UNVEIL: Capture and Visualise WiFi Data Leakages

Shubham Jain
Imperial College London
sjain@imperial.ac.uk

Eden Bensaid
Massachussets Institute of Technology
edenbd@mit.edu

Yves-Alexandre de Montjoye
Imperial College London
demontjoye@imperial.ac.uk

ABSTRACT

In the past few years, numerous privacy vulnerabilities have been discovered in the WiFi standards and their implementations for mobile devices. These vulnerabilities allow an attacker to collect large amounts of data on the device user, which could be used to infer sensitive information such as religion, gender, and sexual orientation. Solutions for these vulnerabilities are often hard to design and typically require many years to be widely adopted, leaving many devices at risk.

In this paper, we present UNVEIL – an interactive and extendable platform to demonstrate the consequences of these attacks. The platform performs passive and active attacks on smartphones to collect and analyze data leaked through WiFi and communicate the analysis results to users through simple and interactive visualizations.

The platform currently performs two attacks. First, it captures probe requests sent by nearby devices and combines them with public WiFi location databases to generate a map of locations previously visited by the device users. Second, it creates rogue access points with SSIDs of popular public WiFis (e.g. _Heathrow WiFi, Railways WiFi) and records the resulting internet traffic. This data is then analyzed and presented in a format that highlights the privacy leakage. The platform has been designed to be easily extendable to include more attacks and to be easily deployable in public spaces. We hope that UNVEIL will help raise public awareness of privacy risks of WiFi networks.

CCS CONCEPTS

• **Security and privacy** → **Social aspects of security and privacy**; • **Human-centered computing** → *Information visualization.*

KEYWORDS

WiFi Security, Smartphones, Visualisation, Public Demonstration

ACM Reference Format:
Shubham Jain, Eden Bensaid, and Yves-Alexandre de Montjoye. 2019. UNVEIL: Capture and Visualise WiFi Data Leakages. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3308558.3314143

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3314143

1 INTRODUCTION

Mobile devices like smartphones and tablets have become extremely popular over the past 10 years. Market research company Newzoo [11] reported that more than 3.3 billion smartphones and 230 million tablets were in use at the end of last year. WiFi is a core component of mobile devices, enabling them to connect to the internet. However, several vulnerabilities and design flaws have been discovered in WiFi protocols and their implementations over the years. These flaws can be exploited to track individuals [16] and to obtain sensitive information, such as religion, gender, and sexual orientation about them [1, 12].

Multiple experiments have been designed to illustrate the data leakage in WiFi networks. However, all previous approaches demonstrate specific vulnerabilities or require a user to install an application. Projects such as Wombat [8], Probr [15], and the Digital Marauder's Map [5] have demonstrated how WiFi can be used to track people's movements in real time. The Haystack Project [14] introduced an Android app to analyze mobile traffic and help individuals identify privacy leaks on their device. MITMProxy [3] is an open source and interactive HTTPS proxy that can be used to reverse-engineer installed applications and detect if any personal information is being leaked by the apps [2].

Our platform, UNVEIL, aims to raise awareness of the privacy risks of WiFi vulnerabilities through an interactive demonstration. The data is collected from nearby mobile devices (unless the owner opts out), analyzed, and the results can be viewed on installed screens. The platform modularity ensures that it can be deployed easily in public spaces like museums or train stations, making it accessible for general public.

UNVEIL currently supports two attacks:

- Generation of a map of locations visited by nearby mobile devices, using probe requests broadcasted by them;
- Profiling of users connected to our rogue access points, using the internet traffic generated by the device.

Neither attack requires users to download an app or perform any action. Moreover, the platform can be easily extended to include new attacks.

2 DATA

In this section, we describe in detail the types of data collected by UNVEIL.

