Otto-Friedrich-University of Bamberg

Chair of Mobile Systems

Master Project Mobile Software Systems
MOBI-PRS-M-2

Summer Term 2023

Evaluating Data Manipulation Threats in WiFi-based People Counting: Risks and Defenses

Project Report

Submitted by:
Syed Ibrahim Khalil (2088815), Ishraq Haider Chowdhury (2029000)

Supervisor: Michael Freitag, Christoph Baum

Bamberg, September 30, 2023
Summer Term 2023

Contents

1 Abstract

The surge in WiFi-based people counting sensors within urban environments has transformed data-driven urban planning. However, this advancement has unveiled vulnerabilities to data manipulation threats, including injection attacks, jeopardizing data integrity and resulting in misleading insights. This report comprehensively examines such risks in WiFi-based people counting systems and offers corresponding defensive strategies. The project comprises both offensive and defensive approaches. It features the vulnerability detection of the sensors and the behavior of the sensors during the injection of spoof MAC addresses, simulating potential adversary attacks. Other potential issues with MAC addresses and sensors are also discussed in the report. In order to avoid these potential attacks by an adversary, a defensive approach introducing an anomaly detection algorithm meticulously scrutinizing incoming JSON sensor data to pinpoint irregularities and potential injection attacks. The study encompasses data acquisition, preprocessing, and the core anomaly detection mechanism, employing the Isolation Forest model. Additionally, we explore alternative models, feature engineering techniques, and long-term system enhancement recommendations. By enabling real-time anomaly detection and secure digital infrastructure integration, our approach guarantees swift alerts and reinforces urban security, elevating smart cities to new heights of resilience. This report acts as a valuable guide for urban planners and administrators striving to protect the reliability of WiFi-based people counting systems.

2 Introduction

WiFi-based people counting sensors have ushered in a new era of data-driven urban planning and management. These sensors, dispersed throughout urban landscapes, collect real-time data on the movements and densities of people, enabling cities to make informed decisions on resource allocation, transportation planning, and infrastructure development. However, this unprecedented access to data also exposes these systems to potential data manipulation threats, which could undermine the very foundation of smart cities.

Data manipulation threats, including injection attacks, have become a critical concern in data security. Injection attacks involve the illicit insertion of untrusted or rogue data into a system. These attacks can yield misleading insights, disrupt system operations, and compromise the reliability of data-driven decision-making.

To address these pressing concerns, the offensive approach of this project tests out and simulates the behavior of an adversary's attacks and finds a zero-day vulnerability. The aim was also to test the sensor's behavior while spoofing and injecting `Media Access Control Address (MAC)`. On the defensive approach, an anomaly detection algorithm is designed to rigorously scrutinize incoming JSON data from WiFi sensors. This algorithm forms the core of our defensive strategy, enabling us to swiftly identify and mitigate potential threats, potentially ensuring that the data we rely on remains uncompromised, fostering high trust in our digital infrastructure.

3 Offensive Part

Contribution - Syed Ibrahim Khalil

Introduction: The “Offensive Part” aims to shed light on the system’s infrastructure and how potential attacks could be performed to simulate an adversary approach. These attacks were meant to analyze the behavior of the sensors and whether or not false data could be injected into the database. The future goal of finding these vulnerabilities is to create a defensive approach in parallel in order to fill in the gaps of these security issues. It is important to state that this approach nor the defensive approach is final, but continuous security validation is supposed to happen in the future evolution of this project.

To further approach the strategies, results, and findings, it would be vital to discuss some of the main concepts that are essential to know for the project of “WiFi-based People Counting” project.

3.1 Probe Requests, MACs, and Limitations

3.1.1 Probe Requests

A *probe request* is a Wi-Fi-based communication packet sent by devices (known as “client devices”) such as laptop, smartphones, and other devices which has the capability of Wi-Fi in order to discover possible Wi-Fi networks in the vicinity to connect. A device sends a probe request when it wakes up from stand-by or sleep mode or when a user actively scans for available Wi-Fi `Access Points (AP)`. A device would also send out a probe burst consisting of probe requests, each containing the `Service Set Identifier (SSID)` of previously known networks, in order to get connected with them, shown in figure 1.

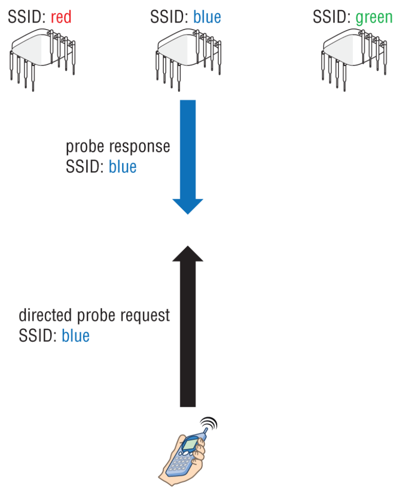


Figure 1: Directed Probe Request[1]

The other type of probe request is with a wildcard SSID, which sets the SSID to “null”, shown in figure 2.

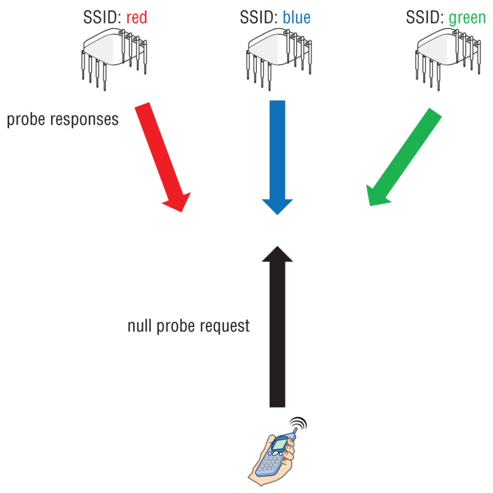


Figure 2: Null Probe Request[1]

The methodology of sending probe requests to AP is to get back the probe responses, which include information about the available network, AP's own SSID, and security protocols. The client device (user device) uses this probe response to decide which network address to connect.

In short, the probe request aims to discover the Wi-Fi networks available to connect efficiently and quickly.

3.1.2 MAC & Its Significance

The general concept of Wi-Fi based people counting is to utilize the MAC address inside the probe request frame to count the number of people in a region. Apart from the theoretical aspect of MAC addresses, the important thing to discuss here is the usage of MAC addresses. MAC addresses are unique to each device, which makes it an important aspect concerning counting the number of people based on;

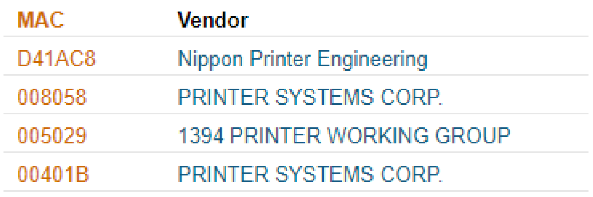
$$1 \quad MAC\ address = 1 \quad Device = 1 \quad Person$$

3.1.3 Limitations & Some Potential Solutions

Despite the potential of using MAC addresses, some limitations still need to be shed light upon.

Single Person Multiple Devices: It is essential to mention here that at this stage, other devices, such as smartwatches and a single person with multiple devices, are not considered in counting the number of people. As the project progresses, this aspect must be considered for more accuracy. It has to be further researched to consider any potential approach, but based on the current knowledge, it would be possible to consider other frames of a probe request.

Printers & Other Devices: As some devices, such as printers and other potential devices, also send out probe requests, it could be considered in the people counting. To avoid such a potential solution is to create such filters in the sensors that could discard all those MAC addresses which have `Organizationally Unique Identifier (OUI)`, the first three octets of a printer company or for all those devices that need to be discarded before sending out to the database. An example of such vendor-specific MAC addresses is shown in figure 3.

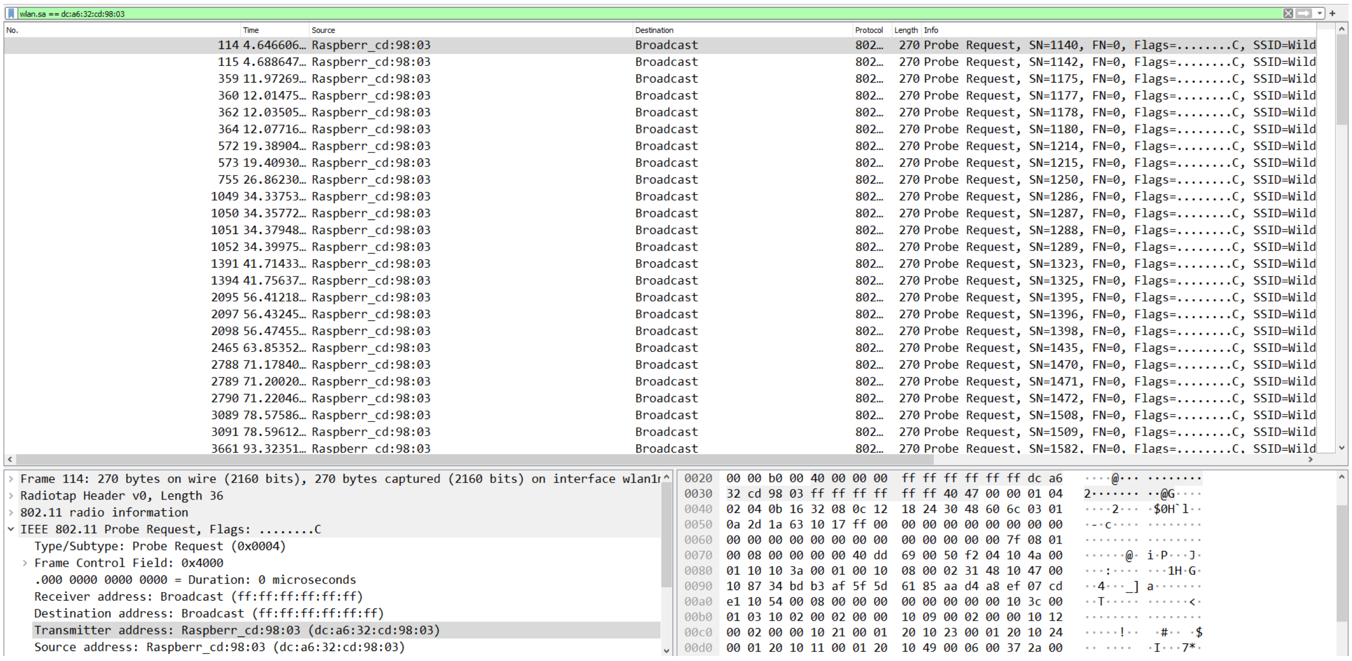


MAC	Vendor
D41AC8	Nippon Printer Engineering
008058	PRINTER SYSTEMS CORP.
005029	1394 PRINTER WORKING GROUP
00401B	PRINTER SYSTEMS CORP.

