Title
Finding Critical Infrastructure Using Wardriving Data

Author
Wu, Huanlei

Publication Date
2022

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA SAN DIEGO

Finding Critical Infrastructure Using Wardriving Data

A Thesis submitted in partial satisfaction of the
requirements for the degree Master of Science

in

Computer Science and Engineering

by

Huanlei Wu

Committee in charge:

Professor Aaron David Shalev, Chair
Professor Patrick Pannuto
Professor Stefan Savage

2022

The Thesis of Huanlei Wu is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2022

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my mentor Nishant Bhaskar and my thesis advisor Aaron Shalev for their patience and guidance throughout this journey. Thank you taking the time to discuss my progress and explaining to me the concepts that were foreign to me at the start of this paper. Without their help, I would not have finished my thesis on time, and I am eternally grateful for their support.

I would also like to thank my Helium team, who were considerate of my predicament. Specifically, I would like thank Alex Yen for not overloading me with work during the final days of my thesis and also taking the time to help review my thesis defense.

Lastly, I would like to thank Stefan Savage and Pat Pannuto for agreeing to be my other 2 thesis committee members and catering to my horrible last minute planning.

VITA

2020 Bachelor of Science in Computer Science and Engineering, University of California Santa Cruz

2022 Master of Science in Computer Science and Engineering, University of California San Diego

PUBLICATIONS

Dhananjay Jagtap, Alex Yen, Huanlei Wu, Aaron Schulman, and Pat Pannuto. "Federated infrastructure: Usage, patterns, and insights from "the people's network"." In Proceedings of the 21st ACM Internet Measurement Conference, IMC '21, page 22–36, New York, NY, United States, 2021.

ABSTRACT OF THE THESIS

Finding Critical Infrastructure Using Wardriving Data

by

Huanlei Wu

Master of Science in Computer Science and Engineering

University of California San Diego, 2022

Professor Aaron David Shalev, Chair

As the need for convenience and accessibility of technology grows, so does the number of wireless devices used in our infrastructure. Users of these devices can range from the average individual to corporations to the government. On one hand, these wireless devices can afford their users with an easier experience, but on the other hand, they can become entry points to malicious attacks. While there is extensive research into the security of wireless devices on a small scale, there lacks an understanding of it on a larger scale: what infrastructures are vulnerable, where these vulnerabilities are located, and what devices make the infrastructure vulnerable. In order to get a better understanding of these wireless-capable infrastructures, we first need to get data on how wireless devices are used in infrastructure. One readily available way to obtain this

data is through wardriving. However, wardriving data is extremely large and noisy; it contains all scanned wireless data, including devices not related to any infrastructure. Thus, it can be hard to get any relevant devices out of the data. In this work, we try to understand the viability of identifying infrastructure-related devices using wardriving data. We find that it is possible through using a variety of cleaning, clustering, and filtering heuristics to remove noisy data. We then discuss the devices that we found and consider the implications and impact of a few of them.

Chapter 1

Introduction

We are currently living in an era where there is a nonstop growth of wireless devices due to their convenience and ease of use. Thus, more and more of these wireless devices are making their way into our critical infrastructure. For example, many cities are starting to deploy smart grids to help mitigate power issues or to simply have an easier way to configure and maintain the power grid. While the abundance of these devices can make life easier, it can also provide another entry point for malicious entities to attack and obtain valuable information. Infrastructures such as the smart grids, hospitals, and other establishments that use Bluetooth devices can be majorly affected by these attacks. Thus, having an understanding of what infrastructure is vulnerable, where the infrastructure is, and what devices make the infrastructure vulnerable is critical.

The first step in understanding these burgeoning wireless infrastructures is through the wireless devices that make up the infrastructure. In order to identify which wireless devices are involved in what infrastructure, there must first exist a database of these devices. Unfortunately, there is no registry for users to register their wireless devices, so having a complete and accurate set of wireless devices data is impossible. As a result, people have found other ways to obtain this information, one of them being through wardriving. Using this technique, people can scan for wireless devices in their immediate vicinity and upload the scan data to a database. However, a by-product of wardriving is the large number of irrelevant, non-infrastructure devices that inundates the database. Because the data gathered comes from uncontrolled environments,

devices such as wireless earphones or Bluetooth car stereos are entered into the database, making it difficult to identify any meaningful devices. Thus, the idea of utilizing wardriving data to understand infrastructure and identify devices can be unappealing.

Still, wardriving is the only way to get a data set comprehensive enough to identify a multitude of wireless infrastructures in the real world, if it could at all. Therefore, we were curious of the feasibility of using wardriving data to find infrastructure-related devices. To answer this question, we explored different ways of cleaning, clustering, and filtering of the data to decrease the number of devices to look at, specifically the data that don't pertain to infrastructures. Because the wardriving data set is extremely large, we used only a small subset of wireless devices, looking only at Bluetooth devices in the United States. More specifically, we used the wardriving data obtained from [2] and the US Bluetooth data from WiGLE. From our experiment, we realized that we indeed could identify devices that are a part of the infrastructure after filtering out the irrelevant, noisy data.

Chapter 2 gives a brief explanation and history of wardriving and describes the two data sets. Chapter 3 describes the different clustering and filtering methods used to remove devices that are not part of an infrastructure. Chapter 4 details the device clusters found by using methods described in Chapter 3. Chapter 5 explains the limitations of the data and discusses the security concerns and impact that the identified Bluetooth device clusters may have if attackers were able to successfully take control of them. Chapter 6 concludes the paper and lists out possible future works.

1.1 Related Work

The act of identifying wireless devices with limited information is not a new concept; many works [9, 12, 15, 16, 17] are concerned over the security risks that arise from the rapid growth of wireless devices. Literature [17] acknowledges the security issues and devises a machine learning framework that uses network traffic to classify IoT devices from a chosen set.

Using a device's network traffic characteristics, Sivanathan et al. trained a model that could take in a device from a list of 28 IoT devices and classify it with 99% accuracy. Like all of the aforementioned works, the data the paper looks at is related to WiFi network traffic. A problem that can arise from using WiFi network traffic is that in order to get the traffic data, the individual must tap into the network first. Thus, there is a limitation when it comes to working with network traffic to identify wireless devices. There are limitations for obtaining scan data for wardriving as well (such as long scan times), but tapping into a network is not a concern. For our paper, we take a look into using Bluetooth scan data to identify devices. Another difference between [17] and our work is that [17] only sampled 28 devices and did not attempt to classify any IoT devices outside of those original 28. In our work, we look at the Bluetooth scan data in the United States from Bluetana and WiGLE, allowing our results to be more representative of the devices that could be found.