2.1 Probe Requests

Probe requests are sent by WiFi-enabled devices to discover the surrounding available WiFi access points (AP). Discovering the network by scanning all possible channels and listening to beacons is not considered to be very efficient (*passive scanning*). To enhance

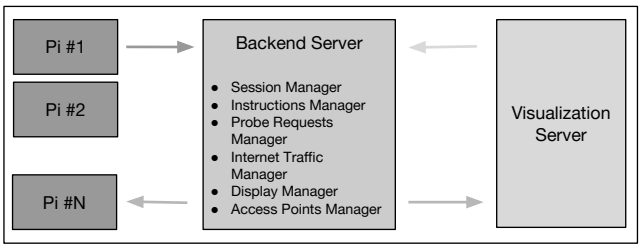

Figure 1: System architecture. UNVEIL has 3 components: Raspberry Pis, Backend Server, and Visualization Server.

this discovery process, devices often use what is called *active scanning*. In active scanning, a device still goes through each channel, but instead of passively listening to the signals on that frequency, they broadcast *probe requests* asking which known networks are available on that channel and receive responses from the APs. These probe requests contain the (possibly randomized) MAC address of the device along with the name of the AP (SSID) being probed. The device keeps a list of SSIDs of previously connected APs. We query publicly available WiFi Location databases to associate SSIDs with geographic coordinates [17, 18].

Some devices use randomized MAC addresses to prevent leaking location history through probe requests [9]. However, recent studies have shown that MAC address randomization policies are neither universally implemented nor necessarily addressing privacy concerns [7]. For instance, timing attacks such as the one described in [10] can be used to group together the probe requests (using different randomized MAC addresses) coming from the same device.

2.2 Internet Traffic

Typically, mobile devices are set by default to automatically connect to known WiFi SSIDs, without any interaction required from the user. UNVEIL creates rogue WiFi access points using popular public network SSIDs that many devices are likely to connect to. Internet traffic generated by any device connected to UNVEIL's access point is sniffed and recorded. The DNS and HTTP requests are analyzed to profile the user. The DNS requests made by the device contain the websites visited by the user and the applications they use. These might contain information allowing an attacker to infer sensitive user attributes such as gender, age group, sexual orientation, religion, mental health, race, and nationality. The HTTP requests, furthermore include the user agent, which often contains details regarding the device manufacturer, model, and version of the OS (see Fig. 3).

3 SYSTEM ARCHITECTURE

The UNVEIL platform is designed to be modular and easily extendable. It is structured into three main components: Raspberry Pis, a Backend Server, and a Visualization Server. Figure 1 shows a high-level representation of the architecture. Raspberry Pis are responsible for collecting the data. The Backend Server manages the Pis and analyzes the data. The Visualization Server runs the frontend component that retrieves the analysis results from the Backend Server and showcases them. The Visualization Server also provides the controller screen for managing the overall system. The communication between all the components takes place via the REST APIs.

3.1 Raspberry Pi

Raspberry Pis are responsible for collecting the WiFi data and for sending it to the Backend Server for analysis. The inbuilt WiFi antenna in the Pi is used to provide internet connectivity for the Pi itself, while an external USB antenna is used to collect the probe requests and make the rogue access point available.

Each Pi runs a Pi-Controller process that starts during the boot. This process runs two subprocesses in parallel. One subprocess checks and updates the state of the Pi every 10 seconds. State is represented by the mode, SSID and channel the external USB antenna is operating in. The other subprocess periodically requests the Backend Server for the instructions to be executed on the Pi and sends back acknowledgement on successful execution. The Pi-Controller maintains an extendable instruction set of operations that can be successfully executed on the Pi.

The Pi can operate external USB antenna in two modes: listening to probe requests or creating an access point depending on the instruction fetched from the Backend Server:

- *ProbeReq* - This mode is used while listening for probe requests. The operating channel changes periodically (every 5 seconds) to capture probe requests across all the commonly used channels (1, 6 and 11). SSID for the state is set to null in this mode.
- *AccessPoint* - In this mode, Pi creates an access point with the SSID and channel received from the Backend Server. It sniffs and collects all the traffic passing through, along with the MAC address of the device.

