JOHANNES KEPLER
UNIVERSITY LINZ

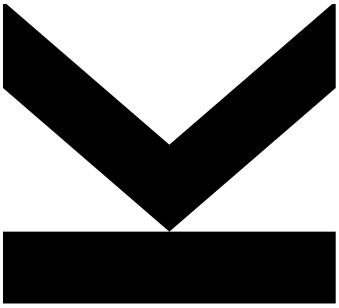
Author
Michael Barth
k01355201

Submission
Institute of
Networks and Security

Thesis Supervisor
Dr. Michael Roland

February 2021

Tracking and position estimation of WLAN clients through passively collected data



Bachelor's Thesis
to confer the academic degree of
Bachelor of Science
in the Bachelor's Program
Informatik

JOHANNES KEPLER
UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Austria
www.jku.at
DVR 0093696

Abstract

This work focuses on methods to capture and analyze data transmitted by Wireless Local Area Network (WLAN) clients in order to track them. This includes evaluation of methods where control of the Access Point (AP) infrastructure is not needed and clients do not need to be connected to a WLAN network. This mainly involves data in probe requests which are transmitted by clients when actively searching for WLAN APs. To evaluate this in a real world scenario a setup consisting of multiple distributed capture devices and a central analysis system is introduced. The captured data is analyzed to verify theoretical concepts. There is still a big part of WLAN client devices that leak lists of stored SSID values when actively scanning for WLAN networks. MAC address randomization helps to protect privacy if enabled. User identities for EAP authentication however are still leaked in default configuration by all major operating systems. Finally some extension ideas and current trends and developments are presented.

Kurzfassung

In dieser Arbeit werden Daten, die WLAN Clients senden gesammelt und analysieren, um diese zu tracken. Dazu werden nur Methoden betrachtet, die keine Kontrolle über die AP Infrastruktur erfordern und Clients müssen nicht unbedingt mit einem WLAN verbunden sein. Daher sind hauptsächlich Daten in Probe Request Frames interessant, die Clients senden, wenn sie aktiv nach WLAN APs suchen. Um das Verhalten von WLAN Clients zu analysieren, wird einen Aufbau aus mehreren Capture-Geräten und einem zentralen Auswertungssystem eingeführt. Die gesammelten Daten werden ausgewertet, um theoretische Konzepte zu untermauern. Ein großer Anteil an WLAN Clients gibt noch immer die Liste der gespeicherten SSIDs preis, wenn aktiv nach WLANs gesucht wird. Wenn aktiviert, bietet MAC Adressen Randomisierung signifikante Vorteile hinsichtlich dem Schutz der Privatsphäre. Benutzeridentitäten, die bei einer EAP Authentifizierung notwendig sind, werden aktuell von allen weit verbreiteten Betriebssystemen ungeschützt übertragen. Schließlich werden Ideen zur Erweiterung des Systems und theoretische Überlegungen erarbeitet, sowie ein Ausblick auf aktuelle und relevante Entwicklungen zum Thema WLAN Tracking gegeben.

Contents

List of Abbreviations

AP	Access Point
BPF	Berkeley Paket Filter
BSS	Basic Service Set
BSSID	Basic Service Set Identifier
cgroup	Control Group
CRC	Cyclic Redundancy Check
EAP	Extensible Authentication Protocol
EAPoL	Extensible Authentication Protocol over Local Area Network
ESS	Extended Service Set
ESSID	Extended Service Set Identifier
FCS	Frame Check Sequence
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
JIT	Just-In-Time
KRACK	Key Reinstallation Attack
LAN	Local Area Network
MAC	Media Access Control
MS-CHAP	Microsoft version of the Challenge-Handshake Authentication Protocol
OSI	Open Systems Interconnection
PEAP	Protected Extensible Authentication Protocol
QoS	Quality of Services
RADIUS	Remote Authentication Dial-In User Service
RF	Radio Frequency
RSSI	Received Signal Strength Indication
SA	Source Address
SSID	Service Set Identifier
STA	Wireless Station
TA	Transmitter Address
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UTM	Unified Threat Management
UUID-E	Universally Unique Identifier-Enrollee
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access
WPS	Wi-Fi Protected Setup

1 Introduction

WLAN is one of the most heavily used communication technologies today. The rise of mobile personal computers, smartphones, smart home and Internet of Things (IoT) devices demands good coverage of WLAN networks. Because of this, WLAN networks can be found in a lot of public places but also in private homes. WLAN itself only provides a replacement for cabling and is the only option in a lot of mobile devices where network connectivity via a copper cable is not possible because of size and mobility reasons. The air is a shared medium and all participants must share the available time and bandwidth with other devices in their coverage area. This is comparable to devices wired to an Ethernet hub, where the connected devices share the same collision domain. There is one other important consideration: Every other network participant can listen on the air and will get all of the data transmitted and received by other network participants. In trusted wired environments this is managable because the broadcasted data is only transmitted via controlled cables and infrastructure hardware. With wireless networks this introduces great security and privacy issues because the Radio Frequency (RF) waves can not easily be limited to tightly controlled infrastructure. They are often receivable from physical locations outside the controlled environment like a company building. Therefore, different encryption modes were developed but some data – especially meta data – has to be transmitted unencrypted.

In this work we take a look at the data that can be collected and analyzed by passively capturing IEEE 802.11 frames. We want to find out how this data affects the users' privacy and we want to find frames that contain identifiable information. Therefore, we build an application consisting of configured capture devices and analysis software to analyze the collected data for trackable information.

We only consider the most recent version of the IEEE 802.11 specification which is IEEE 802.11-2016 [1] in this thesis. Further we will not perform an advanced crypto analysis of the cryptographic algorithms used in the specification and will only focus on data that is easily accessible by collecting data from the air interface directly.

2 IEEE 802.11

To improve interoperability between multiple vendor chipsets and RF hardware the IEEE created a standard how to communicate with other network participants. IEEE 802.11 specifies the physical layer and media access control layer, which are part of the layers 1 and 2 in the Open Systems Interconnection (OSI) layer model [1]. As Institute of Electrical and Electronics Engineers (IEEE) 802.11 is part of the IEEE 802 family of Local Area Network (LAN) protocols it is designed to interwork with IEEE 802.**2** which specifies parts of the data link layer.

IEEE 802.11i. The first version of IEEE 802.11 includes the Wired Equivalent Privacy (WEP) security algorithm to prevent easy stealing of data by listening on the air interface. It is not very strong as it used 64 bit encryption which was later improved up to 256 bits but it suffers from other issues und was deprecated in 2004 [1]. We do not consider WEP in this paper as it is practically not used anymore.
Wi-Fi Protected Access (WPA) is the successor of WEP and was introduced as WPA2 in IEEE 802.11i-2004 [2]. The previous WPA version is known as IEEE 802.11i *draft* standard.

WPA3. In 2018, WPA3 was released as an improvement over WPA2 as some issues were found. The most famous ones are KRACK [3].

2.1 IEEE 802.11 Terminology

To understand important parts of the standard some basic terminology has to be introduced.

Wireless Station (STA). In IEEE Std 802.11, the addressable unit is a station (STA) [1]. It is an addressable destination similar to an address in wired LANs.

Basic Service Set (BSS). This is the basic building block of WLAN networks and describes a set of wireless stations that are in the same physical area and operate at the same medium access parameters like radio frequency and modulation scheme.

Basic Service Set Identifier (BSSID). This uniquely describes a BSS. It has to be 48 bit and, therefore, almost always the Media Access Control (MAC) address of the access point is used for that. Other possibilities are randomly generated numbers.

AP. A station with access to a distributed system and the ability to access the distributed system from the wireless medium forms an AP.

Distributed System (DS). This provides a connection between multiple APs – often via a wired IEEE 802.3 network.

Extended Service Set (ESS). This is used to combine multiple BSSs via a distributed system to a bigger unit to cover a larger physical area. The distributed system itself is not part of the ESS, so it only consists of the APs and devices connected to the APs taking part in the ESS.

Extended Service Set Identifier (ESSID). Similar to the BSSID this uniquely descibes an ESS. It is usually simply referred as Service Set Identifier (SSID) and is an up to 32 byte long identifier that can be chosen freely. It is displayed on devices that can connect to a WLAN network and acts as a name for the wireless network.

2.2 IEEE 802.11 MAC Frames

Beside other things IEEE 802.11 defines the smallest unit of data exchange via the wireless medium on the medium access control layer between two wireless stations. Each frame consists of

- a header containing control information, addresses and optional Quality of Services (QoS) information,
- a frame body described by type and subtype fields, and
- a 32 bit Cyclic Redundancy Check (CRC) Frame Check Sequence (FCS).

IEEE 802.11 defines four basic frame types with several subtypes defining the actual use of the frame.

- *Management Frames* are used between stations to exchange management information needed during connection establishment to an AP or the discovery of available APs and their parameters.
- *Data Frames* carry the actual payload data of higher OSI layers.

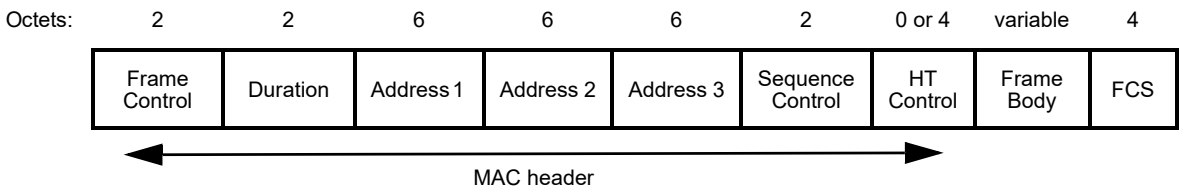


Figure 2.1: Management Frame Format [1].

- *Control Frames* assist with the delivery of data and management frames. Unlike data and management frames they do not have a frame body.
- *Extension Frames* are reserved for frames needed in the future but not defined yet. There are some frames with this type defined in IEEE 802.11-2016 but not considered in this thesis.

For our research probe requests, which are management frames, and data frames with Extensible Authentication Protocol (EAP) inside IEEE 802.1X transport frames contain valuable information and will be subject of further analysis.

2.2.1 Management Frames

The format of management frames is shown in Figure 2.1. As we do not consider multihop management frames here the *Address 2* field in all other subtypes of management frames contains the transmitter address (TA) which is also the source address (SA) in management frames and is the MAC address of the transmitting radio.

