BLuE CRAB: RSSI Detection Pattern Analysis for Flagging System Development

Zhi Qu

July 18 2025

1 Abstract

BLuE CRAB is a program that aims to address the vulnerability of BLE(Bluetooth Low Energy) devices used for malicious tracking. With the advancement in technology, notably Apple's AirTag, the risk of being stalked with the aid of BLE devices is on the rise. This project works to assess patterns in RSSI (Received Signal Strength Indicator) levels between suspicious and safe devices from sample data. The program works with parsing, sorting, and filtering the raw data into understandable graphs. Further assessment of devices with significant detection time adds a possible perimeter for the classification system. My analysis aims to determine patterns to support the development of a new RSSI strength classification system.

2 Introduction

BL(u)E CRAB is an app that detects RSSI (Received Signal Strength Indicator) signals from BLE (Bluetooth Low Energy) devices. It aims to address the threat of malicious tracking devices being used for stalking. (Conklin et al. (2025))

2.1 Survey

A survey was conducted by the developers of AirGuard a app for both Android and IOS users to detect Bluetooth trackers. the survey aimed to assess the scale of the tracking issue and effectiveness of protection measures. Users detects 2.9 tracker daily, though the amount of those devices that are false positives is unknown. (Turk and Hutchings (2023))

3 Others' Solutions

3.1 Apple

In an effort to address the risk their cheap and accessible AirTags created, Apple phones' update included a warning system to alert the user to detected devices. Along with a update on AirTags to sound an alarm when disconnected from their owner for Issues:

- the software only detects devices that are disconnected from their owners. This
- device mac address rotates to avoid software tracking by querying the server for a specific address(Rescorla (2023))

3.2 AirGuard

Airguard is a app developed by Alexander Heinrich, Niklas Bittner, and Matthias Hollick to provide a tracker detection tool for android users. They reverse engineered the feature Apple developed for IOS. (Heinrich et al. (2022)) Flagging perimeters:

- over 30 minute detection duration
- 3 detections occurrence
- 400 meters traveled with user
- if device has been flag with in 7 hours.

4 Methodology

4.1 Datasource

the data analysised in this paper was collected by the DIPr Lab with their demo app BL(u)e CRAB. The test were conducted with known "suspicious" devices.

4.2 Reading Data Files

First the data file needs to be combed through to extract the MAC address, RSSI value, and time stamp. the json file is a dictionary with detections key which contains keys for the "mac", "rssi", "t". In a straight forward manner, mac and rssi values could be collected in a loop for each detection log. The time required a more specified format to extract the information. The time zone recorded was in EDT (Eastern Daylight Time) and not in the default operator datetime.strptime. Thus the following code is used to remove the time zone information from the temporary time variable.

```
        timestamp_str = entry['t']
#removing the time zone (EDT) which can't be read by the operater
        timestamp_str_clean = " ".join(timestamp_str.split()[:-2] + [timestamp_str.split()[-1]])
        timestamp = datetime.strptime(timestamp_str_clean, "%a %b %d %H:%M:%S %Y")
```

Figure 1. Code for time readings

4.3 Distinguish Devices

After close examination of the output from the newly stored variables, it is clear that the detection data contains a distinct log for each device at every second detected. This creates a difficult situation where the information much keep the same index in their respective variables to produce the correct pairing. So first a separate list, fmac, is assigned each distinct value in mac, thus only storing one of each device. Then, in a loop for every device in fmac, every value in mac is checked for the same device address. based on the index of the matching device in mac, the rssi and time values are stored into a temporary list then stored as a list in a list.

```
def sort():
    mac, rssi, time = make_rssi()
    fmac = list(set(mac))
    arssi = []
    atime = []
    for i in fmac:
        r = []
        t = []
        for idx, e in enumerate(mac):
            if i == e:
                r.append(rssi[idx])
                t.append(time[idx])
        arssi.append(r)
        atime.append(t)
```

Figure 2. Code of the sort function

4.4 Sort Based on Risk

By cross referencing the logged device address with a dataset of known suspicious devices, all the corresponding information of that device is removed and stored in a new set of variables. When plotting, the suspicious devices are graphed in red and safe ones in blue. With is comparison easily observed, a pattern starts to arise. Seen in figures 3 and 4, the suspicious devices tend to have a higher rssi value than safe devices.