The Pi instruction set currently supports 5 operations:

- *Start ProbeReq.* Start the probe request collection.
- *Stop ProbeReq.* Stop the probe request collection.
- *Start AccessPoint.* Get an SSID and channel number from the Backend Server and create an access point.
- *Stop AccessPoint.* Stop the access point.
- *Stop All.* Stop the data collection process, if any.

Whenever any *Stop* instruction is executed, the collected data is sent to the Backend Server for analysis and is deleted from the Pi.

We use the Raspberry Pi 3B+[1] with external antenna USB WiFi adapter Alfa Network AWUS036NHA[2] and Kali Linux as the OS. The Pi-Controller provides various modules to control WiFi antennas. It is implemented in Python 3.6 and is designed to work on any device running Kali Linux with two wifi antennas. We use hostapd[3] and dnsmasq[4] to create access points and Wireshark to collect the data.

[1] https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
[2] https://wikidevi.com/wiki/ALFA_Network_AWUS036NHA
[3] https://w1.fi/hostapd/
[4] http://www.thekelleys.org.uk/dnsmasq/doc.html

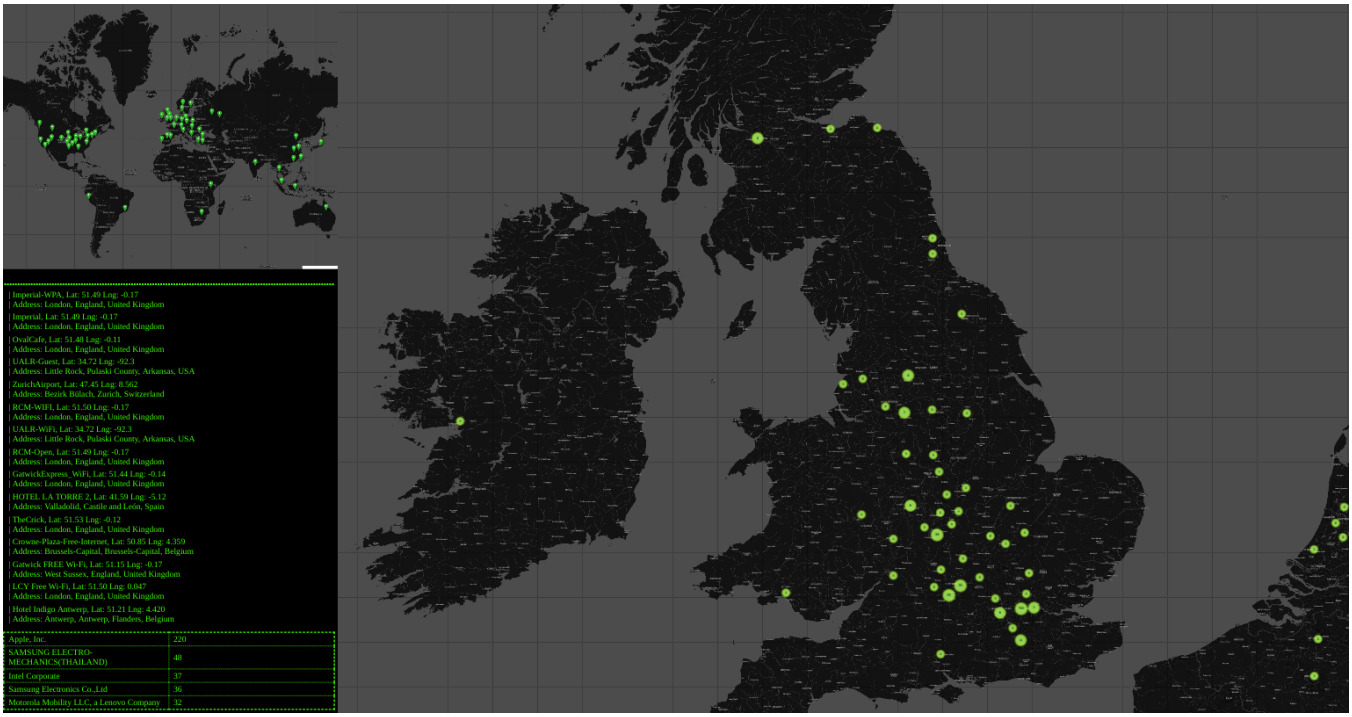

Figure 2: Visualization of the probe requests analysis. The top left corner shows the global map with markers denoting the location of all the observed SSIDs. The text in the center left contains geographic addresses of the SSIDs. The table on the bottom left displays the count of devices per manufacturer, with the most frequent ones at the top. The figure on the right contains the map of UK with the estimated count of the number of number of devices in each detected location. For privacy reasons, the numbers shown in this figure are fake. (Maps provided by ©OpenStreetMap [13])