2.2.2 Probe Requests

Simple wireless stations must be able to scan for available AP. This can be achieved by passive or active scanning.

Passive Scanning. An AP continuously sends out beacon frames containing the ESSID, BSSID and other relevant parameters like supported data rates of APs configured to do so. The duration between two beacon frames is configurable with a default value of 100 ms. An STA that wants to connect to the AP continuously listens for beacon frames. Therefore, it switches through the available wireless channels which takes some time. This is considered too slow especially in context of mobile devices like smartphones.

Active Scanning. To compensate the issue of slow scanning the STA itself can initiate and speed up the search by sending out probe requests containing supported data rates to the broadcast address which should be received by all STAs being physically able to receive the frame on the MAC layer. If an AP receives a probe request frame it should

answer with a probe response containing similar data as beacon frames. The probe request frame is a management frame and can contain a list of fields in the frame body. The most interesting one is the *SSID* field, which acts as a filter to select only APs with a special SSID. If the *SSID* field is empty all APs reply with a probe response. If the *SSID* field is set to a special SSID then only APs which provide that service set answer the probe request.

2.2.3 WPS Data in Probe Frames

Wi-Fi Protected Setup (WPS) itself it not part of the IEEE 802.11 specification. It was created by the WiFi Alliance as an extension to IEEE 802.11 [4]. One of its main goals is to simplify connecting devices without a display or input method to a WLAN network and it is mainly used for printers and IoT devices to connect them to a home WLAN network consisting of a single AP. To establish a connection, one config method of WPS is the *PushButton Configuration* where the user has to press a physical or virtual button on the AP as well the STA device within 120 seconds to connect then to the same WLAN network.

Tracking Devices by Data in the WPS Extention Fields. Some of the extensions data is transmitted unencrypted in probe request frames. This includes the *Model Name* and *Model Number* fields which describe the device by ASCII strings and a *UUID-E* value – a unique GUID generated by the Enrollee (STA) that should be unique per device and generated by a pseudo randomnumber generator. This is a problem because the *UUID-E* field has to be constant for a device implementing WPS and can be used to uniquely track devices as this information is transmitted unencrypted in probe requests. Previous work has shown that it can be difficult to generate high entropy random numbers on embedded devices [5]. Therefore, the WPS standard proposes to use the MAC address and a random number which is initialized once per device as seed for the pseudorandom number generator. Some devices analyzed show that the manufacturer did not even initialize this random number properly and set the value to all-zeros. Knowing such devices from the *Model Name* and *Model Number* fields allows reconstructing the MAC address used to initialize the random number generator [5].

Tracking Devices by SSID Parameter in the Probe Request. If the SSID field is present in probe requests it can be a huge privacy issue because the STA device publishes the list of stored SSIDs, which the device was connected to. It sends out bulks of probe requests, each containing a SSID stored on the STA device. Therefore, the device can be tracked if the list of SSIDs is unique enough across other devices which seems to be the case pretty often.

2.2.4 IEEE 802.1X in Data QoS Frames

Because of its scalability IEEE 802.1X is especially interesting in educational and corporate environments. Eduroam is one of the biggest WLAN networks in the world and uses IEEE 802.1X with Extensible Authentication Protocol over Local Area Network (EAPoL) with a central RADIUS-Server (Remote Authentication Dial-In User Service) for authentication [6].

The Eduroam WLAN at Johannes Kepler University Linz uses the Protected Extensible Authentication Protocol (PEAP). It defines two tunnels to transmit data between supplicant and authentication server – an outer unencrypted tunnel which carries Transport Layer Security (TLS) data that protects an inner tunnel, which in addition uses the MS-CHAPv2 (Microsoft version of the Challenge-Handshake Authentication Protocol) protocol trying to add another layer of protection to the login information [7]. MS-CHAPv2 allows users to securely connect to the WLAN network using username and password. PEAP further defines an outer identity, transmitted in the unencrypted outer tunnel, which can be used by the Remote Authentication Dial-In User Service (RADIUS) server to decide which certificate to present to the supplicant for the TLS protected inner tunnel. MS-CHAPv2 inside the encrypted inner tunnel defines an inner identity which acts as a username or user identifier used for the actual authentication.

Many implementations suffer from a privacy issue, that if there is no outer identity specified explicitly, the inner identity (the username) is implicitly used as the outer identity too. This means that the username is transmitted unencrypted in the outer tunnel as outer identity which can easily be captured and extracted as defined by RFC 2865. Besides tracking individual devices the username may provide the possibility to directly track users using the device.

2.3 Data Considered in this Work

There are a lot of possibilities to track devices and persons via WLAN. This thesis only considers some specific portions of these possibilities. The first principle is to only use data that can be captured passively by using off-the-shelf WLAN adapters with drivers supporting monitor mode under Linux. Furthermore, we consider the following data:

- Private data in probe requests – SSIDs, MAC addresses and WPS UUID-E and
- Private data in EAPoL and IEEE 802.1X.

MAC addresses could be easily extracted from nearly all data frames but due to performance limitations of the capture devices these frames are ignored in this work.

MAC based Identification and Tracking. In addition to other captured data, non-randomized MAC addresses are considered unique and constant per network interface

of a device. This allows device reidentification and tracking by comparing previously captured MAC addresses.

3 Application

To be able to further investigate private data, some capture of WLAN frames in public environment has to be done. For efficient collection and analysis a set of tools is used from capturing over decoding protocol layers, aggregating data from multiple capture devices to storing extracted data in a database, analyzing and visualizing it.

3.1 Application Architecture and Tools

Figure 3.1 shows the general high level architecture of the implemented solution to capture and analyze IEEE 802.11 frames.

Application Architecture. For each specific task in the pipeline, tools licensed under some open source license exist and can be configured to work as needed. Therefore, the application consists mainly of tool configurations and redirection of outputs and inputs of some form. Physical devices are used in this part of the setup.

Capture Devices. To be able to capture on multiple physically different locations and wireless medium frequencies multiple capture devices are needed. Because of easy availability the hardware of Cyberoam CR 15wi UTM Appliances is used. It consists of a 500 MHz VIA Eden CPU which is able to execute x86 code, 1 GB of DDR2 SDRAM, three 100 Mbps Ethernet Adapters and a miniPCI WLAN card with a Mediatek RT2860 chipset and three external antennas. This application is designed to only use the 2.4 GHz WLAN band with the IEEE 802.11bgn standards. The operating system loads from a 4 GB CompactFlash card. The original Unified Threat Management (UTM) software is replaced by an up to date Debian Linux for simple customization and due to availability of precompiled tools.

Transport Network. To be able to transmit the data between the capture devices and the analysis machine some kind of OSI layer 3 network is used. In our case the pcap data is sent via a TCP connection. As each capture device can produce up to 100 Mbps stream of capture data the transport network has to be able to handle that but this is not a problem for an up to date local network.

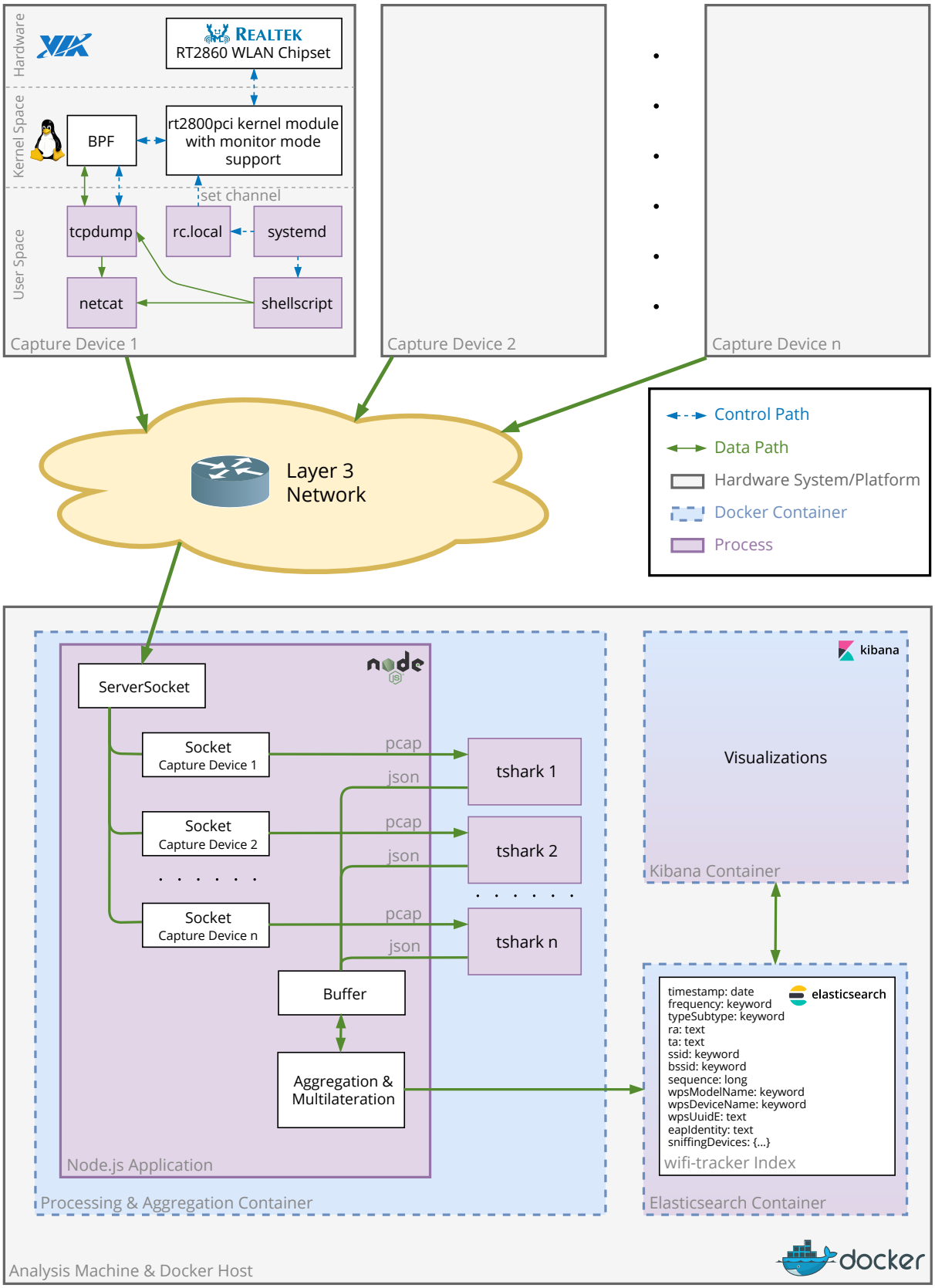


Figure 3.1: Application architecture

Analysis Machine. This is the place where all data are aggregated, stored and analyzed. As we packed the tool stack into a set of Docker containers it has to run a recent version of Linux or a Windows 10 version that is supported by Docker.