Figure 3: Vendor-Specific Printer Companies

MAC Randomization: Although a good privacy-oriented approach, this brings limitations when the only thing the project relies on, MAC Address, is getting randomized, resulting in inaccuracies. As of now, a newer version of Android device[2], Apple devices[3], and Windows devices[4] has the functionality of using the MAC randomization; this could also lead to inaccuracy for the people-based counting. As newer devices are also decreasing the use of SSID in a probe request[5], utilizing SSID would not be a vital approach to prevent the issue mentioned about MAC randomization.

Security issue with the Sensor's Network Packets: One of the security issues during the project was that the probe requests and other network packets sent by the sensor, including the MAC address and other Wi-Fi management properties, are visible in monitoring the network traffic. Although the potential threats still need to be discovered, in general, this exposes the sensor's location and the MAC address, which an adversary could utilize to create exploits in the system. Figure 4 shows the captured probe requests of the sensor, while figure 5 shows other network packets of the same sensor.

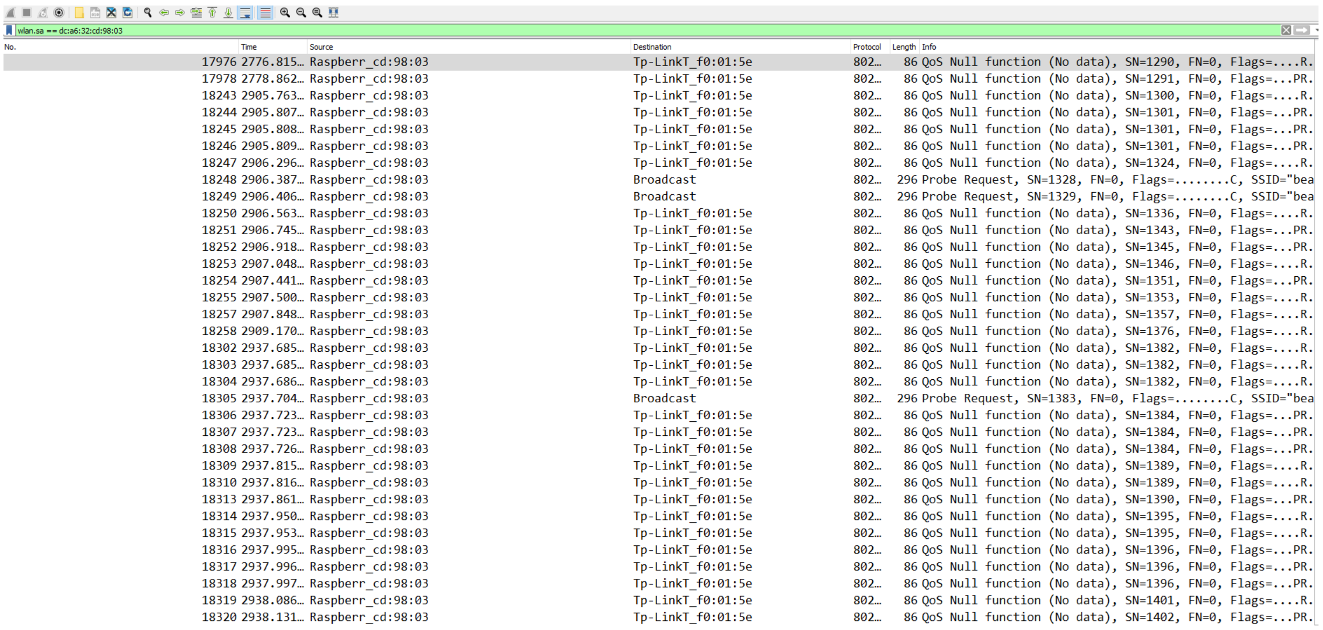


```
wlan.sa == dc:a6:32:cd:98:03
No.      Time          Source                Destination   Protocol Length Info
 114 4.646606… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1140, FN=0, Flags=........C, SSID=Wild
 115 4.688647… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1142, FN=0, Flags=........C, SSID=Wild
 359 11.97269… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1175, FN=0, Flags=........C, SSID=Wild
 360 12.01475… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1177, FN=0, Flags=........C, SSID=Wild
 362 12.03505… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1178, FN=0, Flags=........C, SSID=Wild
 364 12.07716… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1180, FN=0, Flags=........C, SSID=Wild
 572 19.38904… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1214, FN=0, Flags=........C, SSID=Wild
 573 19.40930… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1215, FN=0, Flags=........C, SSID=Wild
 755 26.86230… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1250, FN=0, Flags=........C, SSID=Wild
1049 34.33753… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1286, FN=0, Flags=........C, SSID=Wild
1050 34.35772… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1287, FN=0, Flags=........C, SSID=Wild
1051 34.37948… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1288, FN=0, Flags=........C, SSID=Wild
1052 34.39975… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1289, FN=0, Flags=........C, SSID=Wild
1391 41.71433… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1323, FN=0, Flags=........C, SSID=Wild
1394 41.75637… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1325, FN=0, Flags=........C, SSID=Wild
2095 56.41218… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1395, FN=0, Flags=........C, SSID=Wild
2097 56.43245… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1396, FN=0, Flags=........C, SSID=Wild
2098 56.47455… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1398, FN=0, Flags=........C, SSID=Wild
2465 63.85352… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1435, FN=0, Flags=........C, SSID=Wild
2788 71.17840… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1470, FN=0, Flags=........C, SSID=Wild
2789 71.20020… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1471, FN=0, Flags=........C, SSID=Wild
2790 71.22046… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1472, FN=0, Flags=........C, SSID=Wild
3089 78.57586… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1508, FN=0, Flags=........C, SSID=Wild
3091 78.59612… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1509, FN=0, Flags=........C, SSID=Wild
3661 93.32351… Raspberr_cd:98:03  Broadcast  802…  270 Probe Request, SN=1582, FN=0, Flags=........C, SSID=Wild
```

```
> Frame 114: 270 bytes on wire (2160 bits), 270 bytes captured (2160 bits) on interface wlan1r
> Radiotap Header v0, Length 36
> 802.11 radio information
v IEEE 802.11 Probe Request, Flags: ........C
    Type/Subtype: Probe Request (0x0004)
  > Frame Control Field: 0x4000
    .000 0000 0000 0000 = Duration: 0 microseconds
    Receiver address: Broadcast (ff:ff:ff:ff:ff:ff)
    Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
    Transmitter address: Raspberr_cd:98:03 (dc:a6:32:cd:98:03)
    Source address: Raspberr_cd:98:03 (dc:a6:32:cd:98:03)
```

```
0020  00 00 b0 00 40 00 00 00  ff ff ff ff ff ff dc a6   ····@··· ········
0030  32 cd 98 03 ff ff ff ff  ff ff 40 47 00 00 01 04   2······· ··@G····
0040  02 04 0b 16 32 08 0c 12  18 24 30 48 60 6c 03 01   ····2··· ·$0H`l··
0050  0a 2d 1a 63 10 17 ff 00  00 00 00 00 00 00 00 00   ·-·c···· ········
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 7f 08 01   ········ ········
0070  00 08 00 00 00 00 40 dd  69 00 50 f2 04 10 4a 00   ······@· i·P···J·
0080  01 10 10 3a 00 01 00 10  08 00 02 31 48 10 47 00   ···:···· ···1H·G·
0090  10 87 34 bd b3 af 5f 5d  61 85 aa d4 a8 ef 07 cd   ··4···_] a·······
00a0  e1 10 54 00 08 00 00 00  00 00 00 00 00 10 3c 00   ··T····· ······<·
00b0  01 03 10 02 00 02 00 00  10 09 00 02 00 00 10 12   ········ ········
00c0  00 02 00 00 10 21 00 01  20 10 23 00 01 20 10 24   ·····!··  ·#·· ·$
00d0  00 01 20 10 11 00 01 20  10 49 00 06 00 37 2a 00   ·· ····  ·I···7*·
```

Figure 4: Probe Requests of the Sensor



```
wlan.sa == dc:a6:32:cd:98:03
No.        Time          Source              Destination        Protocol Length Info
17976 2776.815… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1290, FN=0, Flags=....R.
17978 2778.862… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1291, FN=0, Flags=...PR.
18243 2905.763… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1300, FN=0, Flags=....R.
18244 2905.807… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1301, FN=0, Flags=...PR.
18245 2905.808… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1301, FN=0, Flags=...PR.
18246 2905.809… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1301, FN=0, Flags=...PR.
18247 2906.296… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1324, FN=0, Flags=....R.
18248 2906.387… Raspberr_cd:98:03  Broadcast          802…  296 Probe Request, SN=1328, FN=0, Flags=........C, SSID="bea
18249 2906.406… Raspberr_cd:98:03  Broadcast          802…  296 Probe Request, SN=1329, FN=0, Flags=........C, SSID="bea
18250 2906.563… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1336, FN=0, Flags=....R.
18251 2906.745… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1343, FN=0, Flags=...PR.
18252 2906.918… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1345, FN=0, Flags=...PR.
18253 2907.048… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1346, FN=0, Flags=....R.
18254 2907.441… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1351, FN=0, Flags=...PR.
18255 2907.500… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1353, FN=0, Flags=....R.
18257 2907.848… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1357, FN=0, Flags=....R.
18258 2909.170… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1376, FN=0, Flags=....R.
18302 2937.685… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1382, FN=0, Flags=....R.
18303 2937.685… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1382, FN=0, Flags=....R.
18304 2937.686… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1382, FN=0, Flags=....R.
18305 2937.704… Raspberr_cd:98:03  Broadcast          802…  296 Probe Request, SN=1383, FN=0, Flags=........C, SSID="bea
18306 2937.723… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1384, FN=0, Flags=...PR.
18307 2937.723… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1384, FN=0, Flags=...PR.
18308 2937.726… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1384, FN=0, Flags=...PR.
18309 2937.815… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1389, FN=0, Flags=....R.
18310 2937.816… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1389, FN=0, Flags=....R.
18313 2937.861… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1390, FN=0, Flags=...PR.
18314 2937.950… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1395, FN=0, Flags=....R.
18315 2937.953… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1395, FN=0, Flags=....R.
18316 2937.995… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1396, FN=0, Flags=...PR.
18317 2937.996… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1396, FN=0, Flags=...PR.
18318 2937.997… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1396, FN=0, Flags=...PR.
18319 2938.086… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1401, FN=0, Flags=....R.
18320 2938.131… Raspberr_cd:98:03  Tp-LinkT_f0:01:5e  802…   86 QoS Null function (No data), SN=1402, FN=0, Flags=...PR.
```

Figure 5: Other Network Packets of the Sensor

The potential solution to avoiding this situation is to mask the MAC address of the sensor with a random MAC address, possibly a frequent change, which is viable for security purposes.