Literature [15] also created a machine learning classification model using certain characteristics of encrypted WiFi packets and the spectrum they use when transmitting data. It improves upon [17] as it allows for previously unseen devices to be labeled as well. However, like the previous work [17], it also uses machine learning, and identifying devices in this fashion requires a lot of labeled data. This is a problem for data sets that do not have ground truth or have a team that can manually go through and hand label the necessary amount of data points. Wardriving, unfortunately, is one such data set that does not provide a device type next to the scanned device; moreover, going through the data to label enough devices for machine learning is a daunting task. Therefore, for our work, we look into ways of classifying devices for infrastructures without the use of ground truth labels.

Chapter 2

Wardriving Data

Wardriving is the act of searching for unsecured wireless networks by scanning for these networks around a reference point. The term wardriving was coined by Pete Shipley, drawing inspiration from the term wardialing in which an individual automatically dials multiple phone numbers in hopes of finding an unprotected modem. However, unlike wardialing, wardriving involves the person to move around as wireless networks cannot be seen in certain areas due to having a limited range and interference. For our project, all the data we look at comes from utilizing the wardriving technique.

2.1 Bluetana Data

Bluetana is a measurement tool created by graduate students at the University of California San Diego that uses the Android Bluetooth APIs to scan for gas pump skimmers. [2] The motivation behind Bluetana was to help law enforcement locate skimmers easier; before Bluetana, officers would have to manually open up gas pumps in order to check for these skimmers, but with the tool, they are able to automatically detect nearby skimmers. For this project, Bhaskar et al. sent the tool to law enforcement in 4 different states — California, Arizona, Nevada, and Maryland — to collect skimmer data.

Besides skimmer data, Bluetana also captures information on other Bluetooth devices within the area and sends it to a secure database over a cellular link. [2] We use the entries of

this database as one of our data sets for clustering together similar Bluetooth devices.

2.2 WiGLE Data

Our second data set comes from a service called WiGLE, which is an acronym for Wireless Geographic Logging Engine.[1] Like the name suggests, it receives data about wireless networks and stores it into a database for users to pull from. Fortunately, WiGLE provides a Bluetooth Search API[2] that automatically retrieves the data for the devices that match a given attribute. For our project, we provided the API with MAC address prefixes that we believed had devices that were part of an infrastructure. The MAC address prefixes that we used can be found at Appendix A.

Unlike Bluetana, where its user base are mainly law enforcement, a few volunteers, and the occasional academics, WiGLE is used by everyone around the world. The only prerequisite is for the person to register for an account. Afterwards, users can search for wireless networks through its web interface or APIs and update the database either manually by inputting wireless network data on the website or using a supported app to scan and automatically enter it into the system. While the worldwide effort is impressive, to not overwhelm ourselves with too much data, we only look at the devices in the United States as a starting point.

Two features of WiGLE that is important to know about is the QoS (Quality of Service) attribute that comes with each WiGLE entry and the weighted-centroid trilateration. The QoS of an access point (AP) is a way for WiGLE to identify the reliability of it. Taken from WiGLE's wiki: "If an AP is seen on more than one day, or by more than one user, the QOS goes up, because it's more likely to be stable." [3] Since devices that are part of an infrastructure tend to be stationary and online for most of its lifetime, the QoS will be important in filtering out uninteresting devices.

[1] https://WiGLE.net/
[2] https://api.WiGLE.net/swagger#/Bluetooth%20search%20and%20information%20tools/search
[3] https://WiGLE.net/wiki/index.cgi?QoS

The other important feature is the weighted-centroid trilateration formula WiGLE uses to pinpoint where a device is. Even if a device is stationary, the coordinates of the device entered by its users might vary since there are many factors that will affect the accuracy of a GPS coordinate. For WiGLE, the final location of the device is an average of all the longitudes and latitudes recorded of it weighted by its RSSI.[4] We use the weighted-centroid trilateration coordinate as the location of the device when identifying devices near infrastructure.

[4] https://WiGLE.net/faq/

Chapter 3

Removing Irrelevant Data From Wardriving Data

Narrowing down the Bluetooth data was tricky: we didn't want to remove too many devices, but we also didn't want to keep too many devices to manually sift through. This chapter describes our approach to the dilemma and explains the reasoning behind our decisions and thresholds we chose.

To give an overview of our methodology, we first filtered out data which we deemed to be inaccurate. For our case, inaccurate data is data that is not representative of its device. We will go into more details in Section 3.1. After filtering out the inaccurate data, we removed all non-stationary devices as devices that are part of an infrastructure are stationary and usually bound to the spot. We then clustered devices by their device names and MAC addresses as devices with similar MAC addresses and device names are usually part of the same infrastructure with the same functionality. Once we obtained all the clusters, we then started filtering out the clusters in order to identify what devices are part of what infrastructure.

3.1 Preliminary Filtering of Inaccurate Data

Before beginning the main clustering and filtering portion described in the following sections, we made sure that the data points we looked at were at least somewhat accurate and relevant. Because we worked with two data sets, the preliminary filtering process for each data

set is different.

For the Bluetana data set, data that did not have a *geo_accuracy* of less than 5 were filtered out. *geo_accuracy* is the "estimated horizontal accuracy radius in meters of this location at the 68th percentile confidence level."[1] In layman's terms, if the *geo_accuracy* returned was a 5, that means the Android library was 68% confident that the device was within 5 meters of the geo-coordinate it provided. Unfortunately, this function only returns the horizontal accuracy and not the vertical accuracy, so while we have some confidence in the longitudinal data, we cannot guarantee accuracy for the latitudinal data.

We also filtered out entries where the major device class was classified as either a 1 or 2 (computer or phone respectively) in the Bluetana data set. We did not want to remove too many devices from the data set when filtering in case we missed a device that was part of some infrastructure, but we were fairly confident that devices belonging in those two categories were not part of any critical infrastructure.

For the WiGLE data set, we simply filtered out devices where the QoS was less than 4. We chose the number 4 because it is the the rounded value of the median of the possible QoS values (0-7).