3.2 Backend Server

The Backend Server is responsible for managing the demonstration, components, data, and analyses in the experiment (Fig. 1). Session Manager is responsible for starting, managing, and ending the demonstration session as well as archiving and cleaning the collected data once the session ends. Instruction Manager maintains a queue of instructions for each pi. The first instruction in the queue is sent when requested by the Pi. The queue is then locked for next 30 seconds or until the acknowledgment of successful execution is not received from the Pi. While the queue is locked, no instruction can be added during that interval. On receipt of successful execution, the instruction is removed from the queue.

Probe Requests Manager and Internet Traffic Manager are responsible for analysis of the captured probe requests and the internet traffic data. Probe Requests Manager associates each SSID with its geographic location (if available) and estimates a count of devices which are near the Pi and probing that SSID. Internet Traffic Manager analyzes the data generated by the devices connected to our rogue access points. All the analysis results for each session for each mode are stored in the database for visualization.

Display Manager is responsible for handling the communications with the Visualization Server. It manages the content to be displayed on the web browser based on the presenter's selections made through the control screen. The Backend Server is implemented in Python 3.6 using Django [4] and MongoDB [6].

3.3 Visualization Server

The Visualization Server is responsible for serving the web pages that visualize the results of data analyses and allow to control the demonstration. The control screen provides controls to start and stop the demonstration, display the live status of the Pis, and select which data to show on the visualization screens. The selected data is retrieved from the Backend Server for visualization.

Figure 2 illustrates the result of the probe requests analysis. The analysis shows the location of the captured SSIDs with the estimated count of the devices associated with those SSIDs.

Figure 3 shows the profile created for a connected user. Each user data is composed of 3 sections: Device Details, DNS Queries, and Internet Traffic. Device Details section displays the MAC address, manufacturer, model number, and OS of the device. DNS queries are used to identify which websites the user visited and which apps are running (including those in background). We filter out the common DNS queries (e.g. Whatsapp, Facebook, Google) to display user-specific ones. The Internet Traffic section is further separated into HTTP and HTTPS requests. HTTP requests reveal the complete URL of the request, which is shown in the visualization.

The control screen can be accessed on a browser, preferably on a laptop or a tablet. It provides buttons to start and stop the demonstration, select data to be visualized, and zoom in and out of the map illustrating the probe requests analysis.