3.2 Methodology

The motivation to perform the spoof MAC address injection attack was driven by an artist based in Berlin, who Used 99 phones to create a fake traffic jam in Google Maps[6].

A similar approach is considered to test the system. The methodology of creating such an approach is discussed in the following.

3.2.1 Generating MAC Addresses

In order to proceed with the injection attack, a spoof MAC addresses generator is needed; thus, a shell script (MAC_Generator.sh) is created to generate as many MAC addresses as a user wants. This helps create a large number of MAC addresses, which could be further used while automatically sending out probe requests with a new MAC address every time. While using this script, the Wi-Fi adapter should not be connected to any wireless network to function correctly. The script generates a text file containing MAC addresses and a .CSV file, which includes all MAC addresses with their timestamp when they were created.

To run the script, the following commands are needed;

```
sudo chmod +x ./MAC_Generator.sh
sudo ./MAC_Generator.sh
```

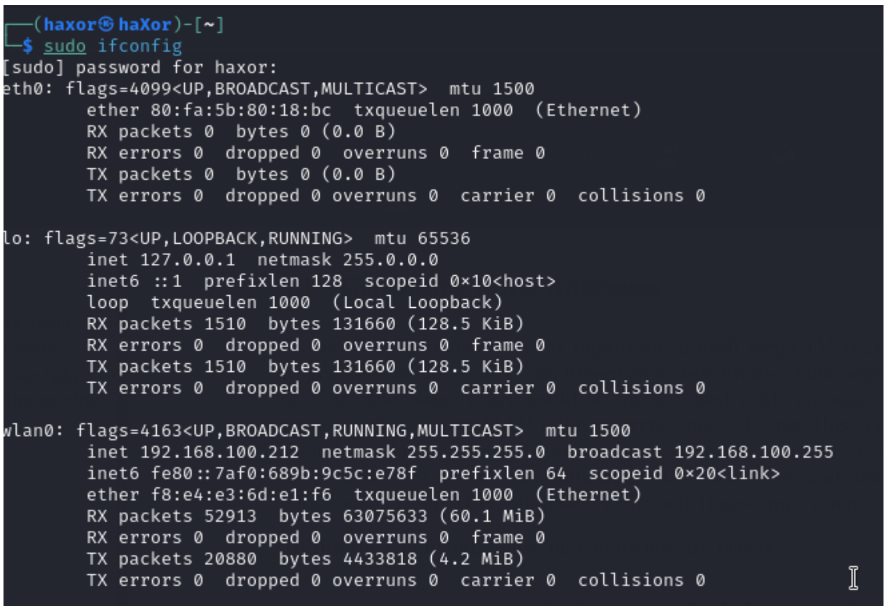
3.2.2 Wireless Adapter -Monitor Mode

In order to send out probe requests, it is a prerequisite to set up the wireless adapter or the built-in wireless card to monitor mode. Some laptop devices have a built-in wireless card with monitor mode functionality, which could also be used. However, advanced attacks like Scapy-based scripts need "packet injection" supported cards, such as `"Alfa AWUS036NHA"`[7]. It is better to use two adapters, one for separately monitoring the network packets and one adapter for packet injection, to analyze the data later.

To enable the monitor mode, the following commands are needed;

```
sudo airmon-ng start *interface name eg. wlan0*
```

Figure 6 shows the available interfaces, while figure 7 shows the output after enabling the monitor mode.

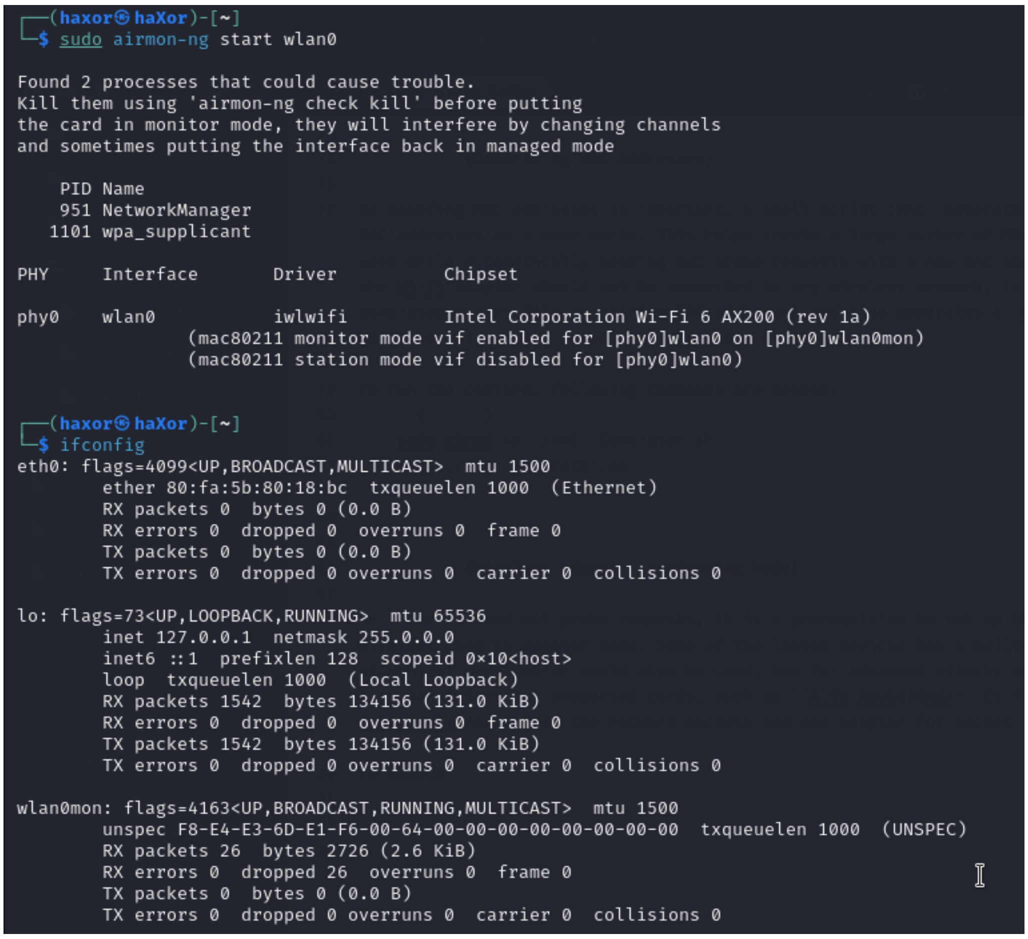


```
┌──(haxor㉿haXor)-[~]
└─$ sudo ifconfig
[sudo] password for haxor:
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 80:fa:5b:80:18:bc  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 1510  bytes 131660 (128.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1510  bytes 131660 (128.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.212  netmask 255.255.255.0  broadcast 192.168.100.255
        inet6 fe80::7af0:689b:9c5c:e78f  prefixlen 64  scopeid 0x20<link>
        ether f8:e4:e3:6d:e1:f6  txqueuelen 1000  (Ethernet)
        RX packets 52913  bytes 63075633 (60.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 20880  bytes 4433818 (4.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 6: Available Interfaces



```
┌──(haxor㉿haXor)-[~]
└─$ sudo airmon-ng start wlan0

Found 2 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

    PID Name
    951 NetworkManager
   1101 wpa_supplicant

PHY     Interface       Driver          Chipset

phy0    wlan0           iwlwifi         Intel Corporation Wi-Fi 6 AX200 (rev 1a)
                (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
                (mac80211 station mode vif disabled for [phy0]wlan0)


┌──(haxor㉿haXor)-[~]
└─$ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 80:fa:5b:80:18:bc  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 1542  bytes 134156 (131.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1542  bytes 134156 (131.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0mon: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        unspec F8-E4-E3-6D-E1-F6-00-64-00-00-00-00-00-00-00-00  txqueuelen 1000  (UNSPEC)
        RX packets 26  bytes 2726 (2.6 KiB)
        RX errors 0  dropped 26  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 7: Enabling Monitor Mode

3.2.3 Attacking Scripts

In order to create a script to send out probe requests, three different types of scripts were created to test out different possibilities.

Macchanger Scripts: The first two scripts are similar, which use the "Macchanger," a tool in Kali Linux (Linux tool)[8] in real time to change the interface MAC address and scans the available networks, which ultimately sends out a probe request. The benefit of this simple approach is that it could utilize a built-in Wi-Fi card to send out probe requests without needing a specific Wi-Fi adapter, but compared to Wi-Fi adapters, it is slower. Both scripts work properly when there is no active Wi-Fi connection established.

The difference between the two scripts is that one generates a user-specific random number of MAC addresses during the runtime, while the other uses saved MAC addresses from a text file. It makes it possible to use an actual device's MAC address or to create a replay attack -based on capturing real MAC addresses from a crowded place, like a concert or a football game.

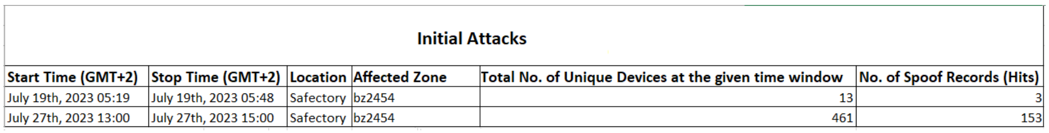
Scapy Script: The third script is more powerful than the others in terms of speed and the powerful potential of using Scapy[9]. Scapy can send out multiple packets in mere seconds, making it possible to flood probe requests and simulate `Denial-of-Service (DoS)` attack.

Mini Tools: Some other mini tools (scripts) are also created to help analyze spoof MAC address data. Scripts like -to create `SHA-224` hash of the given MACs in a text file, -to extract MAC addresses (source address) of all probe requests from a Wireshark .pcap file, -for making a PostgreSQL query from hash values of MAC addresses which can help in extracting the data from the database.

3.3 Simulating Attacks

3.3.1 Initial Attacks

To simulate the attack of injecting spoof records into the database, initially, only one location is tested for the successful simulation attack, which happens on the 19th and 27th of July, 2023. The result is shown in figure 8.



