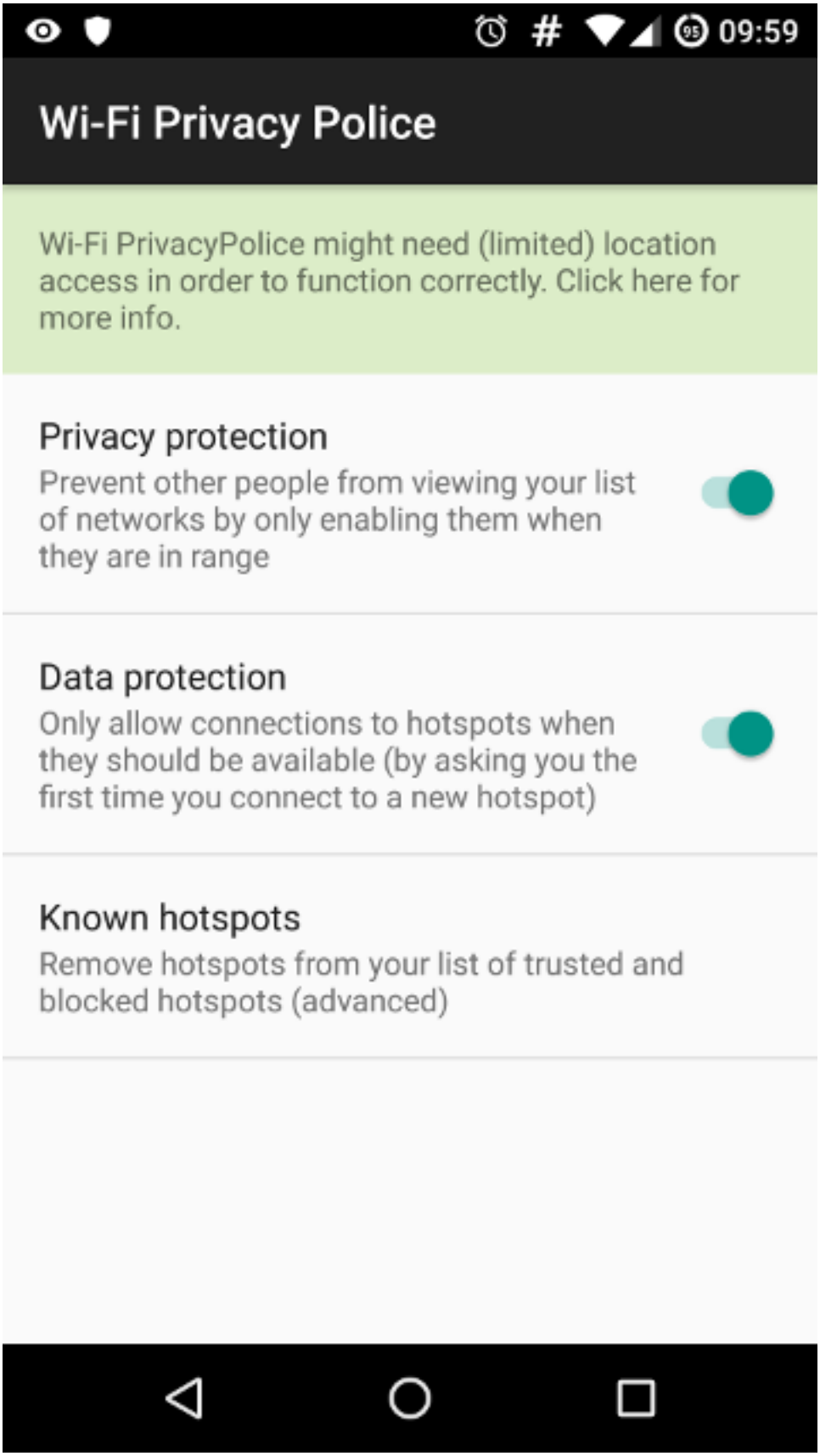
FIVE

MITIGATION

As far as I know, there are two mitigation techniques:

- Don't use probe requests at all. It is by far the most efficient way not to leak any piece of information. As said earlier, it is not necessary to rely on probe requests to get the list of the nearby access points since they broadcast their name by themselves.
- Randomise the source MAC address of each probe request sent. This way, it's no longer possible for a third party to link probe requests to a specific device based on the Wi-Fi data collected. However, using a Software-Defined Radio to capture RF metadata such as the frequency offset, it would be possible to fingerprint each Wi-Fi packet and so each Wi-Fi device, regardless of their source MAC address (this technique will be implemented in ProbeQuest).