3.2 Stationary Devices

Bluetooth devices are not only seen in infrastructure. Everyday devices such as headsets and speakers utilize the technology for user convenience. Because we wanted to only look at devices that could pertain to infrastructure, we had to take into consideration that many entries in the database would contain such devices. We realized early on that many of these non-infrastructure devices move from one place to another throughout the day. Therefore, we defined what it means for a device to be stationary and removed those that did not fit the criteria. The following bullet points outline our definition of a stationary device:

[1] https://developer.android.com/reference/android/location/Location#getAccuracy()

- There must be at least 2 entries for the device within the data set; we cannot tell whether a device is stationary or not with only one data entry.
- The time between the first and last recorded entry of the device must be at least 10 hours apart. We didn't want to have too large of a time threshold as we didn't want to remove too many devices from our device pool. Furthermore, a typical workday usually maxes out at 10 hours, and any non-stationary devices follow its owner home after work.
- All recorded coordinate points of the device must be within a 50-meter radius. This is done by finding the centroid of all the device's coordinates and measuring the distance between the centroid to a device coordinate using the Haversine formula. If any distance is greater than 50 m, we declare that the device is not stationary.

Finding the stationary devices for the Bluetana data set was simple since the database provided an entry for each time a device was seen. However, for WiGLE, the Bluetooth Search API does not provide that information. Instead, we had to use the Bluetooth Detail API[2] to get a list of all the times a device was seen and its geo-coordinate at that time.

3.3 Clustering

For the most part, devices from the same infrastructure with the same functionalities have similar names; devices can be grouped up based on their device names' prefixes or suffixes. For example, SP800-027C and SP800-02F5 are both of the SP800 family of speed signs. John's iPhone and Teri's iPhone are both iPhones. However, clustering by both both prefixes and suffixes of a device name at the same time is difficult as we want a one-to-one matching between a device cluster and device. From the data, we concluded that similar devices (devices of the same infrastructure and functionality) more often than not shared a prefix as opposed to a suffix, so for this project, we only group device names by prefixes.

[2] https://api.WiGLE.net/swagger#/Bluetooth%20search%20and%20information%20tools/detail

3.3.1 Name Clustering: Original Method

At the beginning of the thesis — when we were using Bluetana's data — we tried to come up with our own way of grouping up similar devices. The method is as follows:

1. The device name is split into sections using spaces, underscores, and hyphens. This includes both the device name that is being compared with and being compared to.

2. For each section, the letters of one device name is compared with the letters of another device name at the same position. If they match, add the number of matching letters before the current position and an additional point to the counter that keeps track of the total amount of points earned in the section. If it doesn't match, nothing is added to the counter. The value of the counter is then divided by the the full score of the section (i.e., the score if every letter matched). Repeat this step for all the sections.

3. Each section is then weighed to provide the final score. Since the assumption is that a matching or similar prefix implies a higher probability that the devices are similar devices, the weight is higher for the sections at the beginning. The general rule is that each subsequent section's weight is half of the current section's unless the subsequent section is the last section. In this case, the last section will share the same weight as its predecessor. When there is only 1 section, that section gets a weight of 1.

4. The device will be clustered with another device if the final score is greater than 0.5.

Algorithm 1 shows the pseudocode for the aforementioned algorithm in more details and Figure 3.1 provides a step-by-step example of the clustering algorithm.

This algorithm, however, has many flaws. As seen in the pseudocode, sections or letters within each section could be cut off if the two device names' lengths did not match, resulting in many false positives and false negatives. Furthermore, devices names that did not have a space, hyphen, or underscore had to have at least half of the name match or else they would not be

Algorithm 1. Original Device Name Clustering Method

```
procedure NAMECLUSTER(devname1, devname2)
    devname1_arr ← split devname1 by spaces                    ▷ device name array 1
    devname2_arr ← split devname2 by spaces                    ▷ device name array 1
    final_score ← 0
    arr_len ← shorter device name arrays

    for idx_arr in range of the shorter device name arrays do
        add_points ← 0
        total_points ← 0
        for idx_sec in range of the shorter section at index idx_arr do
            if devname1_arr[idx_arr][idx_sec] = devname1_arr[idx_arr][idx_sec] then
                total_points ← total_points + add_points + 1
                add_points ← add_points + 1
            else
                add_points ← 0
            end if

            if arr_len equals 1 then
                final_score ← final_score + total_points
            else
                if idx_arr not equals arr_len − 1 then
                    final_score ← final_score + 1/2^(idx_arr+1) * total_points
                else
                    final_score ← final_score + 1/2^(idx_arr) * total_points
                end if
            end if
        end for
    end for
end procedure
```

STEP 1	RNBT-5D68 → [**RNBT, 5D68**]	RNBT-5408 → [**RNBT, 5408**]

STEP 2	RNBT RNBT	5D68 5408
	1 + 2 + 3 + 4 = 10 / 10 = 1	1 + 0 + 0 + 1 = 2 / 10 = 0.2

STEP 3	**Section**	**Points**	**Weight**
	RNBT vs RNBT	1.0	0.5
	5D68 vs 6408	0.2	0.5
	Total		1.0 * 0.5 + 0.2 * 0.5 = **0.6**

STEP 4 0.6 > 0.5 → Cluster devices together

Figure 3.1. Original method explained with a simple example using device names RNBT-5D68 and RNBT-6408.

clustered together. Therefore, we switched to using Jaro-Winkler similarity for name clustering for the WiGLE data.

3.3.2 Name Clustering: Jaro-Winkler Distance

The Jaro-Winkler distance is a way of measuring the similarity between two sequences. [6] It takes the Jaro distance metric and adds in a prefix length constant and a prefix scale constant to favor sequences that have similar prefixes. This feature is essential as our observation was that similar devices have matching or similar prefixes. While this method also has its flaws, it is less restrictive than our original method on prefix matching.

3.3.3 MAC Address Clustering

A problem with both of the name clustering methods is that a nontrivial amount of devices do not have a device name associated with them, and thus, are not placed in clusters. In

order to mitigate this problem, we decided to utilize the MAC address as a means of sorting the devices with no names. For the devices with no names, their MAC addresses were compared to the MAC addresses of all the devices already in a cluster. The nameless device was then placed into the cluster that contained the device that had the closest MAC address value to the nameless device's.