3.1.1 Tcpdump

rt2800pci Driver. The Mediatek RT2860 chipset is supported by the rt2800pci driver which implements the needed monitor mode functionality [8]. rt2800pci is part of the standard Debian package repositories so it is very easy to set it up [9].

Radiotap. Radiotap is a de facto standard for IEEE 802.11 frame injection and reception [10]. This means it is a protocol that adds meta data to captured frames like hardware timestamps and RSSI values received by the WLAN hardware and driver.

Tcpdump. Tcpdump is a widely known command line packet sniffer [11]. Besides its name it can capture data on various OSI layers including TCP but also IEEE 802.11 WLAN and radiotap frames. It is able to compile a pcap-filter expression for Berkeley Paket Filter (BPF) which is Just-In-Time (JIT) compiled to native code on the executing platform. The BPF filtering is executed in kernel space and therefore is more performant than (zero-)copying data to the userspace program and discarding non-needed packages there. Performance improvements are beneficial in this situation because packet drops appear occasionally when filtering in userspace as a reason of limited CPU power of the capture devices.
In our setup we discard all frames except probe request frames and frames containing EAPoL data. EAPoL data has a specific ethernet type which can be matched without further dissecting and decoding the protocol data.
Tcpdump's native libpcap savefile format can be seen as the common denominator of formats used by open source tools.

3.1.2 netcat

Netcat is a simple tool that sends standard input to a TCP connection or UDP socket and redirects data read from the TCP connection or UDP socket to standard output. It can act as a server by listening for incoming connections on a specific port or act as client connecting to a listening TCP or UDP socket [12]. When using standard input and output streams some data copying by the shell in userspace is required so the use of netcat is potentially not the most efficient one but it provides great flexibility compared to kernel space or zero-copy optimizations. Piping standard output to standard input streams in userspace was considered acceptable in terms of CPU load compared to the more CPU intensive packet filtering.

3.1.3 Systemd Script on Capture Devices

To automate startup after the boot process and restart after an unexpected failure a service definition for the default init system in modern Linux distributions (systemd) is used. Systemd starts a shell script that runs tcpdump and pipes the filtered output to netcat which connects to another netcat in listening mode on the analysis machine and sends the captured data to that analysis machine.

3.1.4 Docker Engine

Docker is a platform which can manage and run a number of containers. Each container internally is an encapsulation by Linux kernel features like namespaces and Control Groups (cgroups). Isolating a process from other processes on the same host can be achieved using namespace isolation. Containers only see a portion of the overall system. cgroups control the process, memory and CPU available to processes running in a cgroup. They also keep track of used memory [13]. It is very convenient to use these kernel features through Docker compared to direct use. In our application it is used to encapsulate the tools and processes running on the analysis machine as they are mostly hardware independent. In addition, encapsulating the processes and needed resources in containers allows a platform independent deployment. On the capture devices only a small set of simple processes run and so encapsulating them would be overkill. Besides that Docker requires a 64 bit CPU and operating system so running Docker on capture devices is not possible.

3.1.5 Node.js

Node.js uses the Google Chrome V8 JavaScript engine and combines it with an ecosystem that is designed to run server side JavaScript. It is very simple to create a server socket which is used to substitute the functionality of the netcat instance running on the analysis machine in listening or server mode [14]. Besides that it is easy to start processes and pipe the output of a socket into the standard input stream of a process. With a dynamic language like JavaScript the things needed in this application can be achieved much more elegantly and dynamically than with plain shell scripts. Besides that there are official elasticsearch libraries to easily access and write data to an elasticsearch instance or cluster.

3.1.6 Wireshark and tshark

Wireshark is a popular network protocol analyzer which can be used to read input in the libpcap savefile format. It has a lot of dissectors for analyzing a great number of protocols on different OSI layers. In our case the most important protocol dissectors are radiotap, 802.11, EAP and EAPoL.

Tshark is a kind of command line version of Wireshark and is released as part of the Wireshark tool family. Since Wireshark version 2.2.0, tshark is able to output Elasticsearch-compatible JSON [15]. This is useful because since the rise of JavaScript in the browser and the REST architecture data in JSON format has got great tool support.

3.1.7 Elasticsearch

Elasticsearch is a NoSQL database and search engine which is based on the search engine software library Apache Lucene. Elasticsearch is often used for providing full-text search or global search functionality in websites. Documents – which are the equivalent to datasets in SQL – are stored in indices – which are the equivalent to tables in SQL databases. The native format of documents is JSON which has the advantage of being easily consumable by web based tools like Kibana. The ability to store and index complex and nested object structure compared to classic rational SQL databases is an advantage for our application. Even if an index schema is not explicitly needed it is specified for our application because it ensures that the indexed fields are of the desired data type independently of the content of these fields. Besides standard field types like `date`, `text` and `keyword` the `nested` for storing nested objects, `ip` for storing IP addresses and `geo_point` for storing geographic coordinates in the WGS84 system are used. Once data is indexed into an Elasticsearch index it can be queried with a JSON based domain specific query language.

3.1.8 Kibana

Kibana is a web based tool for visualization of data queried from an Elasticsearch database. It provides a flexible way of defining multiple visualizations combining them to dashboards and has an autorefresh feature. Is is primarily meant for analysis of filtered and processed logs stored in an Elasticsearch cluster in an easy configurable way to extract information quickly.

3.2 Application Pipeline

The application pipeline describes the combination of data and control flows between the basic building blocks mentioned above. Figure 3.1 shows the main pipeline components and their interactions.

Layer 3 Network. To connect the capturing devices with the analysis machine a network with OSI layer 3 connectivity is needed – NAT will cause problems if involved because the capture devices are identified by their IP address. The capturing devices obtain their IPv4 address via DHCP so a DHCP server is required that assigns static leases based on the MAC address of the capture devices. In the simplest case a layer

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.

iw phy phy0 interface add mon0 type monitor
iw dev wlan0 del
ifconfig mon0 up

exit 0
```

Listing 3.1: Content of `/etc/rc.local`

2 switch and a DHCP server is enough to connect all necessary physical components of the application. The bandwidth needed depends on the number of sniffing devices active and the traffic inside a "WLAN air cell". The sniffing devices feature only a 100 Mbps Ethernet port but it is typically only utilized by a single digit percentage as most of the traffic in WLAN are frames filtered directly by the sniffing devices.

Analysis Machine Bootup. As this is the central point in collecting frames the analysis machine should be started before all other components. As long as the machine can run Docker containers and has enough CPU power, memory, and storage any host system will do the job.

Capture Device Bootup. Finally, we need to start the actual capture hardware. After bootup of the prepared image a virtual device in monitor mode is created. The channel is set to a hardcoded value which is channel 1 with 20 MHz bandwidth and the device in monitoring mode listens for frames according to the 802.11n standard by default. This is done by the `rc.local` compatability service started by systemd. `rc.local` configuration is shown in listing 3.1.

After creation of the capturing device a second systemd service, configured by the `/etc/systemd/system/wifi-tracker-capture.service` file with content in listing 3.2 starts the `/etc/wifi-tracker-capture/wifi-tracker-capture.sh` shell script with content in listing 3.3.

The shell script started by systemd starts tcpdump with hard coded filter parameters which result in compiling and installing Berkely Packet Filter in the kernel. Unnecessary packets are directly discarded in kernel space without copying them to the user space which helps to achieve a better performance because the processing power of the small VIA Eden CPU is quite limited. The shell script pipes the output of tcpdump to netcat.

```
[Unit]
Description=Wifi Tracker Capture Service
After=rc-local.service

[Service]
Type=simple
ExecStart=/etc/wifi-tracker-capture/wifi-tracker-capture.sh $OPTIONS
ExecReload=/bin/kill $MAINPID
KillMode=process
Restart=always
RestartSec=10s
TimeoutStartSec=infinity

[Install]
WantedBy=default.target
```

Listing 3.2: Systemd service definition of `wifi-tracker-capture.service`

```
#!/bin/sh

# "-s" bytes per packet to store (0 means the entier packet)
# "-U" do not wait for the buffer to fill, before sending the data out
# "-n" prevent converting addresses (host addresses, port numbers) to names
# "-w -" write the pcap output to file while "-" is the stdout
# "-i mon0" input device to read from
# "wlan type mgt" BPF for filtering 802.11 Management Frames
# "ether proto 0x888e" ether type for 802.1X Authentication