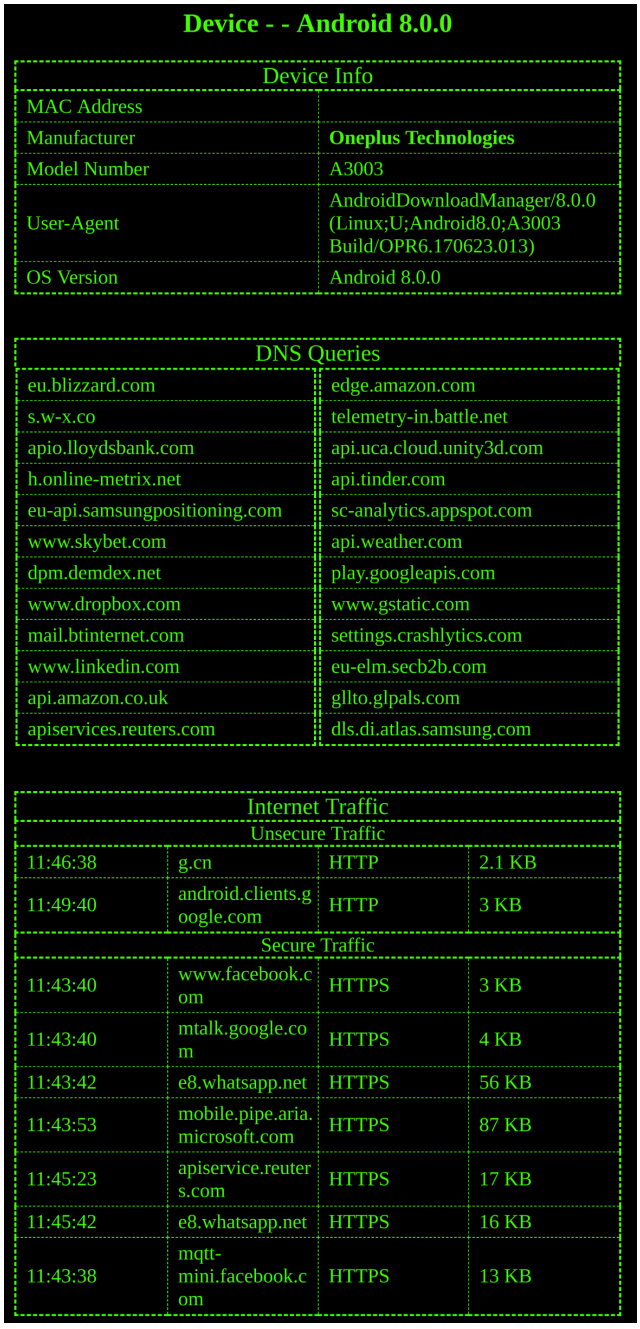

Figure 3: Visualization of the connected users. The top section illustrates the retrieved mobile device details. The center section shows relevant DNS queries generated from the user. The bottom section shows the details of the internet traffic generated by the user. For privacy reasons, the data shown in this figure is fake.

The Visualization Server has been developed using ReactJS[5], Leaflet.js[6], and OpenStreetMap [13].

[5]https://reactjs.org/
[6]https://leafletjs.com/

3.4 Extending the Platform

The platform can be easily extended to demonstrate other data leakages in WiFi networks. The Pi instruction set can be appended with new instructions. These can be designed using the existing modules in Pi-Controller. Backend Server can be extended by adding new analysis managers which can use the available infrastructure of the database, session and instruction management, helping streamline the development and deployment process. A new web page can be added to the Visualization Server to show the analysis results and buttons can be added to the control screen for interacting with the display.

4 DEMONSTRATION

We demonstrate the platform by collecting data in real time and visualizing it. Mobile device users can interact with the geographic data using the buttons on the control screen, as well as try to infer characteristics of the profiled users from the collected data.

A session runs with only one Raspberry Pi collecting probe requests while other Pis operate as rogue access points posing as common UK public WiFis. We provide an option for the user to opt out by entering the MAC Address of their device on our website. Instructions to opt out are clearly publicized with posters around the demonstration space.

A live demonstration requires deploying Raspberry Pis around the space and at least two screens to view the results. Screens should be large enough (preferably 4K) for the audience to view the data. Control screen will be available on the tablet for the audience to zoom into the probe requests map, and see if the data from their smartphones is being captured. The users can also see if their phone has connected to our access points and what data has been collected. All the data (raw and results of analyses) are deleted at the end of each demonstration session.

The demonstration can be deployed in any public or closed space and only requires (wireless) internet connection, electricity supply for the Raspberry Pis, and two screens for visualization. Through this demonstration, we hope to show how sensitive individual-level data can be collected and processed with only £75 worth of hardware.

The platform will be open sourced before the demonstration, along with instructions to deploy and extend it. It will be made available at https://cpg.doc.ic.ac.uk/blog/unveil.

5 CONCLUSION

We have designed and developed a platform to demonstrate privacy attacks on mobile device WiFi. The platform can be easily deployed in public spaces to help raise awareness of the privacy risks in WiFi networks.