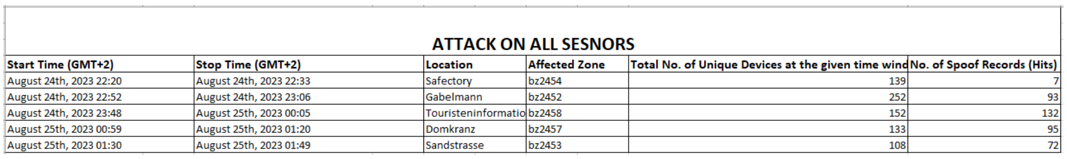
Initial Attacks

Start Time (GMT+2)	Stop Time (GMT+2)	Location	Affected Zone	Total No. of Unique Devices at the given time window	No. of Spoof Records (Hits)
July 19th, 2023 05:19	July 19th, 2023 05:48	Safectory	bz2454	13	3
July 27th, 2023 13:00	July 27th, 2023 15:00	Safectory	bz2454	461	153

Figure 8: Initial Attacks - Logs

3.3.2 Attacking on All Sensors - In One Day

For the final observation of simulating the attack, all sensors were targeted on the same date (24th of August, 2023) with the three attacking scripts. The result is shown in figure 9.

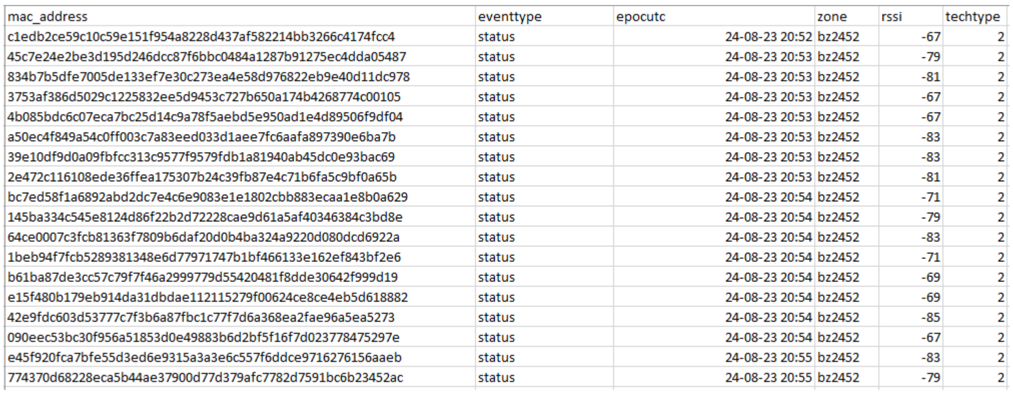


ATTACK ON ALL SESNORS

Start Time (GMT+2)	Stop Time (GMT+2)	Location	Affected Zone	Total No. of Unique Devices at the given time wind	No. of Spoof Records (Hits)
August 24th, 2023 22:20	August 24th, 2023 22:33	Safectory	bz2454	139	7
August 24th, 2023 22:52	August 24th, 2023 23:06	Gabelmann	bz2452	252	93
August 24th, 2023 23:48	August 25th, 2023 00:05	Touristeninformatio	bz2458	152	132
August 25th, 2023 00:59	August 25th, 2023 01:20	Domkranz	bz2457	133	95
August 25th, 2023 01:30	August 25th, 2023 01:49	Sandstrasse	bz2453	108	72

Figure 9: Attack on All Sensors - Logs

An example of the spoof MAC records can also be seen in figure 10, showing spoof MAC records after successfully injecting them. Similarly, other locations' only spoof MAC records data files are collected.



mac_address	eventtype	epocutc	zone	rssi	techtype
c1edb2ce59c10c59e151f954a8228d437af582214bb3266c4174fcc4	status	24-08-23 20:52	bz2452	-67	2
45c7e24e2be3d195d246dcc87f6bbc0484a1287b91275ec4dda05487	status	24-08-23 20:53	bz2452	-79	2
834b7b5dfe7005de133ef7e30c273ea4e58d976822eb9e40d11dc978	status	24-08-23 20:53	bz2452	-81	2
3753af386d5029c1225832ee5d9453c727b650a174b4268774c00105	status	24-08-23 20:53	bz2452	-67	2
4b085bdc6c07eca7bc25d14c9a78f5aebd5e950ad1e4d89506f9df04	status	24-08-23 20:53	bz2452	-67	2
a50ec4f849a54c0ff003c7a83eed033d1aee7fc6aafa897390e6ba7b	status	24-08-23 20:53	bz2452	-83	2
39e10df9d0a09fbfcc313c9577f9579fdb1a81940ab45dc0e93bac69	status	24-08-23 20:53	bz2452	-83	2
2e472c116108ede36ffea175307b24c39fb87e4c71b6fa5c9bf0a65b	status	24-08-23 20:53	bz2452	-81	2
bc7ed58f1a6892abd2dc7e4c6e9083e1e1802cbb883ecaa1e8b0a629	status	24-08-23 20:54	bz2452	-71	2
145ba334c545e8124d86f22b2d72228cae9d61a5af40346384c3bd8e	status	24-08-23 20:54	bz2452	-79	2
64ce0007c3fcb81363f7809b6daf20d0b4ba324a9220d080dcd6922a	status	24-08-23 20:54	bz2452	-83	2
1beb94f7fcb5289381348e6d77971747b1bf466133e162ef843bf2e6	status	24-08-23 20:54	bz2452	-71	2
b61ba87de3cc57c79f7f46a2999779d55420481f8dde30642f999d19	status	24-08-23 20:54	bz2452	-69	2
e15f480b179eb914da31dbdae112115279f00624ce8ce4eb5d618882	status	24-08-23 20:54	bz2452	-69	2
42e9fdc603d53777c7f3b6a87fbc1c77f7d6a368ea2fae96a5ea5273	status	24-08-23 20:54	bz2452	-85	2
090eec53bc30f956a51853d0e49883b6d2bf5f16f7d023778475297e	status	24-08-23 20:54	bz2452	-67	2
e45f920fca7bfe55d3ed6e9315a3a3e6c557f6ddce9716276156aaeb	status	24-08-23 20:55	bz2452	-83	2
774370d68228eca5b44ae37900d77d379afc7782d7591bc6b23452ac	status	24-08-23 20:55	bz2452	-79	2

Figure 10: Spoof MAC records of a certain location

The implications of such records raise intriguing research questions. Even in the case of seemingly arbitrary attacks, anomalies become evident, as illustrated in figure 11. On the right side of the figure, the map displays data without any false information, depicting a location with low crowd density, requiring minimal attention. In contrast, on the left side of the map, the heatmap resulting from false data reveals a shift in the crowd density, turning a previously less crowded location into a crowded one. These observations, though general, depict the significance of accurate data in managing crowd densities and implementing appropriate protocols. However, it is crucial to note that issues arise when such attacks occur unexpectedly.

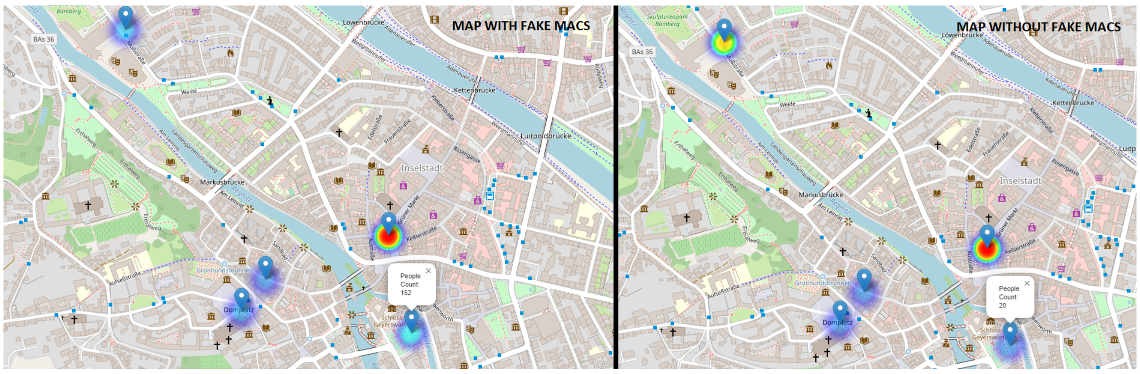


Figure 11: Heatmap Depicting Both Actual & False Records

To summarize, figure 12 shows how this attack affected the sensors' data.

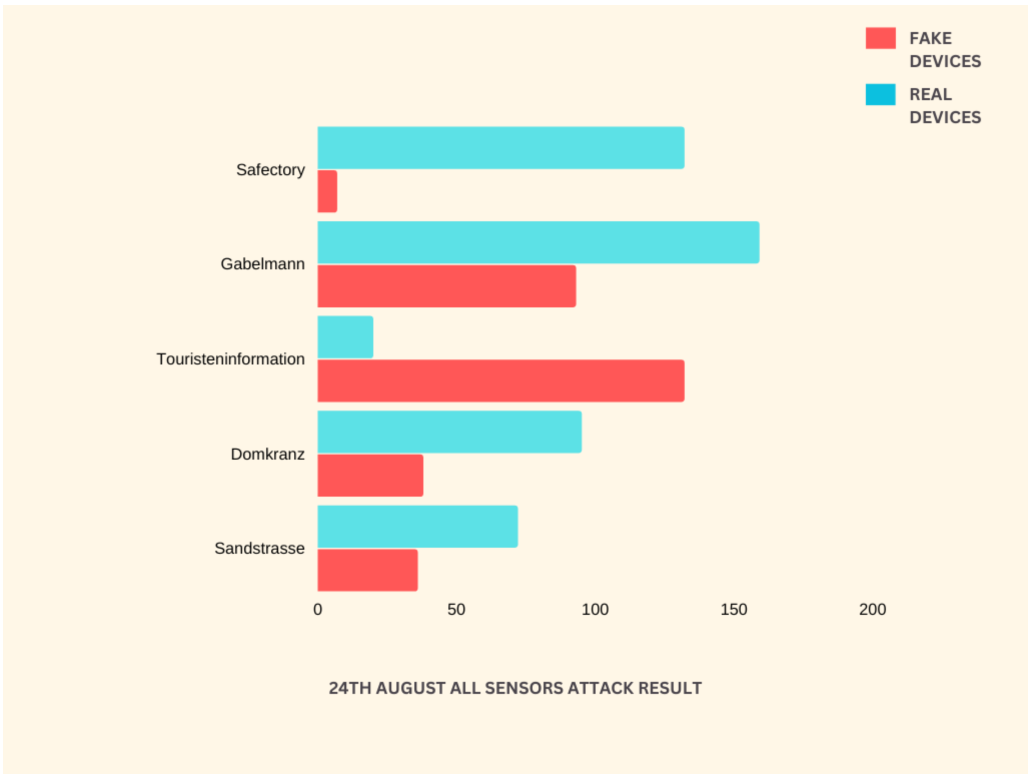


Figure 12: Bar Chart Depicting Sensors' Records

3.4 Challenges Encountered

Simulating an adversary and attacking the sensor seemed simple throughout the project's timeline. However, in reality, it was more challenging than expected. Limited resources and a tight schedule make it difficult to thoroughly test the sensors' functionality before initiating an attack. Apart from these issues, there were also technical issues that were encountered.