tcpdump -s 0 -U -n -w - -i mon0 "wlan type mgt subtype probe-req or (wlan type data subtype qos-data and (ether proto 0x888e))" | nc 192.168.99.1 11000
```

Listing 3.3: Content of `wifi-tracker-capture.sh`

netcat connects via TCP to the hard coded IP address of the analysis machine. In case of a connection error both tcpdump and netcat and therefore the shell script terminate. Systemd restarts the shell script after a few seconds. This not only means it is not strictly necessary to start the analysis machine first but also provides reconnection functionality if the connection is lost because of arbitrary reasons.

Incoming Connection at the Analysis Machine. If a netcat instance on a capture device connects to the analysis machine – more precisely to the listening ServerSocket of the Node.js application inside the Docker container – the Node.js application starts a tshark process and pipes the incoming data stream to the tshark process. This means per connection to the Node.js analysis application a tshark process is started. If the connection is closed the tshark process is terminated.

Incoming Frame Processing in Capture Devices. If the virtual device in monitor mode captures a 802.11 frame the driver adds metadata in Radiotap protocol format to the beginning of the frame. Radiotap fields interesting for this application include

- *Antenna signal* in dBm which is later used for path loss based position estimation, and
- *Antenna* which is the index of the antenna on which the frame was received. In combination with *Antenna signal* this can help to identify antenna or reflection issues which are difficult to debug. However the rt2800pci driver used in the application as described in section 3 does seem to report only antenna index 1 for each captured frame which is the second antenna as the antenna index starts with 0. Other Antenna indices have never been received during analysis work.

Tcpdump encodes these frames in the industry standard pcap format and sends them to standard output where the shell pipes them into netcat which transmits the pcap data via TCP to the analysis machine.

Incoming Frame at the Analysis Machine. The frames in pcap format arrive at the analysis machine where the Node.js application pipes them to the tshark process which dissects them and extracts the following fields:

- `frame.time_epoch` the time the frame was captured at the capturing device.
- `wlan_radio.frequency` the base frequency of the WLAN channel – e.g. 2412 MHz for channel 1.
- `wlan_radio.signal_dbm` the received signal strength – also known as RSSI.
- `wlan.fc.type_subtype` the 802.11 type including the subtype.
- `wlan.ra` the 802.11 receiver address – this contains the MAC address of the receiving WLAN hardware.
- `wlan.ta` the 802.11 transmitter address – this contains the MAC address of the transmitting WLAN hardware.
- `wlan.ssid` the 802.11 SSID string – this is especially interesting because if it is included in probe requests it can leak private information about stored WLAN networks on the device sending the probe request.
- `wlan.bssid` the 802.11 BSSID – this is the MAC address of the access point.
- `wlan.seq` the 802.11 Sequence Number – in 802.11 it is used to detect retransmitted frames, in our application we use it to improve the entropy when comparing frames received from multiple capture devices.
- `wps.model_name`, `wps.model_number`, `wps.device_name`, `wps.uuid_e` are interest-

ing fields belonging to the WPS – Wireless Protected Setup – part of 802.11. This information is included in the probe requests of some devices that support WPS. As noted in section 2.2.3 it may include interesting and trackable information about the device.

- `eap.identity` is not directly related to 802.11 but is used in 802.1X which is used as part of the authentication procedure in WPA2-Enterprise. It contains the outer identity sent to the RADIUS server and in several implementations copies the username to the unencrypted tunnel between the supplicant and the authentication server – typically RADIUS.

The frames received on multiple capturing devices are aggregated into a single object containing information about the capture devices on which the frame was captured. The output of tshark is a stream of JSON objects – also known as streaming JSON which the Node.js application parses. The objects are stored in a buffer for at least one second to wait for some time to fully receive the frame that may be captured on multiple devices and may vary in arrival times due to jitter in the kernel, network or other points on the way to the analysis machine.
If the frame is received on at least four capture devices multilateration as described in section 3.3 is calculated and attached to the frame object.
After completely processing the frame it is stored in an elasticsearch index for further analysis and visualization.

Analysis in Kibana. For convenient and flexible analysis Kibana is used to query and visualize the data captured. As there are only four capture devices with overlapping coverage ranges involved in the evaluation setup, IEEE 802.11 frames are captured only in a small geometric area.

3.3 Position Estimation with Multilateration

Multilateration is a technique used in navigation and surveillance to find the position of a source transmitting RF waves. It is often based on the time of arrival in media with known propagation velocity. With our application time of arrival is not easily feasable to measure because the speed of propagation of RF waves is near to the speed of light and this would require more specialized hardware to measure. An interesting approach for IEEE 802.11n in this direction can be found in [16].

Besides time of arrival, estimating the position based on the angle of arrival should work in theory with the concept of multilateration. If the WLAN supports multiple antennas and streams it is technically possible to calculate the angle of arrival of an RF wave based on the time difference of arrival. IEEE 802.11n typically does not do that because there is no need in the rest of the protocol. In IEEE 802.11ac there is support for beamforming where all available antennas are combined into a phased-array antenna to force propagation of an RF wave in one direction. Besides the implementation issues

in IEEE 802.11bgn there is no defined radiotap field for the angle of arrival. Position estimation based on angle of arrival with specialized hardware can be found in [17]. The method shown here is based on [18].

Path Loss Model. Multilateration also works for Received Signal Strength Indication (RSSI) values in combination with the path loss model. The pass loss model assumes homogenious signal attenuation and is defined as

$$P_{r,i} = P_0 - 10n \log_{10}\left(\frac{d_i}{l_0}\right) + X_\sigma, \tag{3.1}$$

where $P_{r,i}$ is the received signal energy measured in dBm received at the capture device, P_0 is the signal energy at the transmitter, n is the path loss exponent describing the attenuation of the homogenious environment, d_i is the distance between capture device i and the WLAN device sending IEEE 802.11 frames, l_0 is the reference length – in this application 1 m makes sense and X_σ is the power added by the noise floor. In radiotap there is a field for the noise floor in dBm received on the antenna but as we assume the noise has uniform distribution the expectation is exactly 0, so we omit X_σ from now on.

In the first step we want to estimate the distance d_i as

$$d_i = 10^{(P_0 - P_{r,i})/10n}. \tag{3.2}$$

Multilateration. On the other hand we geometrically calculate the distance from the i^{th} capture device to the WLAN device sending the frames as

$$d_i = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2}. \tag{3.3}$$

After eliminating d_i from both equations above and doing some linear refactoring of the equation system resulting from multiple capture devices the equation system results in the form of $y = Xb$:

$$\begin{aligned}
&\begin{bmatrix}
-x_1^2 - y_1^2 + x_k^2 + y_k^2 + 10^{\frac{\left(P_0 - P_{r,1}\right)}{5n}} - 10^{\frac{\left(P_0 - P_{r,k}\right)}{5n}} \\
-x_2^2 - y_2^2 + x_k^2 + y_k^2 + 10^{\frac{\left(P_0 - P_{r,2}\right)}{5n}} - 10^{\frac{\left(P_0 - P_{r,k}\right)}{5n}} \\
\cdots \\
-x_{k-1}^2 - y_{k-1}^2 + x_k^2 + y_k^2 + 10^{\frac{\left(P_0 - P_{r,k-1}\right)}{5n}} - 10^{\frac{\left(P_0 - P_{r,k}\right)}{5n}}
\end{bmatrix} \\
&= \begin{bmatrix}
-2x_1 + 2x_k & -2y_1 + 2y_k \\
-2x_2 + 2x_k & -2y_2 + 2y_k \\
\cdots & \cdots \\
-2x_{k-1} + 2x_k & -2y_{k-1} + 2y_k
\end{bmatrix}
\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}.
\end{aligned} \tag{3.4}$$

$b = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$ could in theory be estimated by solving this system with

$$b = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \left(X^T X\right)^{-1} X^T y. \tag{3.5}$$

The result includes the unknown transmit power at the sender P_0 and the path loss exponent n. The system gets much easier by approximating the path loss model with

$$d_i = 10^{\frac{\left(P_0 - P_{r,i}\right)}{10n}} \approx a_0 + a_1 \frac{P_0 - P_{r,i}}{10n}. \tag{3.6}$$

After doing some linear refactoring of the equation system resulting from multiple capture devices again the equation system results in the form of $y = Xb$.

$$\begin{bmatrix} -x_1^2 - y_1^2 + x_k^2 + y_k^2 \\ -x_2^2 - y_2^2 + x_k^2 + y_k^2 \\ \cdots \\ -x_{k-1}^2 - y_{k-1}^2 + x_k^2 + y_k^2 \end{bmatrix} = \begin{bmatrix} -2x_1 + 2x_k & -2y_1 + 2y_k & \frac{a_1\left(P_{r,1} - P_{r,k}\right)}{5} \\ -2x_2 + 2x_k & -2y_2 + 2y_k & \frac{a_1\left(P_{r,2} - P_{r,k}\right)}{5} \\ \cdots & \cdots & \cdots \\ -2x_{k-1} + 2x_k & -2y_{k-1} + 2y_k & \frac{a_1\left(P_{r,k-1} - P_{r,k}\right)}{5} \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ \frac{1}{n} \end{bmatrix} \tag{3.7}$$

This can again be solved with equation 3.5.