We also used the MAC addresses to further fine-tune the clustering of similar devices. MAC address prefixes (the first 3 bytes of the MAC address) tells us the OUI of that device. OUI stands for Organizationally Unique Identifier, and like the name suggests, it is used to differentiate and identify companies and organizations. While each 3 bytes have a one-to-one mapping to an OUI, the OUI can have a one-to-many mapping to the MAC address prefixes. For instance, the MAC address prefix 00:18:32 is owned by Texas Instruments, but Texas Instruments has hundreds more MAC address prefixes such as 00:18:31 and 00:18:33.

We believed grouping by MAC address prefixes made sense since similar devices usually come from the same company or organization. However, we wanted to cluster with a finer granularity of the MAC addresses, so we also matched 1 additional byte of the MAC address. The final device clusters had all devices in it match the first 4 bytes of their MAC addresses as well as their device names.

3.4 Filtering

Device name clustering is a start to determining which devices are similar to each other, but it is not helpful in pinpointing the devices that are part of an infrastructure what type of devices they are. We may infer that devices SP100, SP80, and SP20 have similar functionalities to one another, but we do not know what infrastructures they are a part of or if they are part of an infrastructure at all. In order to understand what devices belonged to what infrastructure, we use 4 different filtering methods.

3.4.1 Infrastructure Filtering: Google Places API

The first of these filtering methods we used involved calling the Google Places API. While we had the coordinates of each device and thus know where they are on a map, this information told us nothing about its surroundings. Therefore, we used Google Places Nearby Search API to obtain the different place types near a device. The Google Places API documentation[3] provides an exhaustive list of Google place types that can be returned from the API call. For the Nearby Search API, only values from Table 1 are returned.

Using the API, we first obtained a list of different place types found within 50 m for each of the stationary devices. We then found the number of devices that were in one of these place types as a way to understand the popularity of the place types in relation to these devices. We also calculated the percentage of devices within a cluster that had a certain place type to understand how devices were spread out and what place types correlated to what device cluster. We kept only the device clusters where this percentage was greater than 50%.

However, there were 2 main flaws with using the Nearby Search API. One of these flaws was the labeling of the nearby points of interests: not all points of interests are labeled as we would like them to be labeled. For example, a point of interest had the place type *embassy*, but when we further investigated the place, it turned out that the place was an embassy for religion, not a country. The other flaw was the efficiency of the code. If we ever wanted to change the radius threshold from 50 m, we would have to call the API again to update the nearby points of interests. One option is to provide a large radius at the beginning of the API call, but this could increase the data drastically, and sorting out the nearby points of interests would take too much time. Thus, when we started using the WiGLE data instead, we decided to remove the Google Places API method as one of our clustering options and focus on specific infrastructures; the Google Places API was used only for the Bluetana data, and the Bluetana data only used the Google Places API to filter data.

[3] https://developers.google.com/maps/documentation/places/web-service/supported_types

3.4.2 Infrastructure Filtering: Intersections

One infrastructure we were interested in — but the Google Places Nearby Search API could not provide — was intersections. Our workaround was to call upon the GeoNames API, an API which provides the closest intersection to a given coordinate, and use the haversine formula to determine the distance of a device to its corresponding intersection. Like the previous method, our radius threshold is 50 m; any device whose distance from their closest intersection is less than or equal to 50 m would be considered as near an intersection. With the new list of devices that are near intersections, we found the percentage of devices within a cluster that were near intersections and removed any clusters that had a percentage less than 50%.

3.4.3 Infrastructure Filtering: Airports

Another infrastructure we were interested in was airports. We found a reliable source[4] for the coordinates of airports in the US and used it to narrow our stationary device list down to only devices within 1 km of an airport. Since the data set for the airports was so large, we only considered medium and large airports.

Identifying whether a device was near an airport is difficult. Unlike intersections where the shape and size are similar to one another, different airports have different sizes and dimensions. Because we had no way of knowing the shape and size of each airport, we generalized the radius threshold to be 1 km; any devices that are within a 1 km radius from the geo-coordinate of the airport will be considered to by near airports. We then calculated the percentage of devices in a cluster that were near airports and removed the clusters that had a percentage less than 30% to make up for the ambiguities.

3.4.4 Geographical Filtering by US States

Infrastructure devices can also vary by region. Therefore, we decided to look at the devices by US states. To cluster each device by state, we first found the boundaries of each state

[4] https://data-usdot.opendata.arcgis.com/datasets/usdot::airports-1

by using a shape file of the US found on the US government's open data website.[5] With the boundaries, we then sorted the devices by state based on their coordinate points and calculated the percentage of devices within a cluster that were in a certain state. Once again, if the percentage dropped below 50%, the cluster was removed.

[5]https://catalog.data.gov/dataset/tiger-line-shapefile-2017-nation-u-s-current-state-and-equivalent-national

Chapter 4

Results

In this chapter, we will be discussing the interesting infrastructure-related device clusters we found after narrowing down the data. The chapter will be split into 2 sections: one section describes the results found using the Bluetana data set and the other the WiGLE data set.

4.1 Bluetana Data

In total, the Bluetana data set had 1,890,076 entries. However, many of these entries were of the same device scanned at different times. By using the preliminary filtering and our original device name clustering methods, we were able to avoid duplicates devices, narrowing the data down to 359 clusters of stationary devices. We then further narrowed down the data by grouping the clusters by their nearby place types from the Google Places Nearby Search API. This was done so that we could find the number of devices that were in one of these place types as we were interested in devices that appeared often near a specific place type.

Table 4.1 shows the top 10 places the Bluetana scanned devices were near and the number

Table 4.1. Top 10 Google place types that devices were near.