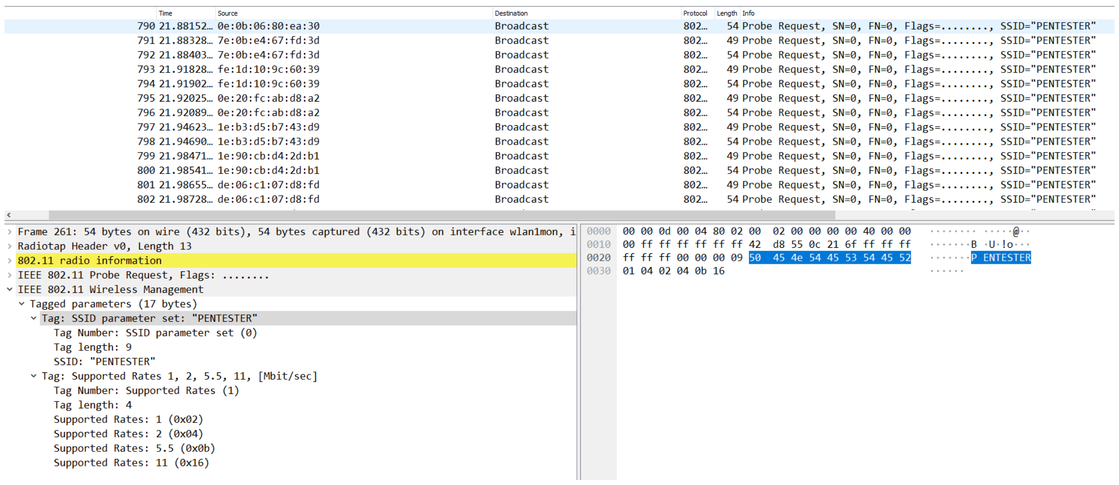
3.4.1 Discrepancy in Packet Transmission

After observing all the successful attacks, including the initial day's attack, it is observed that there is a difference in packet transmission performance. The initial day's attack was performed using a newer laptop equipped with `Intel® Killer™ Wi-Fi 6 AX1650`[10]. The final day attack was performed using an older laptop (due to battery issues) equipped with a basic Wi-Fi card, while the `Alfa AWUS036NHA`[7] adapter was used to monitor the network packets on Wireshark. The older laptop, with a basic Wi-Fi card, encountered issues in sending some packets, leading to an incomplete transmission of data, as opposed to the newer laptop, which features Wi-Fi 6 technology, successfully transmitted all packets without any issues in the initial day's attacks.

Further research is needed to pinpoint the exact cause of the problem. Nonetheless, the appropriate way to inject packets would be to use the `Alfa AWUS036NHA` adapter or a similar capability Wi-Fi adapter, as the scripts for sending probe requests are tested to perform correctly.

3.4.2 Issue With Probe Request Packet Crafting in Scapy

With the Scapy script, crafting precise details for the probe request packet is essential for successful transmission. During the execution of the Scapy script, an issue occurred in the transmission of probe request packets. The data showed that the problem came from incomplete wireless management details within these packets, resulting in transmission failures. Figure 13 displays a Wireshark screenshot illustrating a captured probe request with insufficient wireless management data, highlighting areas for potential improvement.



```
    Time         Source                 Destination   Protocol Length Info
790 21.88152…  0e:0b:06:80:ea:30      Broadcast     802…   54 Probe Request, SN=0, FN=0, Flags=........, SSID="PENTESTER"
791 21.88328…  7e:0b:e4:67:fd:3d      Broadcast     802…   49 Probe Request, SN=0, FN=0, Flags=........, SSID="PENTESTER"
792 21.88403…  7e:0b:e4:67:fd:3d      Broadcast     802…   54 Probe Request, SN=0, FN=0, Flags=........, SSID="PENTESTER"
793 21.91828…  fe:1d:10:9c:60:39      Broadcast     802…   49 Probe Request, SN=0, FN=0, Flags=........, SSID="PENTESTER"
794 21.91902…  fe:1d:10:9c:60:39      Broadcast     802…   54 Probe Request, SN=0, FN=0, Flags=........, SSID="PENTESTER"
795 21.92025…  0e:20:fc:ab:d8:a2      Broadcast     802…   49 Probe Request, SN=0, FN=0, Flags=........, SSID="PENTESTER"
796 21.92089…  0e:20:fc:ab:d8:a2      Broadcast     802…   54 Probe Request, SN=0, FN=0, Flags=........, SSID="PENTESTER"
797 21.94623…  1e:b3:d5:b7:43:d9      Broadcast     802…   49 Probe Request, SN=0, FN=0, Flags=........, SSID="PENTESTER"
798 21.94690…  1e:b3:d5:b7:43:d9      Broadcast     802…   54 Probe Request, SN=0, FN=0, Flags=........, SSID="PENTESTER"
799 21.98471…  1e:90:cb:d4:2d:b1      Broadcast     802…   49 Probe Request, SN=0, FN=0, Flags=........, SSID="PENTESTER"
800 21.98541…  1e:90:cb:d4:2d:b1      Broadcast     802…   54 Probe Request, SN=0, FN=0, Flags=........, SSID="PENTESTER"
801 21.98655…  de:06:c1:07:d8:fd      Broadcast     802…   49 Probe Request, SN=0, FN=0, Flags=........, SSID="PENTESTER"
802 21.98728…  de:06:c1:07:d8:fd      Broadcast     802…   54 Probe Request, SN=0, FN=0, Flags=........, SSID="PENTESTER"
```

```
> Frame 261: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface wlan1mon, i
> Radiotap Header v0, Length 13
> 802.11 radio information
> IEEE 802.11 Probe Request, Flags: ........
v IEEE 802.11 Wireless Management
  v Tagged parameters (17 bytes)
    v Tag: SSID parameter set: "PENTESTER"
        Tag Number: SSID parameter set (0)
        Tag length: 9
        SSID: "PENTESTER"
    v Tag: Supported Rates 1, 2, 5.5, 11, [Mbit/sec]
        Tag Number: Supported Rates (1)
        Tag length: 4
        Supported Rates: 1 (0x02)
        Supported Rates: 2 (0x04)
        Supported Rates: 5.5 (0x0b)
        Supported Rates: 11 (0x16)
```

```
0000  00 00 0d 00 04 80 02 00  02 00 00 00 00 40 00 00   ········ ·····@··
0010  00 ff ff ff ff ff ff 42  d8 55 0c 21 6f ff ff ff   ·······B ·U·!o···
0020  ff ff ff 00 00 00 09 50  45 4e 54 45 53 54 45 52   ·······P ENTESTER
0030  01 04 02 04 0b 16                                  ······
```

Figure 13: Scapy Probe Requests With Insufficient Wireless Management Data

Scapy is very powerful in transmitting network packets; thus, in future developments of this project, improving the Scapy script to handle wireless details more accurately would help send packets more reliably and quickly. This tool would be essential to pentest the system's security resilience.

3.5 Conclusion

Whether or not the sensor could be injected with spoof MAC probe requests is answered. Future research questions will examine further implications, consequences, privacy protocols, and other key factors to see what changes could make the system as secure and resilient to potential adversaries as possible.

4 Defensive Part

The purpose of this report is to provide a comprehensive overview of our Anomaly Detection Algorithm designed to safeguard the integrity and reliability of the data collected from WiFi sensors in smart cities. This algorithm combines input validation, data sanitization, unsupervised machine learning for anomaly detection, and protection against Denial-of-Service (DoS) attacks. Here is the working model 14.

4.1 Data Acquisition and Preprocessing

4.1.1 Data Import: Streamlining Data Collection

Our meticulous data collection process commences with the acquisition of two pivotal datasets: the primary dataset, training_dataset.csv, and the secondary dataset, testing_dataset.csv. These datasets serve as the bedrock of our analysis. To ensure the seamless import of these datasets, we harness the versatile capabilities of the pandas framework [11].

4.1.2 Data Cleaning: Elevating Data Integrity

Elevating data quality to a paramount status, our data cleaning procedures are nothing short of rigorous. A pivotal step in this process is the randomization of data, a measure that guarantees an unbiased representation of information. We also adopt a stringent policy towards rows lacking essential data in columns such as mac_address, eventtype, and zone, which are promptly omitted.

4.1.3 Feature Engineering: Transforming Data for In-Depth Analysis

A cornerstone of our data preprocessing efforts, feature engineering plays a pivotal role in elevating the quality and suitability of our data for analysis [11]. A battery of transformations is applied to our datasets, including:

- Encoding the eventtype column based on predefined conditions.
- Validation of epocutc values to ensure accuracy.
- Encoding of recognizable zone codes.
- Stringent validation of mac_address using regex patterns.
- Encoding of the techtype column based on the presence of specific values.

4.1.4 Feature Pruning: Enhancing Dataset Efficiency

To streamline our dataset and reduce redundancy, we engage in the strategic process of feature pruning. This involves the removal of original columns that now have their encoded counterparts.

4.2 Anomaly Detection Mechanism

4.2.1 Model Selection: The Core of Anomaly Detection

At the heart of our anomaly detection mechanism lies the critical decision of model selection. We have strategically opted for the Isolation Forest model, renowned for its exceptional proficiency in isolating anomalies [12]. This unsupervised machine learning model excels in segregating observations and identifying outliers based on individual point isolation levels.

4.2.2 Training: Fortifying the Model

Our Isolation Forest model undergoes rigorous training using the primary dataset (train_df). An essential parameter to highlight is contamination=0.5, a crucial factor providing an initial estimate of the proportion of anomalies present in the data, or malign data.

4.2.3 Prediction: Identifying Anomalies

Following comprehensive training, the model takes center stage in identifying anomalies within our secondary dataset (test_df). Predictions follow a straightforward scheme: an output of -1 denotes anomalies, while a value of 1 signifies regular data points.

4.2.4 Evaluation: Measuring Effectiveness

To gauge the efficiency and accuracy of our anomaly detection model, we employ the accuracy metric. This metric offers a valuable ratio, representing the proportion of correctly identified anomalies relative to the total number of data points within the test dataset.

4.3 Additional Methods and Strategic Considerations

4.3.1 Feature Hasher: A Potential Enhancement

While not actively deployed within our current algorithm, the Feature Hasher remains a potent tool in our arsenal. It holds the potential to transform categorical data into a machine-learning-friendly format [11]. Its utilization remains an option for future refinements, promising improved performance and insights.