Limitations and Edge Cases. This needs at least four capture devices receiving the same frame. No other parameters are needed. There are limitations in geometric alignment of the capture devices – these includes all cases where the $X^T X$ matrix in equation 3.5 with X from equation 3.7 is singular. This is the case if more than two capture devices are on the same geometric line – in this case one capture could be removed without loss of information but this is not implemented in the application and in such a case an error is thrown and the position estimation via multilateration is skipped.

4 Findings

4.1 Capture Setup and Environment

With the application described in section 3 two datasets were captured. The test setup consists of four capture devices located in the inner courtyard of the student dorm of Akademikerhilfe in Linz in Pulvermühlstraße 41. The geometric alignment of the capture devices and test point locations can be seen in figure 4.5.

- One dataset obtained from 2019-07-04 01:24 to 14:30 local time contains all probe requests and EAPoL outer identities for WPA2-Enterprise authentication. Capturing during night and day helps to approximate the average amount and data of probe requests. This dataset consists of 101,877 captured and aggregated frames – 101,717 of them are probe requests 160 are data QoS frames and 133 of them contain an EAP identity from the outer tunnel of EAPoL authentication.

- The second dataset was obtained by capturing and discarding all probe requests except from one test device. This dataset is trimmed to evaluate the position estimation via multilateration in a simple field test. The test device without MAC address randomization was set to 12 test locations spread across an inner courtyard of a U-shaped building. WLAN was enabled after positioning the device until 10 probe requests were received on all four capture devices without human interaction. The test device was a Samsung Galaxy Nexus (GT-I9250) with Android 4.4 – this device sends approximately one probe request per second if the screen is on and the WLAN settings menu is open displaying currently available WLAN networks.

4.2 MAC Address Randomization

IEEE 802.11 Transmitter Address. Every 802.11 management frame consists of multiple address fields. One that is always populated is the TA field in the MAC header of the management frame [1]. The TA is a 6 byte field which according to the 802.11 standard has to be the MAC address of the transmitter radio.

MAC Address. As in 802.3 Ethernet, every 802.11 capable device needs a unique Extended Unique Identifiers 48 bit (EUI-48) address which is also known as the MAC address. As shown in figure 4.1 a MAC address is made up of two parts each consisting

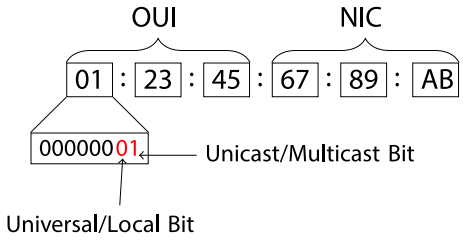


Figure 4.1: 48-bit MAC Address Structure

of three bytes in size. The first half is made up by addresses from a MAC Address Block Large (MA-L) which were previously known as the Organizationally Unique Identifier (OUI) [19]. The second half can be assigned freely by the organization that has bought the three octet prefix. A MA-L block can be purchased and registered by the IEEE for use as the first three octets of the MAC address. By default OUIs do not have the local bit set which means that they are globally unique. Besides OIDs there is the possibility to register Company Identifiers (CIDs) at the IEEE which do not have to be unique in a global manner and therefore can be used by multiple devices. If the local bit is set the address is also known as Locally Administered Address (LAA). LLAs are typically used as MAC addresses for software or virtual interfaces like a VLAN interface or at APs with multiple SSIDs per radio.

MAC Address Randomization. As the MAC address is usually globally unique and mobile devices send probe requests on a regularly basis this can be used to track mobile devices and their users. Operators that run large city or nation wide WLAN infrastructure networks can track mobile devices very accurately. In theory big ISPs that provide modem devices that also act as router, switch and WLAN access point to their customers can in theory capture probe requests and track mobile devices across all customers. As an example UPC – an ISP operating in Switzerland and under the Magenta brand in Austria runs a service called "Wi-Free" that adds a second SSID to APs at customer sites which are available for all other UPC customers [20]. Such projects let assume that it is technically possible for ISPs to also collect and process probe request captures.

To prevent such tracking, mobile device and operating system vendors have implemented counter measures to prevent trivial MAC address based tracking which are mainly based on changing the MAC address for certain frames. This is similar to the Privacy Extensions defined in RFC 4941 for IPv6.

When MAC addresses are changed.

- If a WLAN station is connected to an access point and transmits or receives data frames to or from this access point carrying payload for higher OSI layers we are not aware of any changes of the MAC address as this could in case of a collision possibly lead to data loss. Therefore, the MAC address is not changed during the period of being connected to an WLAN.

- If a WLAN station is not connected to an access point MAC address randomization is typically used for probe requests. Devices change their MAC address for probe requests in an interval of several minutes or per active scan procedure. This could be verified by capturing data of devices in an environment where reception of probe requests with a very specific SSID from other devices is very unlikely. Figure 4.2 shows captured probe requests for a specific SSID with different TAs over time from the same HUAWEI P9 lite running Android 7.0.

- Some devices change their MAC address for probe requests also when they are connected to a WLAN access point – see the following section on, "Implementation on specific devices and OS versions" for more details.

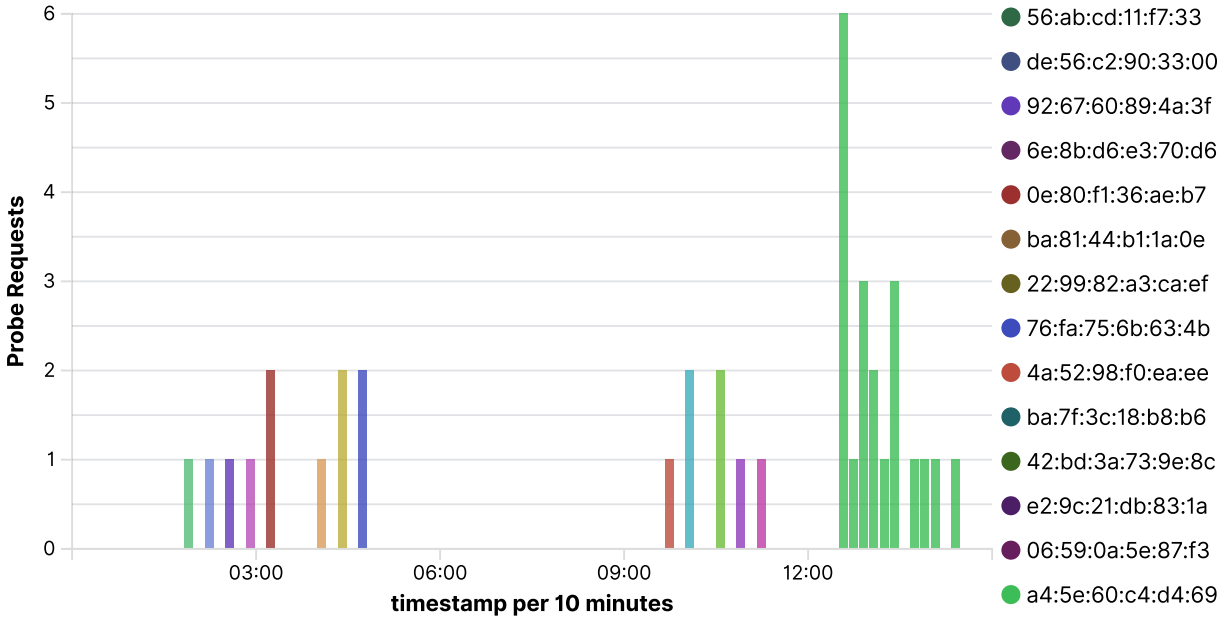


Figure 4.2: Directed Probe SSID Distribution of the first Dataset

How MAC addresses are changed.

- One way is to use LAAs for MAC addresses for probe requests. MAC addresses used for 802.11 should be globally unique but in field use it does not make a huge difference if there are two or more devices using the same MAC address for probe requests.

- During MA-L block registration the default option is to list the MA-L block to customer relation publicly. As of publishing of this paper a MA-L block can be registered by a one time fee of US$2,905. If customers do not want to be listed publicly by the IEEE they can prevent this by a yearly confidentiality fee of US$3,360. It seems that the vast majority of IEEE MA-L block customers do not pay extra for confidentiality. This lets vendors that implement MAC address randomization assume unpublicly listed assignments are unassigned and are not used in the field by other devices so these devices use MAC addresses not assigned publicly by the IEEE as temporary random MAC addresses for probe requests.

According to [21] the majority of MAC address randomizing devices use unassigned MAC address prefixes for their random MAC addresses. Even through this is against the IEEE idea of global uniqueness to our knowledge there is no reaction to this from the IEEE.

Implementation on specific devices and OS versions. The behavior of MAC address randomization depends largely on the software stack used on devices and can change with software updates. There is a trend to implement MAC address randomization and other privacy protecting features in general in more recent operating system versions.

- "*Apple platforms use a randomized Media Access Control (MAC) address when performing Wi-Fi scans when not associated with a Wi-Fi network. These scans can be performed to find and connect to a known Wi-Fi network or to assist Location Services for apps that use geofences, such as location-based reminders or fixing a location in Apple Maps. Note that Wi-Fi scans that happen while trying to connect to a preferred Wi-Fi network aren't randomized. [...] Wi-Fi MAC address randomization support is available on iPhone 5 or later.*" [22]

- "*If the device always uses the same Wi-Fi MAC address across all networks, network providers and other network observers can more easily relate that address to the device's network activity and location over time. This allows a kind of user tracking or profiling, and it applies to all devices on all Wi-Fi networks.*
  To reduce this privacy risk, iOS 14, iPadOS 14 and watchOS 7 use a different MAC address for each Wi-Fi network. This unique, static MAC address is your device's private Wi-Fi address for that network only." [23]

- "*Starting in Android 8.0, Android devices use randomized MAC addresses when probing for new networks while not currently associated with a network. In Android 9, you can enable a developer option (it's disabled by default) to cause the device to use a randomized MAC address when connecting to a Wi-Fi network.*

  In Android 10, MAC randomization is enabled by default for client mode, SoftAp, and Wi-Fi Direct. [...]

  Additionally, MAC addresses are randomized as part of Wi-Fi Aware and Wi-Fi RTT operations." [24]

  The implementation in Android is only a starting point as Google, as the creator and maintainer of Android, does not produce a notable share of the available Android devices and the device vendors have to add support for MAC address randomization by implementing the Android Wi-Fi Hardware Interface Design Language [25].

- According to [26] and [27] Windows 10 has support for MAC address randomization if the hardware and driver supports it. Personal experience shows that this option is available on an HP EliteBook 840 G5 with an Intel Dual Band Wireless-AC 8265 WLAN adapter and Windows 10 Pro 1903 but not on a Packard Bell EasyNote

LS with an Atheros AR5B97 WLAN adapter and Windows 10 Home 1809.

4.3 SSID Values