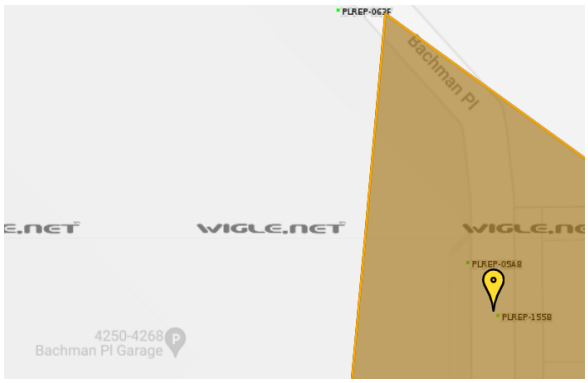
Google Place Type	Count	Google Place Type	Count
restaurant	344	cafe	132
transit_station	288	bar	119
atm	232	parking	119
convenience_store	227	clothing_store	112
gas_station	208	car_repair	107

of devices near it. Although *restaurant* has the most number of devices near it, many of these devices were not part of an infrastructure. In fact, most of the places on the list were inflicted with the same problem: they were buffered by Bluetooth TVs, printers, smart speakers, and other smart devices that are not part of infrastructure. Table 4.1 exemplifies just how noisy the data is and how many devices in a data set are not related to infrastructures. However, we were able to identify 2 device clusters that we believe were part of some infrastructure,

One of these clusters was the cluster that saw devices sharing the name "Bluetooth USB Host Controller." This cluster was found near the financial infrastructures, with 10 out of the 11 devices found near the *atm* and *bank* place type. When we further investigated the devices by inputting their locations into Google maps, we found that all 11 devices were near grocery shops as well.

However, 11 devices is too small of a sample size, so we decided to query the device prefix (60:02:b4) and name using WiGLE to see what data they had on the devices. We randomly sampled a few of WiGLE's results whose QoS were 4 or above and found that all the Bluetooth USB Host Controller devices were near grocery stores. This confirms to us that the "Bluetooth USB Host Controller" devices are related to grocery stores, but we are unsure of their purpose. While further looking into "Bluetooth USB Host Controller" devices, we also found that devices of the same name with a prefix 28:24:ff are located in Walmarts. Like the previous "Bluetooth USB Host Controller" devices, we are unsure of this device's functionality and the reason for their placement in Walmarts.

The other cluster we believed that was part of an infrastructure was the cluster where the device names started with "PLREP." These "PLREP" devices were found near the place types *parking* and *doctor*. However, we ruled out the *doctor* place type after searching for "PLREP" devices on WiGLE since it seemed that the *doctor* place type was a one time occurrence. Moreover, we believe that the first two letters "PL" stand for parking lot. When we looked at more examples of these "PLREP" devices near parking lots, we found that most of them had car counters. Therefore, we concluded that these devices were parking lot car counters that keeps



(a) A "PLREP" device found on the WiGLE map.

(b) The blue parking lot counter on the left side of the image is what we believe the device to be.

Figure 4.1. The location of a "PLREP" device and what we believe the device is.

track of how many cars there are to report on the number of free parking spots. Figure 4.1a shows a "PLREP" device on the WiGLE map and Figure 4.1b shows what we believe to be the device at that location.

4.2 WiGLE Data

While working with Bluetana's data set, we realized that it did not provide us with enough data to continue working, so we switched to WiGLE's database. We also decided to use the Jaro-Winkler method to cluster devices by names as the original method was too stringent on clustering device by names.

4.2.1 Infrastructural Clustering

Intersection

Associating device clusters to intersections produced intriguing results as not all the clusters we discovered with a high percentage of devices near an intersection were expected. An example of an expected cluster was the pedestrian crossing push buttons. Figure 4.2 shows what a few pedestrian crossing push buttons with Bluetooth capabilities look like. Users with the associated app may use it to push the pedestrian crossing push button without physically touching it. According to Polara, one of the major manufacturers of this technology, the app can

Figure 4.2. Pedestrian crossing push button that contains Bluetooth.

also provide audio functionalities for the visually impaired in case the pedestrian crossing push button does not give audio cues.[1]

The WiGLE data set was rife with pedestrian crossing push button entries, and many of these entries had specific letterings in their device names. Device names that contained the shorthand version of cardinal directions (“nw”, “sw”, “ne”, “se”), “PED” — which we assume to be shorthand for pedestrian — or repeating portions of the name such as “A1A1” or “C3C3” were found to be all pedestrian crossing push buttons. Unfortunately, we cannot provide the percentage of all pedestrian crossing push buttons near an intersection since we are unsure if we found all clusters that are pedestrian crossing push buttons, but from the clusters that we do know, the percentage is between 80%-100%.

Another group of clusters is the cluster where the devices started with “P” followed by 7 digits. Although we could not definitively conclude what these devices were, we believe they are intricately tied with traffic lights. In fact, 93% of the devices in these clusters were near intersections where there were traffic lights. We ruled out the possibility that these devices

[1]https://polara.com/ins-inavigator-accessible-pedestrian-signals

Figure 4.3. A recloser control box in LA found on WiGLE that was created by SEL.

are part of the pedestrian crossing push button as not all the intersections of these devices had a Bluetooth pedestrian crossing push button as seen in Figure 4.2, and some didn't have any pedestrian crossing push buttons at all.

One of the more surprising cluster that was near intersections is the cluster involving reclosers. Reclosers are automatic electric switches on utility poles that maintain the power line. They are responsible for switching off the power in case of power line failures such as the power line short-circuiting and restoring the power once the problem is fixed. Reclosers are not widely associated with intersections; while they are found on utility poles that run along roads, there is no expectation for them to be near intersections. However, according to the WiGLE data and the GeoNames API, around 64% of the reclosers were near intersections.

The recloser clusters began with the letters "BLUE" followed by 13 digits. We concluded that these devices were reclosers after manually surveying these devices' geo-coordinates and linking them to the data found in the FCC (Federal Communications Commission) database regarding Bluetooth devices. When we manually surveyed the devices, we saw Figure 4.3, which is a recloser controller box in Los Angeles, California with the word SEL on it. SEL-2924 and

Table 4.2. Summary of the interesting clusters found when filtering by intersections and the percentage of the devices in the cluster near the intersection. x represents a digit 0-9.

Cluster	Percentage
Pedestrian Crossing Push Buttons	80%-100%
Pxxxxxxx	93%
BLUExxxxxxxxxxxxx	64%
CITIX	92%
EV	54%
SP	53%

SEL2925 are Bluetooth serial adapters whose default device name is BLUE*xxxxxxxxxx* where *x* represents any digit 0-9. [14]

The pedestrian counter cluster "CITIX" was also somewhat an unexpected discovery. "CITIX" is made by the company Eco-Counter that specializes in pedestrian and bicycle counters.[2] 92% of "CITIX" devices were located near intersections, but this is within reason as most people cross intersections to enter malls and shopping centers, where most of the devices are found. Surprisingly, a few of these devices were also found near libraries.