4.3.2 Alternative Models: Expanding the Toolkit

In our relentless pursuit of robust anomaly detection, we explore the utilization of alternative models, including One-Class SVM, Local Outlier Factor, and Autoencoders. These models offer deeper insights into potential outliers, particularly in high-dimensional datasets [12].

4.3.3 Recommendations: Ensuring Long-term Efficacy

For long-term efficacy, we recommend periodic model retraining to adapt to evolving data patterns. Fine-tuning hyperparameters, such as the contamination factor, can further optimize the model's accuracy. Additionally, an essential augmentation we're considering is a feedback mechanism that allows the system or users to flag false positives or negatives, thereby enhancing the model's precision over iterations [13].

4.4 Real-time Anomaly Detection: Integration and Testing

The next phase of our algorithmic development focuses on real-time application and prompt anomaly detection.

4.4.1 Flask Integration

We leverage Flask, a lightweight web framework, to enable real-time data ingestion. This integration allows for immediate anomaly checks on incoming JSON data from WiFi sensors.

4.4.2 Loading the Model

Using the joblib library, the pre-trained Isolation Forest model is loaded into memory for real-time predictions.

4.4.3 Data Handling

Upon receiving new data, it's promptly converted into a DataFrame structure, facilitating compatibility with the pre-existing feature engineering function.

4.4.4 Feature Engineering

As before, rigorous transformation processes are conducted, including encoding and validating various elements in the incoming data.

4.4.5 Anomaly Prediction

The data, once processed, is fed into the model to determine its nature. It's then tagged as benign or malign, based on the model's prediction.

4.4.6 Immediate Response

This real-time system offers swift decision-making. If incoming data is identified as an anomaly (anomaly value of -1), it triggers an alert stating ”The request is malign.” Conversely, regular data returns the message ”The request is benign.” This mechanism proves instrumental in rapid response and decision-making.

4.4.7 Integration in a Real System

Such a real-time anomaly detection system offers substantial advantages. It provides prompt alerts, adaptability to different types of data inputs, resource efficiency, and enhanced security by identifying malicious injections promptly.

4.5 Contribution - Ishraq Haider Chowdhury

4.5.1 Fake Data Generation with Python library Faker

One of the essential contributions to the project was the development of a fake data generation mechanism using the Python library Faker. This mechanism replicates the original data from the data stream, which is crucial for various aspects of the project. Fake data generation serves as a vital component in preparing datasets for training machine learning models to detect anomalies effectively. This synthetic data enables comprehensive testing and model development without the need for extensive real-world data.

4.5.2 Code for Input Validation in the Flask App

Another noteworthy contribution involved writing robust input validation code within the Flask application. Input validation is a critical security measure to prevent malicious data injection and maintain data integrity. By implementing thorough input validation, we ensure that only valid and safe data is processed by the application. This significantly enhances the security and reliability of the system.

4.5.3 Threading Implementation

Threading was introduced as a performance optimization measure. Threading allows concurrent execution of tasks, improving the application's efficiency and responsiveness. It was a crucial addition to handle data streams efficiently and ensure real-time processing without significant delays.

4.5.4 Integration Attempts with Odysseus

An ambitious effort was made to integrate the functionality directly into Odysseus. Odysseus is a powerful data stream management system. However, this endeavor faced challenges due to resource limitations. Despite the challenges, the attempt to integrate our anomaly detection mechanisms into Odysseus demonstrates a commitment to leveraging existing technologies for enhanced data processing.

4.5.5 Conversion Challenges to PQL (Procedural Query Language)

A significant learning experience was encountered while attempting to convert Python code to PQL (Procedural Query Language). PQL is a language often used for querying and processing data streams. While this conversion presented difficulties, it provided valuable insights into the complexities of adapting code to different environments and languages.

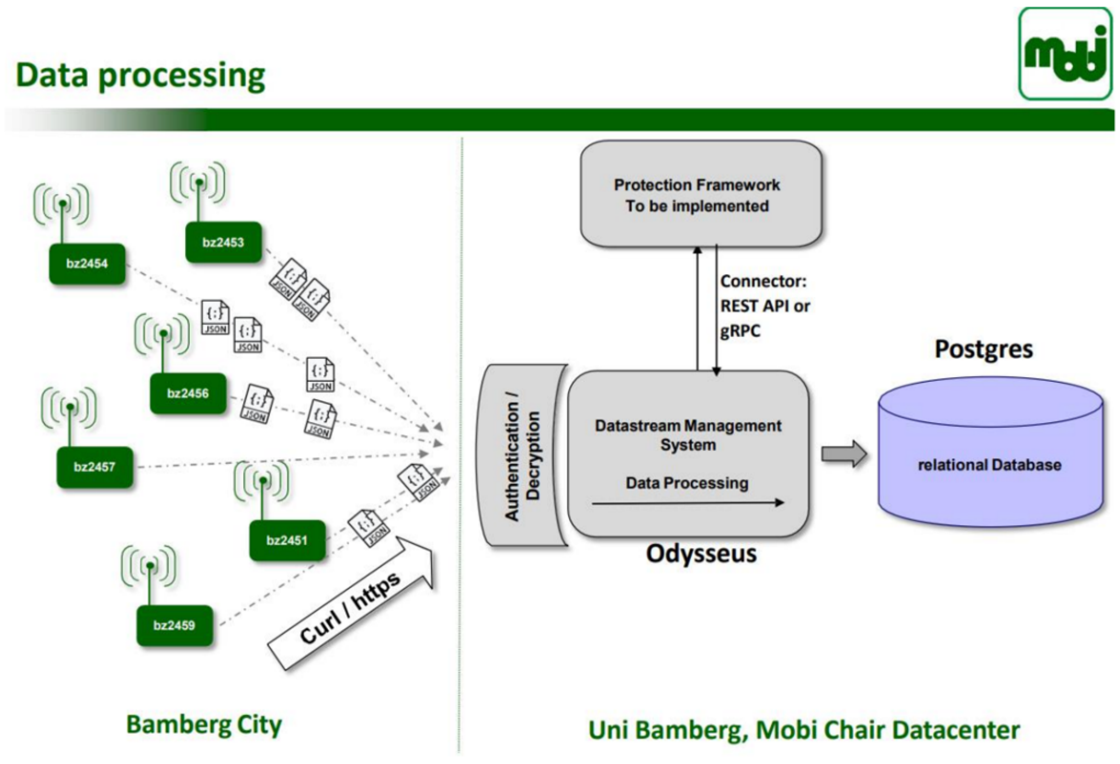

Data processing
bz2454
bz2453
bz2456
bz2457
bz2451
bz2459
JSON
Curl / https
Bamberg City
Protection Framework To be implemented
Connector: REST API or gRPC
Authentication / Decryption
Datastream Management System
Data Processing
Odysseus
Postgres
relational Database
Uni Bamberg, Mobi Chair Datacenter