As described in section 2.2.2 there is a difference between directed and null probe requests. In the first captured dataset 24.28% of the captured probe requests were null probe requests while the rest had an SSID value included. To put that into perspective, devices that send directed probe requests including the SSID typically send bursts of probe requests – one for each SSID they have stored connection credentials on the device and maybe have been used recently. Due to the fact that we do not know which share of the probe requests use randomized MAC addresses further analysis is complicated.

Figure 4.3 shows the top hit SSIDs. It is evident that the SSIDs in the probe requests are not distributed evenly because some SSIDs are used by much more devices than others.

- `BJNPSETUP` with 48,621 captured probe requests: This is by far the most seen SSID in probe requests and is sent by a single TA approximately 50 to 60 times a second. It seems to originate from a WLAN enabled CANON printer that is constantly probing for the default SSID to which it tries to connect to. This SSID is omitted from now on as it does not add any further information to the analysis.

- Missing SSID in 24,577 captured probe requests: This is a considerable share taking into account that devices that send directed probes send bursts of probe requests.

- `AHL` with 12,228 captured probe requests: AHL is the official SSID of the student dorm the capture was run. Almost all students in the dorm are connected with at least one device to this SSID. The network consists of 80 APs so seeing probe requests from devices in this area is not unusual.

- `TP-LINK_714A` with 9,780 captured probe requests: This is an AP not under control of the network administrators at the student dorm and may be a rogue AP operated by one of the students or it could be an AP located in the near neighborhood of the dorm or a device desperately probing for that SSID. To further investigate this a service called Wireless Geographic Logging Engine (WiGLE) where community members can share and publish their wardriving results was queried. The maps at WiGLE.net showed an AP near the student dorm. Even if several WLAN devices probe for this SSID the number seems high compared to other captured SSID data in probe requests but may be reasonable.

- `eduroam` with 513 captured probe requests: As eduroam is one of the biggest WLAN networks in the world this is an interesting observation. In the student dorm there are two APs sending eduroam beacons. As the capture location is mainly used by students that use the eduroam WLAN on their devices when being on campus. Compared to probe requests for `AHL` this is interestingly low as almost

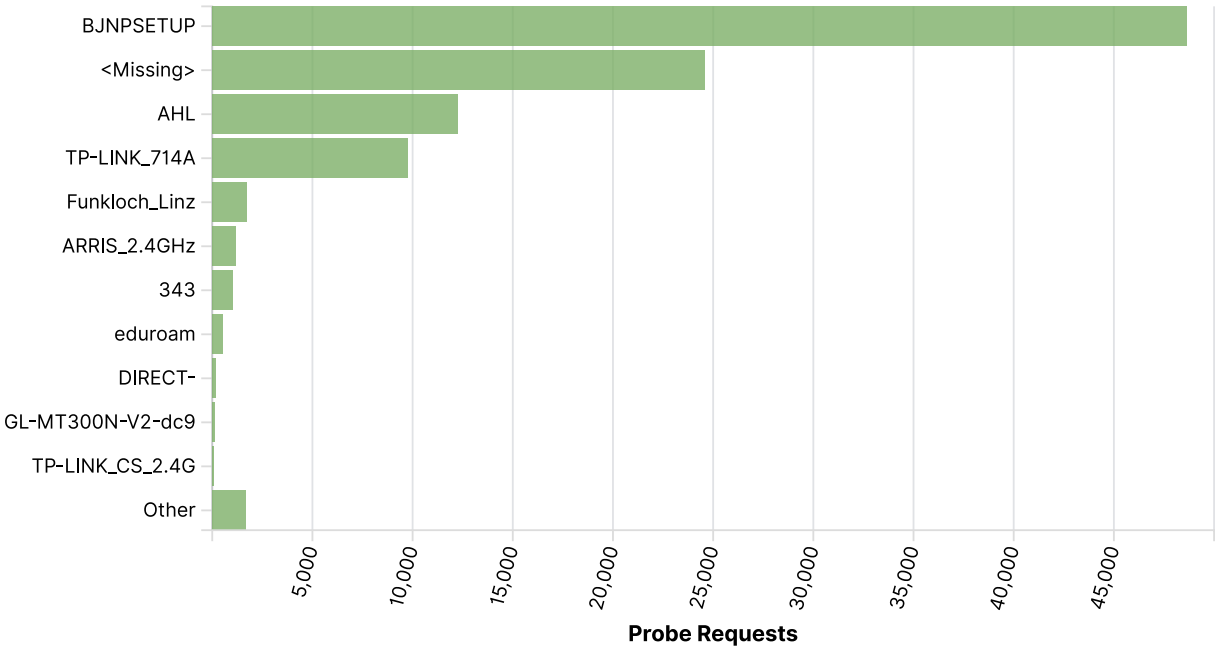


Figure 4.3: Directed Probe SSID Distribution of the first Dataset

all students living at the dormitory can connect to the eduroam WLAN. As most devices use the `AHL` WLAN, which provides good coverage, these devices probe the `AHL` much more often than the `eduroam` SSID which they are not connected that often. The exact reason is subject for further investigation.

- `JKU` which is broadcasted by the same APs as `eduroam` seems, with only 10 directed probe requests, to be used much less. This assumtion is reinforced by looking at the JKU-wide WLAN device registration statistics at `https://ekg.jku.at`.

During the capture process 140 SSIDs were captured in directed probe requests.

4.4 WPS UUID-E

As shown in [28] a wrongly initialized random number generator can enable tracking of users with the UUID-E entry in the WPS extension of probe requests. On current mobile devices WPS is disabled by default and can be enabled if needed. As shown in figure 4.4 some devices with WPS UUID-E data in probe requests could be found during obtaining the first dataset. The table shows that there were no MAC address randomization or changes of the UUID-E over time as far as this can be assumed by the number of captured probe requests containing WPS data. Some devices seem to send probe requests with WPS data frequently while others may be captured only while walking near the capture devices.

wpsDeviceName: Descending	wpsModelName: Descending	wpsModelNumber: Descending	wpsUuidE: Descending	ta: Descending	Count
DESKTOP-7300SKN			31:26:71:34:e1:b3:46:59:a9:e7:b7:fa:da:50:ac:f4	42:9f:38:56:ea:57	135
DESKTOP-QCKEUQ1			0f:c4:d4:d7:51:c7:4b:3e:a3:e0:e1:d6:59:eb:1d:c5	34:e1:2d:ed:93:8e	60
Che2-L11	Che2-L11	Che2-L11	a3:b9:9a:7f:02:53:52:4c:bc:7a:92:e3:41:77:cc:8a	58:7f:66:14:6b:fc	8
S10A	Doro 8040	Doro 8040	25:75:4a:ba:b4:09:59:6d:ab:af:40:b5:16:b2:c9:5e	00:1d:29:c8:bd:fb	8
BL7000	BL7000	BL7000	a0:d5:ac:b8:3b:af:52:b0:8e:cb:52:d7:9c:9d:c2:68	00:08:22:a2:75:fc	2
SVP4KDTV15_EU	BRAVIA 4K 2015	BRAVIA 4K 2015	7f:a9:de:f2:85:ff:56:34:b5:f9:7f:cd:2f:37:8a:d7	68:14:01:64:26:7f	1
Viera Lan	WPS_SUPPLICANT_STATION	01234567	be:27:71:78:8d:20:52:b4:81:3c:a5:cd:26:d6:98:47	24:ec:99:cb:a6:bc	1
emporia S1	emporia S1	emporia S1	97:cb:27:5f:c5:6e:5c:94:8a:6e:91:3a:1b:af:0c:f3	bc:e5:9f:53:50:eb	1
suez	KFSUWI	KFSUWI	b2:5e:2b:56:dc:2c:5d:48:94:16:05:1d:5e:bd:15:d7	cc:f7:35:ca:84:e7	1

Figure 4.4: WPS Data in Probe Requests in the first Dataset

4.5 EAPoL

In the EAPoL data encapsulated in IEEE 802.11 QoS data frames during the authentication process we only take a look at the EAP identity field. This is typically used for the WPA2 and WPA3 Enterprise authentication methods. Especially EAP-PEAP with the password based phase 2 authentication method MS-CHAPv2 was observed while other methods are not taken into account. There are several implementation specific details that can be observed from the dataset:

- Android devices up to version 7.1 but most likely even up to version 10 copy the user identity (username for MS-CHAPv2) to the anonymous identity field if this is not populated. This anonymous identity is also known as the outer identity in RADIUS protocol and is transmitted unencrypted. This is a good source of tracking information because the device is already in the transition to association state with the AP and does not use randomized MAC addresses. Besides that, a mapping between the MS-CHAPv2 username and the devices non-randomized MAC address can be established. If further data frames transmitted in associated state are captured they can be mapped to a username with high probability. In this case FreeRADIUS – a popular open source RADIUS server implementation used by eduroam – logs a warning message "`WARNING: Outer and inner identities are the same. User privacy is compromised.`" [29] but this message is typically only visible to the administrators of FreeRADIUS. Eduroam operators are aware of the problem and state a warning in their eduroamCAT Android app [30] `Anon ID missing (optional)`. If users are aware of the privacy problem they can explicitly specify an outer identity.

- In iOS up to version 14 there is no dedicated GUI field for the outer identity and users can only enter data into a username and a password field when connecting to an SSID.

- In Microsoft Windows 10 there is the possibility to specify an `identity privacy`

`(outer identity)` in enterprise profiles [31] which helps to protect privacy. As observable from the dataset Windows 10 versions 1903 uses the actual MS-CHAPv2 username as PEAP outer identity if the WLAN profile is created by using the GUI by clicking the WLAN icon in the tray area next to the clock on the right bottom corner of the screen. Since Windows 10 Version 2004 there is a GUI field for entering the value for `identity privacy (outer identity)` when manually adding a WLAN profile.

4.6 Position Estimation by RSSI Measurements

Section 3.3 describes a method of estimating the position by using RSSI values of probe requests captured by at least four capture devices and using a simple path loss model. According to this model, steady changes in the position of the sender result in steady estimated position changes. The following measurement results show that the method in section 3.3 only partly correspodents with the measurement results.

Setup. Table 4.1 and Figure 4.5 show the measurement results. The raw data for each captured and aggregated probe request frame can be found in section A. The setup is placed in the area under the open sky in the center of a U-shaped building. We use four capture devices – labeled with the *AP* prefix – with fixed positions near the corners of the measurement area with a capture software setup as described in section 3. During measurement line of sight was ensured between the probe sending device and all four capturing APs. The big colored circles in the measurement area labeled with an "M" prefix and a number are the known positions of the mobile device sending probe requests. The small colored and slightly transparent cycles are the position estimations where the number references measurement position index of the probe sending device. At each position a set of 10 probe requests that were received by all four APs was used to estimate the position.

Findings and Speculations. Due to lack of available measurement equipment for more detailed RF measurement the following findings are just speculations without proper verification by measurements or more advanced simulations.

- The minimum average distance between the actual and the estimated position is of more than 7 meters quite high.
- With some exceptions and in comparison to the high average distance the standard deviation of most the sender positions are interestingly low.

Both findings in statistical numbers can be found in the map visualization too: Clusters of small spread estimated positions are some distance away from the actual position. This leads to the assumption that there is some systematic error in the model used to estimate the positions:

Figure 4.5: Position Estimation by RSSI Measurements – Result Visualization

- As mentioned by the authors proposing the position estimation method used in this work, there is a systematic error due to the linearization. But it only explains a single digit percentage distance error for the four measurement positions at the center.