6 ACKNOWLEDGEMENTS

The authors would like to thank Imperial College London's Data Science Institute and the Computational Privacy Group, especially Axel Oehmichen, Luc Rocher, Ali Farzanehfar, and Andrea Gadotti for their support. We acknowledge support from the French Development Agency (AFD). The opinions expressed in this article are the authors' own and do not reflect the view of the AFD.

REFERENCES

[1] John S Atkinson, John E Mitchell, Miguel Rio, and George Matich. 2018. Your WiFi is leaking: What do your mobile apps gossip about you? *Future Generation Computer Systems* 80 (2018), 546–557.

[2] Timothy A Chadza, Francisco J Aparicio-Navarro, Konstantinos G Kyriakopoulos, and Jonathon Chambers. 2017. A look into the information your smartphone leaks. (2017).

[3] Aldo Cortesi, Maximilian Hils, Thomas Kriechbaumer, and contributors. 2010–. mitmproxy: A free and open source interactive HTTPS proxy. https://mitmproxy.org/ [Version 4.0].

[4] Django Software Foundation. 2018. Django (Version 2.1). Retrieved 2019-01-01 from https://docs.djangoproject.com/

[5] Xinwen Fu, Nan Zhang, Aniket Pingley, Wei Yu, Jie Wang, and Wei Zhao. 2009. The digital Marauder's map: a new threat to location privacy. In *Distributed Computing Systems, 2009. ICDCS'09. 29th IEEE International Conference on*. IEEE, 589–596.

[6] MongoDB Inc. 2018. MongoDB. Retrieved 2019-02-19 from www.mongodb.com/

[7] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C Rye, and Dane Brown. 2017. A study of MAC address randomization in mobile devices and when it fails. *Proceedings on Privacy Enhancing Technologies* 2017, 4 (2017), 365–383.

[8] Célestin Matte and Mathieu Cunche. 2017. Wombat: An experimental Wi-Fi tracking system. In *8e édition de l'Atelier sur la Protection de la Vie Privée (APVP)*.

[9] Célestin Matte and Mathieu Cunche. 2018. *Spread of MAC address randomization studied using locally administered MAC addresses use historic*. Ph.D. Dissertation. Inria Grenoble Rhône-Alpes.

[10] Célestin Matte, Mathieu Cunche, Franck Rousseau, and Mathy Vanhoef. 2016. Defeating MAC address randomization through timing attacks. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 15–20.

[11] Newzoo. 2019. Global Mobile Market Report. Retrieved 2019-01-01 from https://resources.newzoo.com/hubfs/Factsheets/Newzoo_The_Global_Mobile_Market_Report_Fact_Sheet.pdf

[12] Piers O'Hanlon, Ravishankar Borgaonkar, and Lucca Hirschi. 2017. Mobile subscriber WiFi privacy. In *Security and Privacy Workshops (SPW), 2017 IEEE*. IEEE, 169–178.

[13] OpenStreetMap contributors. 2017. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org.

[14] Abbas Razaghpanah, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Christian Kreibich, Phillipa Gill, Mark Allman, and Vern Paxson. 2015. Haystack: In situ mobile traffic analysis in user space. *ArXiv e-prints* (2015).

[15] Joel Scheuner, Genc Mazlami, Dominik Schöni, Sebastian Stephan, Alessandro De Carli, Thomas Bocek, and Burkhard Stiller. 2016. Probr-a generic and passive WiFi tracking system. In *Local Computer Networks (LCN), 2016 IEEE 41st Conference on*. IEEE, 495–502.

[16] Edwin Vattapparamban, Bekir Sait Çiftler, İsmail Güvenç, Kemal Akkaya, and Abdullah Kadri. 2016. Indoor occupancy tracking in smart buildings using passive sniffing of probe requests. In *Communications Workshops (ICC), 2016 IEEE International Conference on*. IEEE, 38–44.

[17] WiFiSpc. 2018. WiFi Space. Retrieved 2019-01-01 from https://wifispc.com

[18] WiGLE. 2019. Wireless Geographic Logging Engine. Retrieved 2019-01-01 from https://wigle.net