Figure 14: Data Protection Model

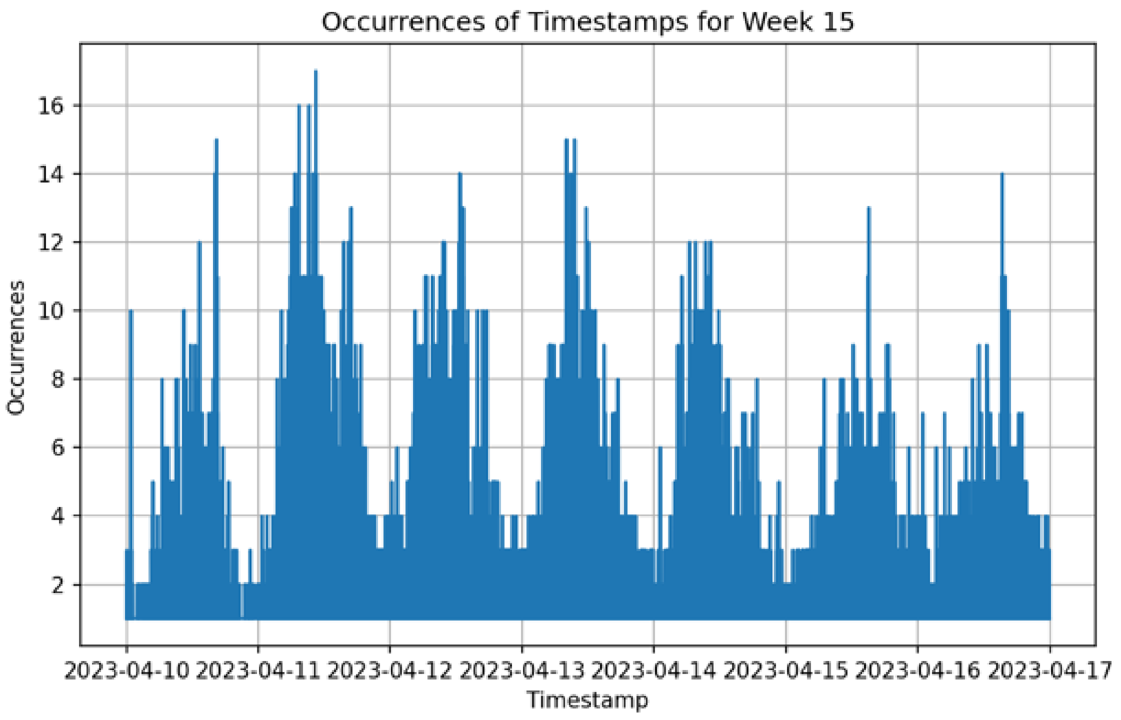


Figure 15: Occurrences of Timestamps for week 15 in 2023

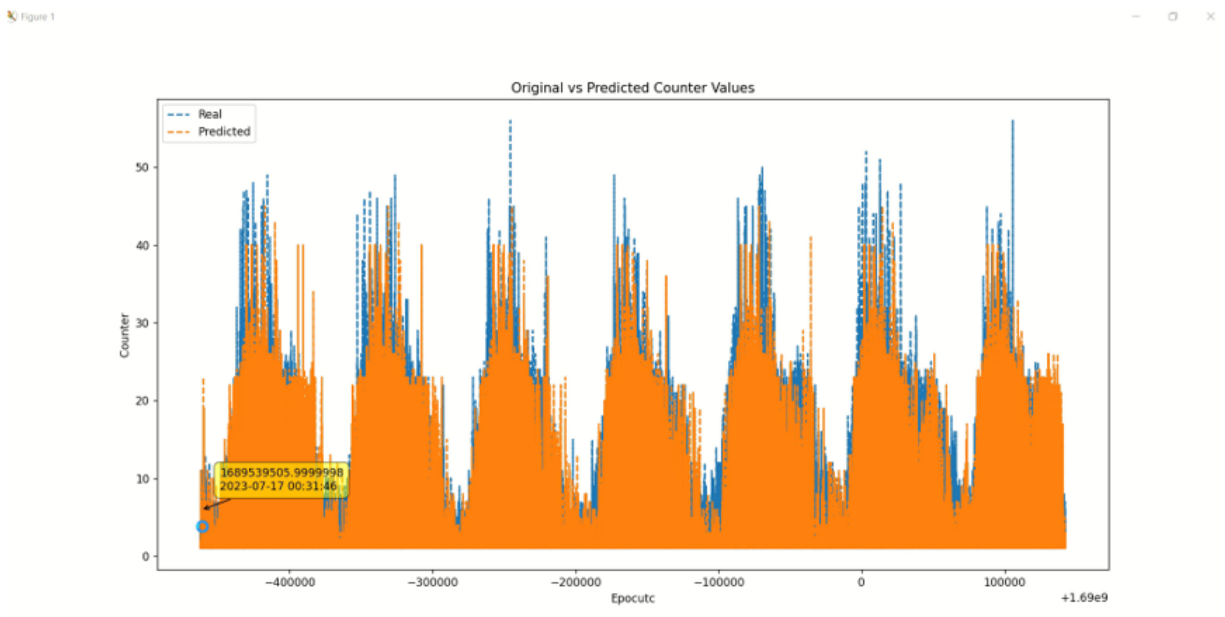


Figure 16: Original vs predicted counter values

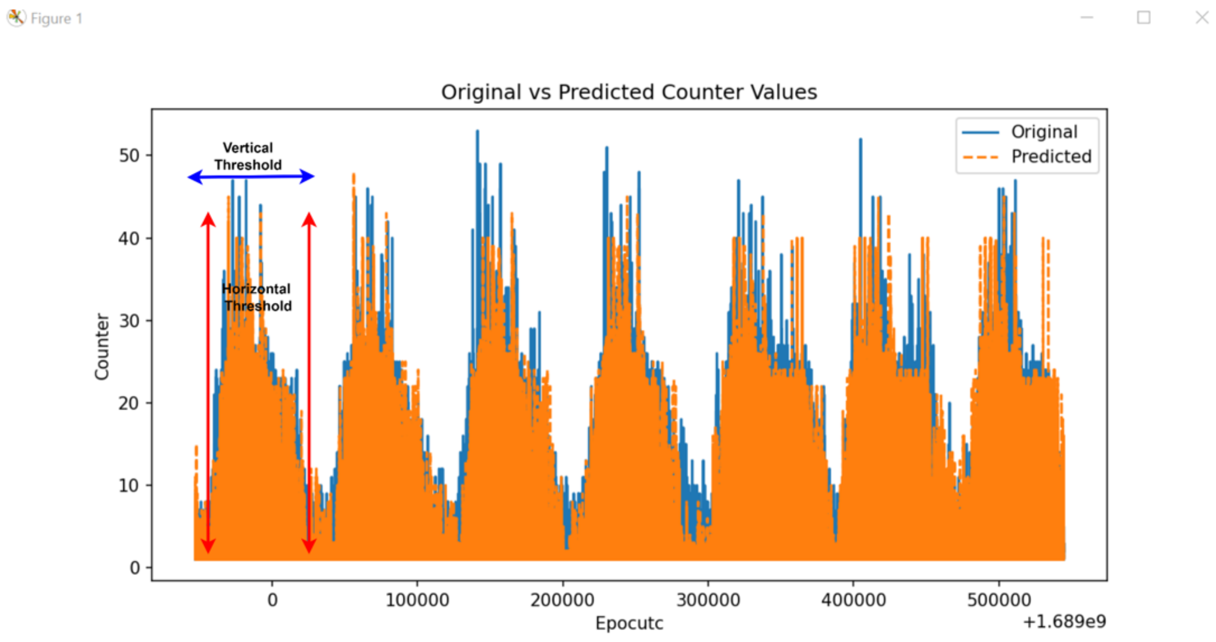


Figure 17: Rate limiting threshold technique

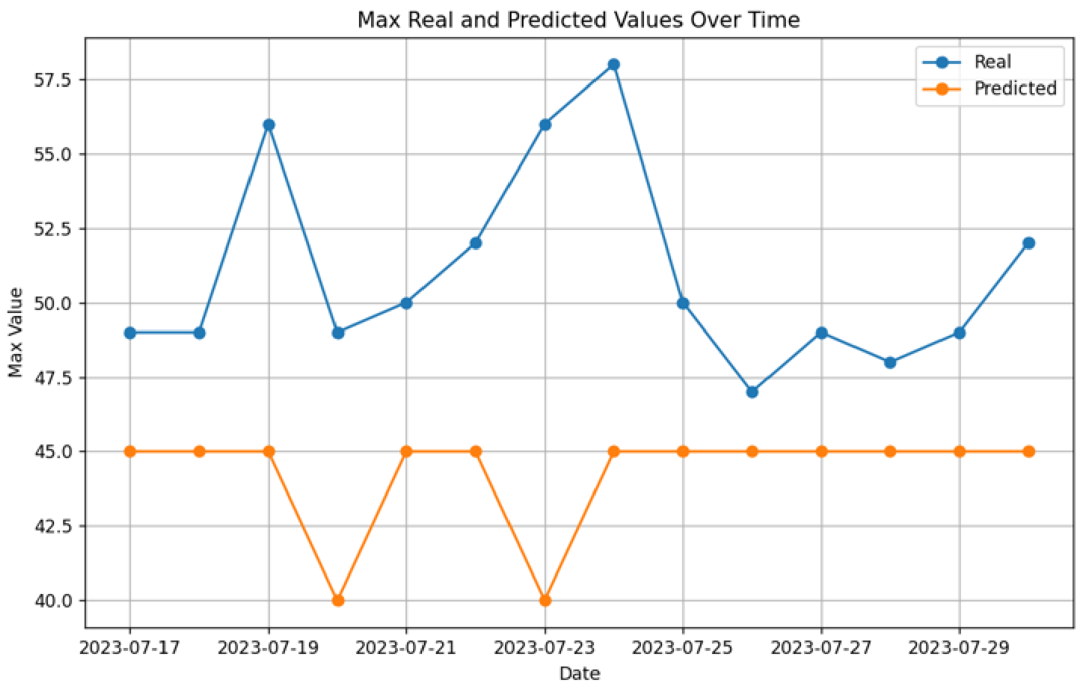


Figure 18: Data to set up a rate limit (Threshold)