In practice, you can install Wi-Fi Privacy Police from F-Droid or the Play Store to prevent your Android devices from leaking their PNL.


09:59
Wi-Fi Privacy Police
Wi-Fi PrivacyPolice might need (limited) location access in order to function correctly. Click here for more info.
Privacy protection
Prevent other people from viewing your list of networks by only enabling them when they are in range
Data protection
Only allow connections to hotspots when they should be available (by asking you the first time you connect to a new hotspot)
Known hotspots
Remove hotspots from your list of trusted and blocked hotspots (advanced)

Once installed, the **Privacy protection** option should be switched on.

On iOS, the source MAC address is randomised since iOS 8.

CHAPTER

SIX

MODULES

6.1 ProbeQuest Configuration

ProbeQuest configuration.

class probequest.config.**Config**
Configuration object.

complile_essid_regex()
Returns the compiled version of the ESSID regex.

property display_func
Callback function triggered when a packet needs to be displayed.

generate_frame_filter()
Generates and returns the frame filter according to the different options set of the current 'Config' object.

property storage_func
Callback function triggered when a packet needs to be stored.

class probequest.config.**Mode**(*value*)
Enumeration of the different operational modes supported by this software.

6.2 Fake Packet Sniffer

Fake packet sniffer module.

class probequest.fake_packet_sniffer.**FakePacketSniffer**(*config*, *new_packets*)
A fake packet sniffing thread.

This thread returns fake Wi-Fi ESSIDs for development and test purposes.

join(*timeout=None*)
Stops the fake packet sniffer.

new_packet()
Adds a new fake packet to the queue to be processed.

run()
Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

stop()
Stops the fake packet sniffer.

Alias for 'join()'.

6.3 Packet Sniffer

Packet sniffer module.

class probequest.packet_sniffer.**PacketSniffer**(*config*, *new_packets*)
Wrapper around the 'AsyncSniffer' class from the Scapy project.

is_running()
Returns true if the sniffer is running, false otherwise.

new_packet(*packet*)
Adds the packet given as parameter to the queue to be processed by the parser.

start()
Starts the packet sniffer.

stop()
Stops the packet sniffer.

6.4 PNL Viewer

6.5 Probe Request

A Wi-Fi probe request.

class probequest.probe_request.**ProbeRequest**(*timestamp*, *s_mac*, *essid*)
Probe request class.

get_mac_organisation()
Returns the OUI of the MAC address as a string.

6.6 Probe Request Sniffer

Wi-Fi probe request sniffer.

class probequest.probe_request_sniffer.**ProbeRequestSniffer**(*config*)
Wi-Fi probe request sniffer.

It is composed of a packet sniffer and a packet parser, both running in a thread and intercommunicating using a queue.

is_running()
Returns true if the probe request sniffer is running and false otherwise.

new_parser()
Creates a new parsing thread.

new_sniffer()
Creates a new sniffing thread.

If the '–fake' option is set, a fake packet sniffer will be used.

start()
Starts the probe request sniffer.

This method will start the sniffing and parsing threads.

stop()
Stops the probe request sniffer.

This method will stop the sniffing and parsing threads.

6.7 Raw Probe Request Viewer

Raw probe request viewer.

class probequest.ui.raw.**RawProbeRequestViewer**(*config*)
Displays the raw probe requests passing nearby the Wi-Fi interface.

start()
Starts the probe request sniffer.

stop()
Stops the probe request sniffer.

CHAPTER

SEVEN

DEVELOPMENT

7.1 Running the unit tests

To run the unit tests:

```
python3 setup.py test
```

7.2 Releasing a new version

Below are the different steps to do before releasing a new version:

- Run all tests and be sure they all pass
- Update the *VERSION* variable in *probequest/version.py*
- Update the requirements in *setup.py* if needed
- Update the package's metadata (description, classifiers, etc) in *setup.py* if needed
- Update *README.rst* if needed
- Update the documentation if needed and make sure it compiles well (*cd ./docs && make html*)
- Update the copyright year in *docs/conf.py* if needed
- Add the corresponding release note to *CHANGELOG.md*

After having pushed the new release:

- Edit the release note on GitHub

CHAPTER

EIGHT

SECURITY POLICY

8.1 Reporting a Vulnerability

If you have found a security issue in ProbeQuest, please disclose it responsibly by emailing me at *skyper(at)skyplabs[dot]net*. My PGP public key can be found on my Keybase profile:

To facilitate the encryption process, you can use this online tool. You can also use it to verify my signatures.

PYTHON MODULE INDEX

p

INDEX