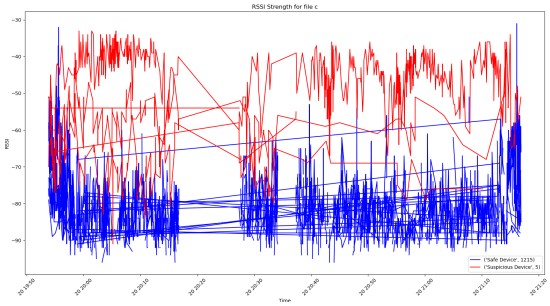


Figure 3. Data file c

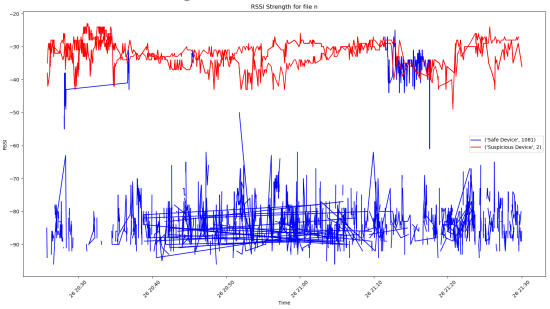


Figure 4. Data file n

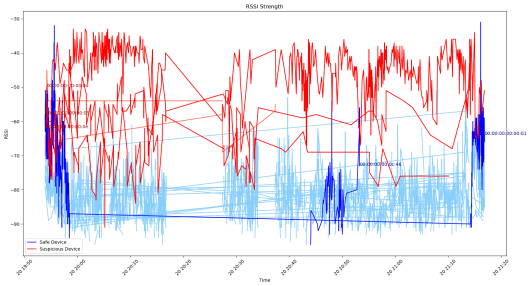
4.5 Highlight Long Detections

The exact difference between devices graphs can get lost with other device of similar signal strength as seen in figures 3 and 4. So to we added a higher parameter, the devices with significant periods of detection is separated, set to 60 seconds. The duration of detection in calculated with the amount of detection, as there is a distinct detection log for every second, thus the total amount of detections for one devices is the same a total seconds it was recorded for. When plotting, the normal data graph's the colors are changed to lighter shade of red and blue while the devices with a significant detection time frame remains pigmented. Note: all suspicious devices in all logs were flagged with significant duration.

```
#the colors were changed to be a lighter verson of the original colors
    for i in range(len(macsafe)):
        sorted_pairs = sorted(zip(timesafe[i], rssisafe[i]), key=lambda x: x[0])
        timesafe_paired, rssisafe_paired = zip(*sorted_pairs)
        plt.plot(timesafe_paired, rssisafe_paired, color="lightskyblue")

# specially plot significant devices with bold color
    for i in range(len(labelsafe)):
        sorted_pairs = sorted(zip(labelsafe_t[i], labelsafe_r[i]), key=lambda x: x[0])
        timesafe_paired, rssisafe_paired = zip(*sorted_pairs)
        plt.plot(timesafe_paired, rssisafe_paired, color = "blue")
        plt.text(labelsafe_t[i][-1], labelsafe_r[i][-1], labelsafe[i], color="navy")
```

Figure 5. the highlighted version of dat file c.



4.6 Compare Time of Day

```
# new plot function to simplify main()
def plot(data, setsus, color1):
    mac, rssi, time = make_rssi(data)
    fmac, arssi, atime = sort(mac, rssi, time)
    macsus, rssisus, timesus, = findsus(setsus, fmac, arssi, atime)

    plt.plot(timesus[0], rssisus[0], color=color1, label=("Suspicious Devices", len(macsus)))

    for i in range(len(macsus)):
        sorted_pairs = sorted(zip(timesus[i], rssisus[i]), key=lambda x: x[0])
        timesus_paired, rssisus_paired = zip(*sorted_pairs)
        plt.plot(timesus_paired, rssisus_paired, color=color1)
```

5 Discussion

All suspicious devices are flagged as significant in the detection time. It is unknown wether these pattern matches real world circumstances or is a trend from the test set up.

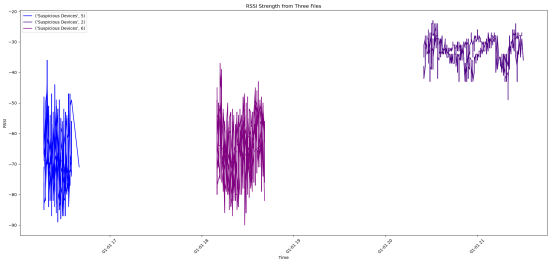


Figure . file g and n with file j in blue
No distinct trend could be identified from the assessment of different times of the day. While some of the data sets graphed together present a possible pattern, it is more likely that the difference in RSSI strength comes from where the tracker was placed in relation to the detector.

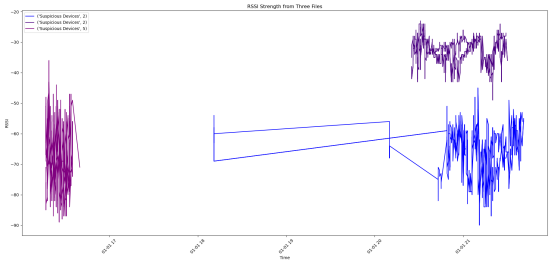


Figure . file g and n with file a in blue

6 Conclusion

- The program is able to produce easily read able graph and provide a general understanding of the device distribution based on RSSI values and time. Suspicious devices tend to show higher RSSI values and all are detected for a significant amount of time(60 seconds)

6.1 Limitations

- The data input must be in the same format as the sample data or the information extraction would result in an error
- individual data points are often difficult to see, meaning that from the observations are high level generalizations rather than confirming a precise pattern.

References

Conklin, D., Pappachan, P., and Yus, R. (2025). Bl (u) e crab: A user-centric framework for identifying suspicious bluetooth trackers. In *2025 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 570–572. IEEE Computer Society.

Heinrich, A., Bittner, N., and Hollick, M. (2022). Airguard-protecting android users from stalking attacks by apple find my devices. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 26–38.

Rescorla, E. (2023). Defending against bluetooth tracker abuse: it's complicated. *Educated Guesswork*.

Turk, K. I. and Hutchings, A. (2023). Stop following me! evaluating the effectiveness of anti-stalking features of personal item tracking devices. *arXiv preprint arXiv:2312.07157*.