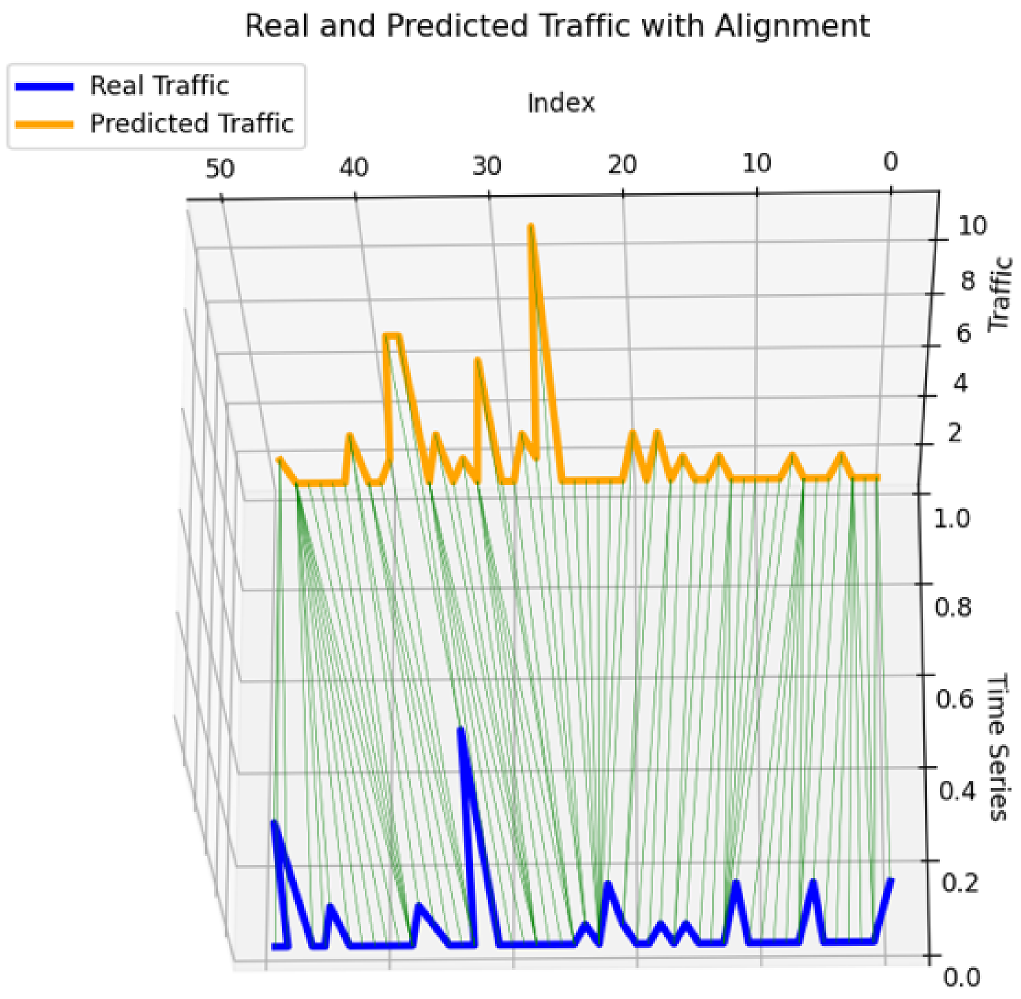


Figure 19: Real and predicted traffic with alignment

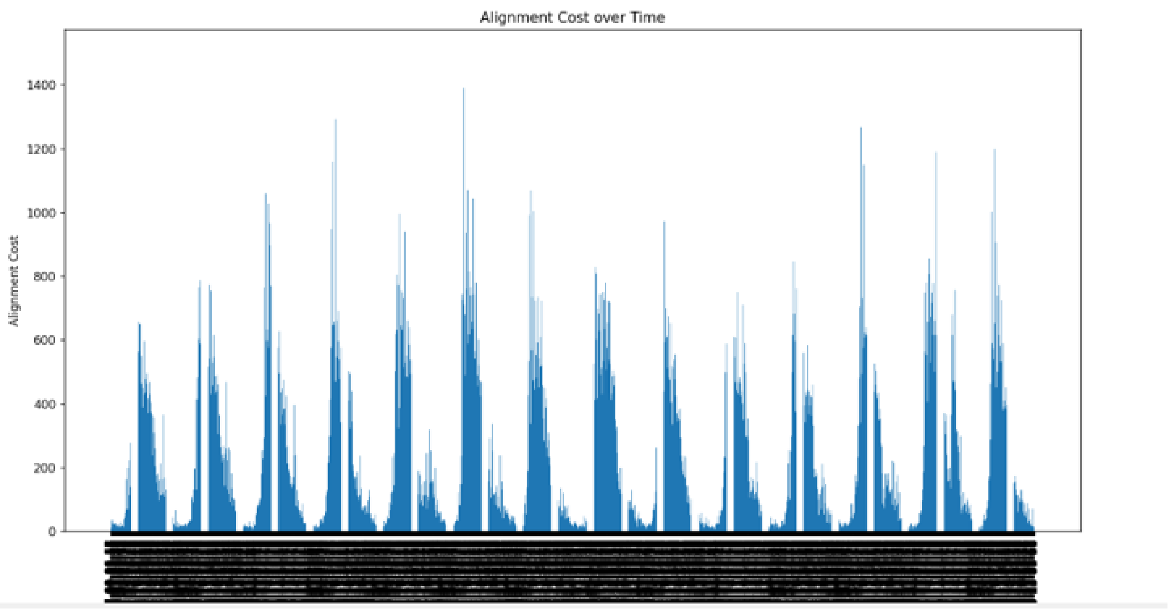


Figure 20: Alignment cost over time

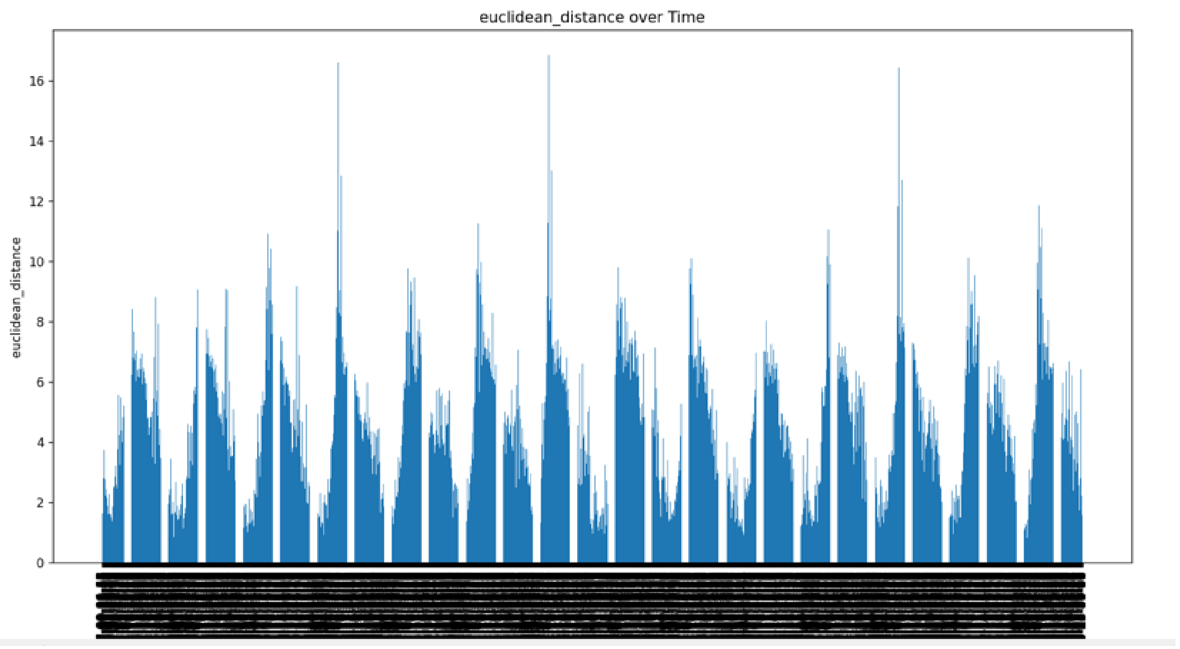


Figure 21: Euclidean distance over time

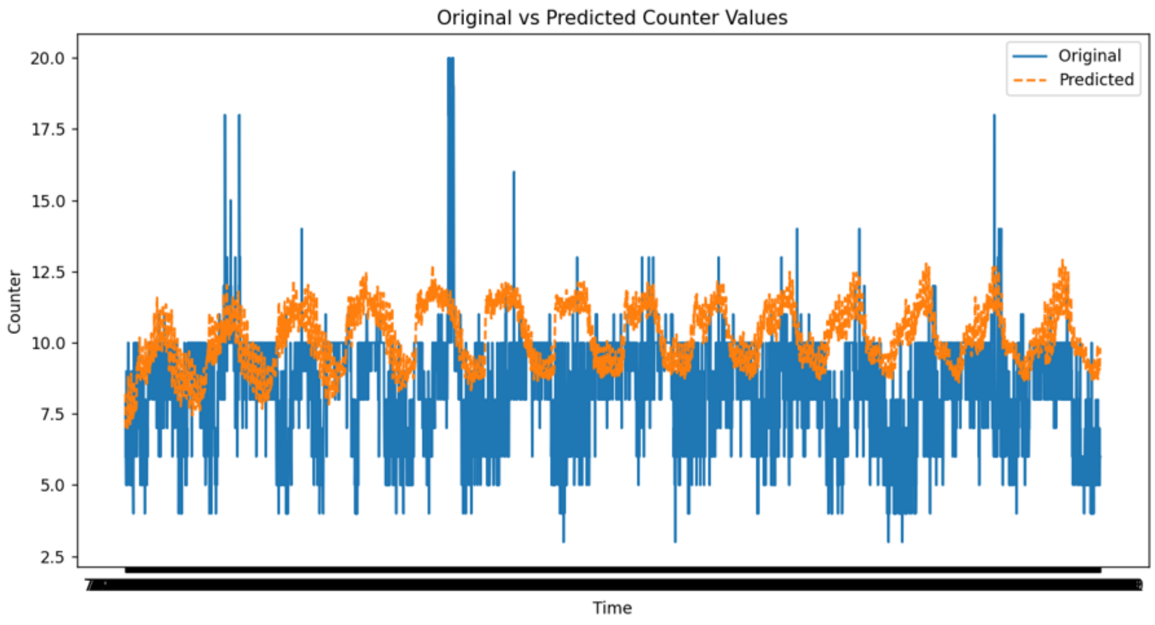


Figure 22: Original vs predicted counter values

5 Acknowledgements

We thank the Chair of Mobile Systems for giving us a great chance to work on this exciting project. We are thankful to Prof. Dr. Daniela Nicklas and our supervisors, Mr. Michael Freitag & Mr. Christoph Baum, for giving us vital feedback, helping us throughout the project, and keeping us on track. Additionally, we are thankful to Safectory, which helped us understand the workings of the sensors, and to the University of Washington students who have shared their ideas and worked with us on this project, especially Mr. Anthony J Bustamante Suarez, for contributing to our group and helping us create a potential defensive approach.

6 Conclusion

Concluding both approaches discussed in this project, it is essential to mention how vulnerable a system infrastructure could be and how it could be avoided. While also highlighting the importance of continuous security validation to prevent any risk to the infrastructure. Furthermore, building upon the foundational algorithm for analyzing sensor data, the real-time testing and integration phase showcases the commitment to enhancing security at every level. The immediate detection of anomalies serves as a deterrent to malicious actors and ensures the preservation of data integrity. By seamlessly integrating such a secured system into the existing digital infrastructure, we are taking a significant stride toward making the city smart but also secure and resilient.

This report provides a comprehensive understanding of our approach to finding the vulnerabilities of the sensor, simulating adversary behavior, anomalies detection in JSON data from WiFi sensors, highlighting the crucial steps in data preprocessing, model selection, real-time integration, and the strategic considerations for future enhancements.

References

[1] D. Coleman and D. Westcott, *CWNA Certified Wireless Network Administrator Study Guide: Exam CWNA-108*. Wiley, 2021. [Online]. Available: https://books.google.com.pk/books?id=DWQXEAAAQBAJ

[2] "Implementing MAC Randomization | Android Open Source Project." [Online]. Available: https://source.android.com/docs/core/connect/wifi-mac-randomization

[3] "Use private Wi-Fi addresses on iPhone, iPad, iPod touch, and Apple Watch," Mar. 2023. [Online]. Available: https://support.apple.com/en-us/HT211227

[4] "Mac Randomization in Windows – Forensic 4:cast," Feb. 2022. [Online]. Available: https://forensic4cast.com/2022/02/mac-randomization-in-windows/

[5] J. Ansohn McDougall, C. Burkert, D. Demmler, M. Schwarz, V. Hubbe, and H. Federrath, "Probing for Passwords – Privacy Implications of SSIDs in Probe Requests," in *Applied Cryptography and Network Security*, G. Ateniese and D. Venturi, Eds. Cham: Springer International Publishing, 2022, vol. 13269, pp. 376–395, series Title: Lecture Notes in Computer Science. [Online]. Available: https://link.springer.com/10.1007/978-3-031-09234-3_19

[6] "An Artist Used 99 Phones to Fake a Google Maps Traffic Jam | WIRED." [Online]. Available: https://www.wired.com/story/99-phones-fake-google-maps-traffic-jam/

[7] "AWUS036NHA." [Online]. Available: https://www.alfa.com.tw/products/awus036nha

[8] "macchanger | Kali Linux Tools." [Online]. Available: https://www.kali.org/tools/macchanger/

[9] "Scapy." [Online]. Available: https://scapy.net/

[10] "IntelÂ® Killerâ„¢ Wi-Fi 6 AX1650 (x/w) - Product Specifications." [Online]. Available: https://www.intel.com/content/www/us/en/products/sku/211609/intel-killer-wifi-6-ax1650-xw/specifications.html

[11] OWASP, "Input validation cheat sheet," *OWASP Cheat Sheet Series*, 2023. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

[12] Wikipedia, "Anomaly detection," *Wikipedia*, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Anomaly_detection

[13] ASQ, "Auditing," *ASQ*, 2023. [Online]. Available: https://asq.org/quality-resources/auditing

Declaration of Authorship

I hereby declare in accordance with Section 9 (12) APO that I have written the above project work independently and have not used any sources or aids other than those specified. Furthermore, I declare that the digital version of the printed copy of the project work corresponds without exception in content and wording and that I have taken note of the fact that this digital version can be subjected to a software-supported, anonymized check for plagiarism.

Bamberg, September 30th, 2023	Ibrahim
(Place, Date)	(Signature)
Bamberg, September 30th, 2023	Ishraq
(Place, Date)	(Signature)

A Appendix

A.1 GitLab Repository

https://gitlab.rz.uni-bamberg.de/mobi/teaching/sose2023/mobi-prs-m-2-crowdanym-teams/secure-tracking-and-data-poisoning