Yet another surprising but reasonable cluster near intersections involves speed signs, specifically devices that began with "SP" or "EV". We believe the "SP" in the device names is short for "SafePace", which is a radar speed sign designed by TrafficLogix and the "EV" in the device names stands for "Evolution", which is another type of radar speed sign from the same company TrafficLogix. 53% of the devices from cluster "SP" are near intersections and 54% of the devices from cluster "EV" are as well.

Airport

While filtering by intersections gave some relevant device clusters that could be part of intersections, filtering by airports was less successful. In total, we found 3 clusters of devices that could be related to airports, but highly unlikely due to our speculation of what these devices could be. These 3 clusters are clusters where the device names began with "XXRAJ", "SLA" and

[2] https://www.eco-counter.com/company/about-us/

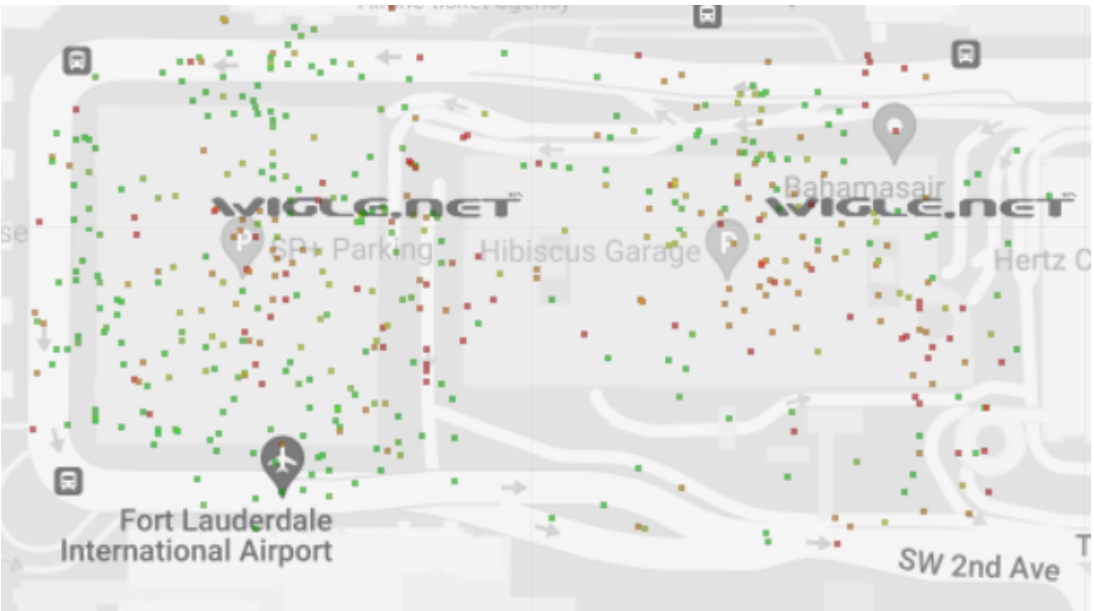


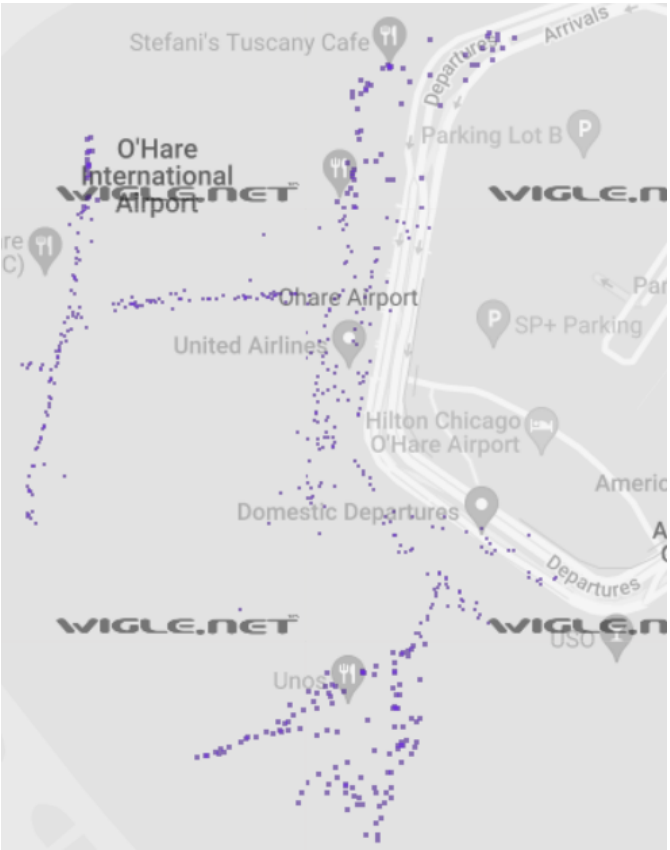
Figure 4.4. "SLA...AAA==" devices found in Fort Lauderdale-Hollywood International Airport.

ended with "AAA==", and "5AAA" followed by a mixture of 25 random symbols, characters, or numbers.

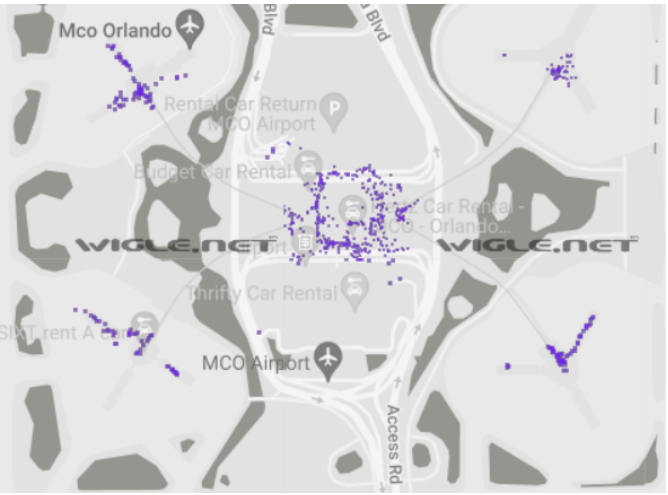
For the "XXRAJ" cluster, 31% of devices were found to be near airports. We found 2 plausible technologies that these devices could be. One of these is a parking enforcement technology that enables cities to monitor parked cars, citing them if they go over the time allotted for them to park at a parking structure of meter.[3] The other is a handheld printer called the Zebra Printer.[4] According to [3], the handheld printer can be used to print out parking ticket for cars that overstay their time. Although handheld printers would not be labeled as stationary by a person, it is possible these devices never leave a 50 m radius and thus, are considered "stationary" by our filtering algorithm. Nonetheless, neither of these options are exclusively bound to airports.