- As the maximum distance between the APs and the probe sending device with approximately 35 meters is nowhere near the limit of WLAN range and the RSSI value delivered by the capture devices driver is only reported in integer steps, these integer steps may lead to some notable systematic error.

- As the test area is far from an ideal anechoic chamber, reflections and multipath effects may lead to notable systematic error.

- Some small metallic obstacles like sheet metal wrapped ventilation shafts (M05-

Measurement Point #	Avg. Distance	Std. Deviation
M01	8.28 m	1.59 m
M02	7.06 m	0.76 m
M03	10.42 m	5.64 m
M04	8.56 m	4.41 m
M05	18.00 m	0.46 m
M06	14.04 m	0.68 m
M07	12.93 m	4.45 m
M08	16.65 m	1.33 m
M09	12.24 m	2.20 m
M10	11.45 m	0.88 m
M11	20.38 m	5.99 m
M12	19.92 m	1.94 m
All	13.33 m	5.32 m

Table 4.1: Position Estimation by RSSI Measurements Result – Statistics

M10) and a stair railing (connecting M01-M04) which parts have a big enough distance to be penetratable by RF waves in the 2.4 GHz band may have some effect on the measurement results. In addition the test area environment is not under complete control and some reflection characteristics may have changed by opened or closed windows or window blinds during the measurement period.

- The position estimation method comes with the assumption of isotropic radiation and reception of RF waves. In real world scenarios with off the shelf consumer devices this assumption is only an approximation. The direction of the only moving device, the probe sending device, was not changed during the measurement run, nevertheless the angle to the different capture devices changes with the position of the probe sending device.

5 Extensions and Future Work

5.1 Advanced Device Fingerprinting

This work mainly focuses on the use of transmitter MAC addresses in probe request frames, easy to capture and extract WPS data in probe requests and IEEE 802.1X data in EAPoL packets to perform tracking of WLAN client devices. These are baseline device fingerprinting methods for more elaborate and advanced tracking techniques which are subject to future work.

OSI Layer 1 Based Methods. Methods described in this work mainly focus on protocols which work on or related to OSI Layer 2. There are well-known fingerprinting methods that work on OSI Layer 1. Physical wireless transmitter and receiver components consists of RF, mixed signal and analog parts which are prone to manufacturing tolerances. Important parts of such physical layer components are the baseband oscillators for transmitter und receiver components. Section 17.3.9.5 *Transmit center frequency tolerance* in IEEE 802.11 specifies an allowed tolerance of ±20 ppm for 20 MHz channels. Doppler frequency shifts caused by moving client devices worn by walking persons are usually more than one decade smaller. [32] shows the possibility of differentiating between more than 130 identical 802.11 NICs with an accuracy in excess of 99%. A more general listing of possible OSI Layer 1 fingerprinting methods can be found in [33]. Compared to the method used in this work, Layer 1 methods typically require expensive specialized hardware and measurement devices like vector signal analyzers or software defined radios.

Other Fingerprinting Methods. Driver and operating system version dependent variations may be detected by taking a look at malformed frames processing behavior [34] and statistic analysis of probe request bursts [35] and other timing behavior [36].

5.2 Tracking

Similar to the fingerprinting methods, tracking methods can be extended.

Statistic Methods. [37] shows approaches and an implementation on how to merge short tracks with a certain possibility to longer pieces using hidden Markov models.

Tracking using other Data. [38] tries to classify device types by statistically analyzing traffic usage patterns with simple heuristics. Compared to the data used in our work these approaches require a significantly bigger amount of data which includes actual partly encrypted payload data.

Active Tracking Approaches. By extending fingerprinting and tracking from passive data collection to actively interacting with the client device which uses randomized MAC addresses, a big amount of devices may be trackable once the devices non-randomized MAC address is known [39]. Actively sending request to send (RTS) frames from a randomly generated source MAC address to the non-randomized MAC address of devices will let some devices send clear to send (CTS) frames in response to the RTS frame, even if the device is currently using randomized MAC addresses for sending other types of frames [40].

Commercial Tracking Systems. Notable AP and network equipment manufacturers have implemented location estimation and tracking capabilities into their product lines allowing targeted retail stores to track and analyze customer behavior to aggregate data for business intelligence decisions.
Aruba Networks as a subsidiary of Hewlett Packard Enterprise integrates analytics capabilities into their AP series and provides a dedicated central software application called *Analytics and Location Engine* collecting the data of an WLAN deployment [41]. It is designed to collect the location data of client devices and transmitting them to third party analytic applications for further analysis and deviation of business intelligence decisions.
Cisco Meraki offers a similar product called *Location Analytics* which integrates data analytic and aggregation capabilities of data from other products lines of the company [42].

6 Conclusion

In this work we have introduced general privacy related data in WLAN probe request frames sent by actively probing client devices. This data includes lists of stored and previously connected SSIDs, device details including exact model names in the WPS extension of probe request frames and outer identities used to connect to a WPA2/3-Enterprise protected WLAN. To raise awareness and show easy capture possibilities of this data using off the shelf hardware, a prototype extracting and aggregating these information was implemented. In addition to pure data extraction and aggregation a position estimation algorithm using a multilateration approach was added to the capture system.

6.1 Current Developments

MAC address randomization is the main measure against easy trackability that is implemented by all major operating system vendors now. In Android 10 as well as in iOS 14 MAC address randomization is enabled by default for probe requests and for newly connected WLANs. iOS 14 devices automatically use a unique MAC address per SSID for all saved SSIDs by default while Android 10 only generates a unique MAC address per SSID for newly seen and connected SSIDs with the option of manually enabling unique MAC addresses for previously saved SSID connection information.
With Windows Version 2004 Microsoft added GUI options to manually specify an outer identity when connecting to a WLAN protected with WPA2/3-Enterprise.
Manufacturers are clearly aware of the privacy implications of trackable client devices and have implemented countermeasures. Besides MAC address randomizations other sensitive fields in probe requests were removed in current implementations.
Other implementation parts still leave room for improvement. Implementations of WPA2/3-Enterprise for PEAP-MSCHAPv2 still compromise privacy by leaking the username in the outer tunnel of the RADIUS protocol by default on the major client implementations of Android 11, iOS 14 and Windows 10 20H2. iOS 14 does not offer a GUI configuration option to configure the outer identity during associating with an AP. This can only be achieved via Apples Mobile Device Management solution typically used in enterprises. For bring your own device scenarios this is still an issue.

6.2 Awareness and Tracking Prevention

A big part of privacy issues of the original implementations is already fixed by the manufacturers. To prevent leakage of personal data it is important to be aware of possible data leakage and to manually check configuration options if not set to reasonable values by default. For the big majority of users, privacy protection mechanisms will only be applied if enabled by default by the operating system or device vendor.

Configuration Recommendations. Aware users can respect the following points to improve privacy and make tracking more difficult:

- If previously saved user credentials for a WLAN are not needed any more, deleting the profile prevents possible leakage of the SSID.
- Check if MAC address randomization is supported and enabled.
- When connecting to a WLAN protected by WPA2/3-Enterprise with PEAP-MSCHAPv2 check if the outer identity is needed at all or can be set to a non-critical string like *anonymous*. For the Eduroam deployment at the JKU setting it to `@jku.at` is sufficient for non-exchange students.
- As WPS is rarely needed on mobile devices, completely deactivating it prevents possible leakage of device details and the possibly long time constant and trackable *UUID-E* field sent with each probe request.
- Try to prevent easy tracking possibility with the help of other RF technologies. E.g. non-randomized Bluetooth MAC addresses can be captured and tracked similar to the ones of WLANs.
- Disabling RF technologies in the operating system options when not needed may help to reduce data leakage but does not guarantee to disable the WLAN radio completely. E.g. some versions of Android actively probe for APs to support their location service accuracy inside buildings where GPS is not a good option even if WLAN is disabled in the operating system settings. In an unmodified Android 10 this background scanning can be disabled in the location settings.

A Raw Measurement Result Data

Table A.1: Measurement AP Positions

AP #	X Coordinate	Y Coordinate
AP1	0.00 m	0.00 m
AP2	-0.15 m	25.00 m
AP3	17.95 m	25.56 m
AP4	27.26 m	-1.11 m

Table A.2: Measurement Probe Sending Device Positions

Position #	X Coordinate	Y Coordinate
M01	8.79 m	3.11 m
M02	8.13 m	13.67 m
M03	12.19 m	15.22 m
M04	16.32 m	3.11 m
M05	1.85 m	19.11 m
M06	1.99 m	10.22 m
M07	2.22 m	1.33 m
M08	17.43 m	20.33 m
M09	19.80 m	13.89 m
M10	22.01 m	7.22 m
M11	4.28 m	-5.89 m
M12	19.35 m	-4.56 m

Table A.3: Estimated Device Positions

Measurement Point #	X Coordinate	Y Coordinate
01	11.88 m	12.57 m
01	9.41 m	8.79 m
01	10.64 m	10.69 m
01	11.38 m	10.37 m
01	10.93 m	11.41 m
01	9.38 m	9.51 m
01	12.94 m	12.05 m
01	10.20 m	10.15 m
01	15.08 m	11.55 m
01	14.08 m	10.53 m
02	14.97 m	9.56 m

Continued on next page

Table A.3 – *Continued from previous page*

Measurement Point #	X Coordinate	Y Coordinate
02	14.60 m	10.96 m
02	15.50 m	11.85 m
02	15.24 m	11.37 m
02	15.02 m	10.54 m
02	14.87 m	11.78 m
02	8.61 m	8.44 m
02	14.10 m	11.10 m
02	14.83 m	11.00 m
02	14.71 m	11.29 m
03	16.59 m	12.60 m
03	10.94 m	9.20 m
03	9.44 m	7.63 m
03	10.67 m	8.06 m
03	-1.49 m	-3.84 m
03	5.33 m	3.31 m
03	9.69 m	-0.36 m
03	9.44 m	7.63 m
03	11.38 m	7.00 m
03	11.38 m	7.00 m
04	10.94 m	9.20 m
04	10.96 m	7.14 m
04	9.44 m	7.63 m
04	12.15 m	6.76 m
04	10.67 m	8.06 m
04	9.49 m	6.14 m
04	9.41 m	8.79 m
04	10.95 m	8.15 m
04	-0.11 m	-9.38 m
04	15.63 m	8.10 m
05	16.48 m	8.17 m
05	16.98 m	9.05 m
05	16.48 m	8.17 m
05	16.53 m	8.81 m
05	16.53 m	8.81 m
05	16.15 m	9.43 m
05	16.03 m	8.55 m
05	16.53 m	8.81 m
05	17.11 m	7.90 m
05	16.11 m	8.88 m
06	15.67 m	8.67 m
06	15.67 m	8.67 m
06	15.67 m	8.67 m
06	17.11 m	7.90 m
06	15.67 m	8.67 m

Continued on next page

Table A.3 – *Continued from previous page*

Measurement Point #	X Coordinate	Y Coordinate
06	14.88 m	9.09 m
06	16.98 m	9.05 m
06	16.11 m	8.88 m
06	15.98 m	7.86 m
06	15.67 m	8.67 m
07	16.48 m	8.17 m
07	16.11 m	8.88 m
07	18.72 m	7.96 m
07	16.15 m	9.43 m
07	16.93 m	8.44 m
07	16.11 m	8.88 m
07	9.43 m	8.11 m
07	6.57 m	5.63 m
07	8.40 m	5.77 m
07	8.36 m	6.49 m
08	8.31 m	5.60 m
08	7.02 m	6.78 m
08	6.57 m	5.63 m
08	8.36 m	6.49 m
08	9.46 m	7.03 m
08	7.22 m	6.85 m
08	9.43 m	8.17 m
08	9.44 m	7.63 m
08	8.36 m	6.49 m
08	6.22 m	5.39 m
09	12.92 m	11.28 m
09	8.55 m	7.32 m
09	8.36 m	6.49 m
09	6.57 m	5.63 m
09	8.48 m	5.91 m
09	9.49 m	6.01 m
09	10.11 m	8.33 m
09	10.31 m	6.79 m
09	10.63 m	7.64 m
09	10.21 m	6.88 m
10	11.22 m	7.84 m
10	11.14 m	7.84 m
10	9.51 m	5.59 m
10	10.21 m	6.88 m
10	9.52 m	5.23 m
10	10.80 m	7.64 m
10	10.45 m	5.43 m
10	12.36 m	8.20 m
10	10.74 m	7.21 m

Continued on next page

Table A.3 – *Continued from previous page*

Measurement Point #	**X Coordinate**	**Y Coordinate**
10	10.21 m	6.88 m
11	9.48 m	6.43 m
11	10.74 m	7.64 m
11	10.74 m	7.64 m
11	11.30 m	7.85 m
11	11.38 m	7.65 m
11	18.32 m	15.54 m
11	19.14 m	16.55 m
11	18.48 m	17.11 m
11	19.14 m	16.55 m
11	15.83 m	14.33 m
12	15.77 m	13.45 m
12	16.49 m	15.03 m
12	18.64 m	18.75 m
12	17.79 m	15.48 m
12	17.79 m	15.48 m
12	16.01 m	13.90 m
12	18.64 m	18.75 m
12	12.65 m	11.77 m
12	15.50 m	13.98 m
12	15.50 m	13.98 m

Bibliography

[1] IEEE, "802.11-2016 - IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE, Standard, Dec. 2016.

[2] ——, "802.11i-2004 - IEEE Standard for information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements," IEEE, Standard, Jul. 2004.

[3] M. Vanhoef and F. Piessens, "Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. New York, NY, USA: ACM, 2017, pp. 1313–1328.

[4] Wi-Fi Alliance, "Wi-Fi Protected Setup Specification Version 2.0.8," https://www.wi-fi.org/downloads-registered-guest/Wi-Fi_Protected_Setup_Specification_v2.0.8.pdf/35517, 2020, accessed: 2020-12-24.

[5] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, and D. Brown, "A Study of MAC Address Randomization in Mobile Devices and When it Fails," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, pp. 365–383, Oct. 2017.

[6] S. Brenza, A. Pawlowski, and C. Pöpper, "A Practical Investigation of Identity Theft Vulnerabilities in Eduroam," in *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec '15)*. New York, NY, USA: ACM, 2015, pp. 14:1–14:11.

[7] JKU Information Management, "Campus Wireless LAN," https://help.jku.at/im/netzwerkzugang/campus-wireless-lan, accessed: 2021-01-07.

[8] L. Coelho, E. Grumbach, and K. Valo, "en:users:drivers:rt2800pci [Linux Wireless]," https://wireless.wiki.kernel.org/en/users/Drivers/rt2800pci, Jul. 2017, accessed: 2021-01-07.

[9] G. Simmons, J. P. Giraud, B. Torracca, and B. Hutchings, "rt2800pci - Debian

Wiki," https://wiki.debian.org/rt2800pci, Dec. 2015, accessed: 2019-07-16.

[10] J. M. Berg and G. Potter, "Radiotap - Introduction," https://www.radiotap.org/, Aug. 2020, accessed: 2021-01-07.

[11] V. Jacobson, C. Leres, and S. McCanne, "Manpage of TCPDUMP," https://www.tcpdump.org/manpages/tcpdump.1.html, Dec. 2020, accessed: 2021-01-07.

[12] G. Giacobbi, "The GNU Netcat – Official homepage," http://netcat.sourceforge.net/, Nov. 2006, accessed: 2021-01-07.

[13] S. M. Biradar, R. Shekhar, and A. P. Reddy, "Build Minimal Docker Container Using Golang," in *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, Jun. 2018, pp. 1–4.

[14] S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs," *IEEE Internet Computing*, vol. 14, no. 6, pp. 80–83, Nov. 2010.

[15] Wireshark, "Wireshark 2.2.0 Release Notes," https://www.wireshark.org/docs/relnotes/wireshark-2.2.0.html, Sep. 2016, accessed: 2019-07-16.

[16] D. Vasisht, S. Kumar, and D. Katabi, "Decimeter-level Localization with a Single WiFi Access Point," in *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation (NSDI 16)*. Berkeley, CA, USA: USENIX Association, 2016, pp. 165–178.

[17] M. Malajner, P. Planinsic, and D. Gleich, "Angle of Arrival Estimation Using RSSI and Omnidirectional Rotatable Antennas," *IEEE Sensors Journal - IEEE SENS J*, vol. 12, pp. 1950–1957, Nov. 2012.

[18] J. Koo and H. Cha, "Localizing WiFi Access Points Using Signal Strength," *IEEE Communications Letters*, vol. 15, no. 2, pp. 187–189, Feb. 2011.

[19] IEEE Standards Association, "MA-L," https://standards.ieee.org/products-services/regauth/oui/index.html, 2020, accessed: 2020-12-24.

[20] UPC Schweiz, "WI-FREE," https://www.upc.ch/en/support/internet/wi-free/, 2020, accessed: 2020-12-24.

[21] C. Matte and M. Cunche, "Spread of MAC address randomization studied using locally administered MAC addresses use historic," Inria Grenoble Rhône-Alpes, Research Report RR-9142, Jan. 2018, hal-01682363.

[22] Apple Inc., "MAC address randomization," https://support.apple.com/guide/security/mac-address-randomization-secb9cb3140c/web, 2020, accessed: 2020-12-24.

[23] ——, "Use private Wi-Fi addresses in iOS 14, iPadOS 14 and watchOS 7," https://support.apple.com/en-gb/HT211227, 2020, accessed: 2020-12-24.

[24] Google LLC, "Privacy: MAC Randomization," https://source.android.com/devices/tech/connect/wifi-mac-randomization, 2020, accessed: 2020-12-24.

[25] ——, "Wi-Fi Aware," https://source.android.com/devices/tech/connect/wifi-aware, 2020, accessed: 2020-12-24.

[26] Microsoft Corporation, "How to use random hardware addresses," https://support.microsoft.com/en-us/help/4027925/windows-how-and-why-to-use-random-hardware-addresses, 2020, accessed: 2020-12-24.

[27] W. Wang, "WinHEC Shenzhen 2015," in *Wireless Networking in Windows 10*, 2015, accessed: 2020-12-24.

[28] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, "Why MAC Address Randomization is Not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIA CCS '16)*. New York, NY, USA: ACM, May 2016, pp. 413—424.

[29] J. Carneal, M. van Smoorenburg, and The FreeRADIUS server project, "freeradius-server/src/lib/server/auth.c," https://github.com/FreeRADIUS/freeradius-server/blob/584fbbc2d42eab31a9889e7d953a8f3a8b510091/src/lib/server/auth.c#L150, 2020, accessed: 2020-12-24.

[30] GÉANT Association, "eduroam CAT," https://play.google.com/store/apps/details?id=uk.ac.swansea.eduroamcat, 2020, accessed: 2020-12-24.

[31] M. Ohlinger, T. Castaldo, and D. Eby, "Add Wi-Fi settings for Windows 10 and later devices in Intune," https://docs.microsoft.com/en-us/mem/intune/configuration/wi-fi-settings-windows, 2020, accessed: 2020-12-24.

[32] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proceedings of the 14th ACM international conference on Mobile computing and networking - MobiCom '08*. ACM Press, 2008, pp. 116–127.

[33] B. Danev, D. Zanetti, and S. Capkun, "On physical-layer identification of wireless devices," *ACM Computing Surveys*, vol. 45, no. 1, pp. 1–29, Nov. 2012.

[34] S. Bratus, C. Cornelius, D. Kotz, and D. Peebles, "Active Behavioral Fingerprinting of Wireless Devices," in *Proceedings of the First ACM Conference on Wireless Network Security (WiSec '08)*. New York, NY, USA: ACM, 2008, pp. 56—-61.

[35] O. Waltari and J. Kangasharju, "The Wireless Shark: Identifying WiFi Devices Based on Probe Fingerprints," in *Proceedings of the First Workshop on Mobile Data (MobiData '16)*. New York, NY, USA: ACM, 2016, pp. 1—6.

[36] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. Van Randwyk, and D. Sicker, "Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting," in *Proceedings*

of the 15th Conference on USENIX Security Symposium - Volume 15 (USENIX-SS'06). USA: USENIX Association, 2006.

[37] H. Hong, G. D. De Silva, and M. C. Chan, "CrowdProbe: Non-Invasive Crowd Monitoring with Wi-Fi Probe," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 2, no. 3, Sep. 2018.

[38] S. Siby, R. R. Maiti, and N. O. Tippenhauer, "IoTScanner: Detecting Privacy Threats in IoT Neighborhoods," in *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security (IoTPTS '17)*. New York, NY, USA: ACM, 2017, pp. 23—-30.

[39] S. S. Sawwashere and S. U. Nimbhorkar, "Survey of RTS-CTS Attacks in Wireless Network," in *2014 Fourth International Conference on Communication Systems and Network Technologies*. IEEE, Apr. 2014, pp. 752–755.

[40] P. Robyns, B. Bonné, P. Quax, and W. Lamotte, "Noncooperative 802.11 MAC Layer Fingerprinting and Tracking of Mobile Devices," *Security and Communication Networks*, vol. 2017, pp. 1–21, Jan. 2017.

[41] Aruba Networks, "Analytics and Location Engine," https://www.arubanetworks.com/products/location-services/analytics/ale/, 2020, accessed: 2020-12-24.

[42] Cisco Systems, Inc., "Location Analytics," https://meraki.cisco.com/solutions/location_analytics/, 2020, accessed: 2020-12-24.