For the "SLA...AAA==" and "5AAA" devices, the former devices littered the parking structures in the Fort Lauderdale-Hollywood International Airport in Florida and the latter devices were found scattered around the terminals in the O'Hare International Airport in Illinois and Orlando International Airport in Florida. Figure 4.4 shows the "SLA...AAA==" devices in

[3]https://github.com/ggerts/BLE-Parking-Enforcement-Detector/blob/master/BLE_inspector.py
[4]https://chms.pushpay.com/s/article/How-do-I-set-up-a-Zebra-printer-for-the-Check-In-App



(a) "5AAA" devices found in O'Hare International Airport.



(b) "5AAA" devices found in Orlando International Airport.

Figure 4.5. The location of all the "SLA...AAA==" and "5AAA" devices in their respective airports.

Table 4.3. Summary of the interesting clusters found when filtering by airports and the percentage of the devices in the cluster near the intersection.

Cluster	Percentage
XXRAJ	31%
SLA...AAA==	65%
5AAA	95%

the Fort Lauderdale-Hollywood International Airport's parking structure, Figure 4.5a and Figure 4.5b shows the "5AAA" devices in the O'Hare International Airport and Orlando International Airport respectively. Although we were unable to ascertain what these devices were used for, from the figures, we can see from the figures that "SLA...AAA==" littered the parking structures of the Fort Lauderdale-Hollywood International Airport and "5AAA" were scattered around the terminals of O'Hare and Orlando International Airport and thus, we can conclude that the "SLA...AAA==" devices in airports are probably related to parking infrastructures and "5AAA" devices are related to shopping infrastructures.

4.2.2 Geographical Clustering

With the geographical clustering by US states method, we found 4 clusters we believe are related to some type of infrastructure. The first of these device cluster is the cluster where the device names begin with "YCH". 9 out of the 10 "YCH" stationary devices were located in Wisconsin, with the other device located in Florida. These devices were found near "CITIX" devices — and in turn found near malls and shopping centers — so we believe they are also pedestrian or vehicle counters. However, one can also argue that because the shopping center already has the "CITIX" pedestrian and bicycle counter, "YCH" must be a different type of stationary Bluetooth device. Either option is more reasonable than earphones, which is the top search result for "YCH" Bluetooth devices and is not stationary.

The second device cluster is the alleged capacitor clusters found only in Florida. They are labeled alleged as there is no evidence that they are capacitors other than the infrastructure near them (utility poles) and their device names. Devices in this cluster had names that began

Table 4.4. Summary of the interesting clusters found when filtering by US states and the percentage of the devices in the cluster near the intersection. x represents a digit 0-9.

Cluster	State	Percentage
YCH	Wisconsin	90%
cap	Florida	100%
xxVCxxxx	Tennessee	59%
FireFly	Kentucky	73%

with “cap”, “cap#”, “capp”, and “cappp” followed by 4 digits. Capacitor bank controls with Bluetooth capabilities do exist, but we were not able to find any solid evidence that linked these devices to capacitors.

The “VC” device clusters are in the same boat as the capacitor clusters. These clusters’ device names come in the form *xx*VC*xxxx* where *x* represents digits 0-9. While they are found mainly in Tennessee (13 devices), they can also be seen in Florida (1 device) and Minnesota (8 devices). Like the capacitor cluster, there is no evidence that these devices are related to the smart grid other than the fact that they are near utility poles.

The final device cluster is the “FireFly” cluster where 30 out of the 41 devices in the cluster were located in Kentucky. These “FireFly” devices use Grid Connect’s FireFly serial to Bluetooth adapter and thus, can be used in many different situations. However, like the P*xxxxxxx* device discussed in Section 4.2.1, we are fairly certain that the “FireFly” devices in Kentucky are used for in traffic light-related technology as while there were many other device types near it such as pedestrian crossing push buttons and utility poles, more often than not the devices were located near traffic lights than the other infrastructures.

Chapter 5

Discussion

5.1 Limitations and Challenges

Despite both data sets containing a large number of data points, both had similar limitations that prevented us to be fully confident in our results. A shared issue between the two data sets is the geo-coordinate accuracy issue. Recall from Section 3.1 that we only kept entries where the *geo_accuracy* was less than 5. The WiGLE data, however, does not have the *get_accuracy()* feature, so instead we use the QoS data provided for each device. We drop devices that have a QoS less than 4. In this case, Bluetana lost 31,173 out of 63,987 unique devices and WiGLE lost 301,164 out of 332,036 unique devices. Furthermore, a nontrivial chunk of the data were non-stationary devices. After filtering out the non-stationary devices, the Bluetana data set lost an additional 30,537 entries and the WiGLE data set lost 19,396 unique devices. While the remaining number of devices are still in the thousand range, this value is before clustering. Furthermore, scanning for devices through wardriving is fickle, and many devices that were part of the infrastructure but were only scanned once would have been removed using our method of identifying infrastructure-related devices.

Another limitation was the lack of names for many of the devices. 8.6% of Bluetana's data had no device name and 31.8% of WiGLE's data had the same infliction. Because of the substantial amount of devices without a name that we were losing out on if we did not cluster them, we used the unconventional and unverified method of placing non-name devices with the

cluster whose device had the closest MAC address to the non-name device. While studies such as [8] and [1] discuss the poor MAC randomization of mobile devices, we do not know if that directly translates over to all the devices in the two databases.

A challenge and time-consuming task unique to WiGLE was the daily API query limit. The WiGLE Search API did not cause too much trouble, though for a few MAC address prefixes, the Search API hit the daily limit, so we had to run the Search API multiple times on those MAC address prefixes. What *did* give us troubles was the WiGLE Detail API. In total, 30,872 devices from WiGLE matched our MAC address prefixes and had a QoS greater than or equal to 4, but the Detail API could only provide information on 500 MAC addresses a day. This number was later bumped up to 2,000, but even then, if we started calling the Detail API with 2,000 daily queries from the start, it would take more than 15 days to finish querying all 30,872 devices.

5.2 Security and Impact

Consumer wireless devices value ease of use as it is directed towards the average person, which leave them open to new attack vectors. Literature [7] addresses this issue in regards to Bluetooth, pointing out that while Bluetooth allows users the convenience and ease of using it, it also lacks a centralized security infrastructure, resulting in many security vulnerabilities that can be exploited. While there are countermeasures to prevent and combat the mentioned vulnerabilities, there are many more vulnerabilities to be discovered. Furthermore, Bluetooth isn't the only wireless technology out there. In 2014, Cesar Cerrudo found a way to theoretically manipulate the traffic lights in cities like Seattle, New York, and Washington, DC. [4] These cities used traffic light sensors that followed the Sensys NanoPower Protocol, which lacked data encryption and authentication. Then in 2020, 2 researchers from Netherlands discovered how to manipulate the traffic lights without being near them. In their DefCon talk [11], Wesley Neelen and Rik van Duijn described their method: Realizing they could reverse engineer the Android app and bypass the authentication step (since there was no authentication process in the first

place), they used a Python script and sent spoofed CAM (cooperative awareness message) data to the traffic light controller.

Now imagine the above scenario for the FireFly traffic light controllers in Kentucky. It is highly likely that the FireFly traffic light controllers uses the same procedures as each other to communicate as, mentioned in the previous paragraph, convenience is a selling point of installing Bluetooth into everyday objects. Therefore, it will only take an attacker to crack one of the FireFly traffic light controller to manipulate the 29 other known controllers. Although 30 does not seem like much, remember that 30 is the minimum number of traffic light controllers that use the FireFly Bluetooth to serial adapter. Furthermore, the major concern should be that Kentucky FireFly traffic controllers are on a unprotected scannable network; major infrastructure devices should not be seen in the first place.

Additionally, reclosers are also a security concern. [10] cautions about malicious actors who can potentially connect to smart grid devices through Bluetooth to wreak havoc and brings up reclosers as an example. Literature [13] confirms this belief, providing a possible attack vector that attackers could use to take control of reclosers. This is a major concern as reclosers are electric switches responsible for turning the power of a power line off or on depending on the condition of the power line. Literature [13] further explains that the attacker can take control of the recloser controller which is the gateway for configuring the recloser's security settings and maintenance. With control over the controller, the attacker can shut down the recloser and turn off the power for those connected to the power line. For some, living without power for a few hours is a minor convenience, but for others, it can end in death. In 2018, a New Jersey woman died when the utility company turned off the electricity for a day, causing her electricity-powered oxygen tank to stop working. [5] Although it is not a common story, it does shine a light on the potential damages and harm a malicious actor can cause if the controlled the recloser.

Chapter 6

Conclusion

In this paper, we found that it was possible to identify infrastructure-related devices by clustering and filtering wardriving data. We were able to reduce the number of devices to manually examine by 96% for Bluetana and 94% for WiGLE. From our experiment, we were able to discover devices that were part of the traffic light, smart grid, and parking infrastructures as well as understand the impact and security concerns of some of these infrastructures. Although our methodology produced devices that were part of an infrastructure, it is far from perfect. Coupled with the noisy and unreliable data, many devices — some that were probably part of an infrastructure but did not have enough entries or good QoS — were removed. However, we believe our methodology is a good start into identifying infrastructure-related devices using noisy wardriving data.s

6.1 Future Work

While we have tested many clustering and filtering heuristics on a large number of data, there are other heuristics and data we have yet to look into. Though the current MAC address prefix list is extensive, there are still many more prefixes out there that represent Bluetooth to serial adapters. We would like to run our filtering and clustering methods on these new prefixes.

Furthermore, we would like to examine the prevalent devices near hospitals. We began looking into hospitals, but due to the time limit, we were unfortunately unable to complete it in

time. The data set we used is provided by ArcGIS.[1] We believe hospitals to be a good addition as they seem to contain many devices that use Bluetooth. Identifying what devices use Bluetooth and the impact if the device is compromised would be an interesting addition to our work.

Another clustering method we would like to test is clustering by the device name and the first time a device is seen in a certain region. Devices related to the infrastructure tend to go live at the same time. If we see a certain device cluster where the device's first seen date are grouped together, the chances of that cluster being part of an infrastructure would be quite high.

Outside of data and clustering heuristics, we would like to study if pedestrian crossing push buttons can be used to manipulate the timing of traffic lights. Hacking traffic light controllers is a popular topic in DefCon, but to the best of our knowledge, there are no works on taking advantage of pedestrian crossing buttons.

[1] https://hifld-geoplatform.opendata.arcgis.com/maps/6ac5e325468c4cb9b905f1728d6fbf0f

Appendix A

List of MAC Address Prefixes

- 00:06:66
- 68:27:19
- 04:91:62
- 34:15:13
- 00:03:19
- 00:17:53
- 00:22:58
- b0:b4:48
- 00:13:04
- 24:71:89
- cc:78:ab
- 74:6f:f7
- 0c:a6:94
- 00:a0:96
- f0:ab:54
- a4:d5:78
- 00:25:ca
- 00:0b:57
- 8c:de:52
- 42:93:1a
- 42:93:1a
- 00:0e:0e
- 98:d3:31
- 98:d3:32
- 98:d3:35
- 20:10:01
- 20:10:02
- 20:10:03
- 20:10:04
- 20:10:05
- 20:10:06
- 20:10:07
- 20:10:08
- 20:10:09
- 20:10:10
- 20:10:11
- 20:10:12
- 20:11:01
- 20:11:02
- 20:11:03
- 20:11:04
- 20:11:05
- 20:11:06
- 20:11:07
- 20:11:08
- 20:11:09
- 20:11:10
- 20:11:11
- 20:11:12
- 20:12:01
- 20:12:02
- 20:12:03
- 20:12:04
- 20:12:05
- 20:12:06
- 20:12:07
- 20:12:08
- 20:12:09
- 20:12:10
- 20:12:11
- 20:12:12
- 20:13:01
- 20:13:02
- 20:13:03
- 20:13:04

- 20:13:05
- 20:13:06
- 20:13:07
- 20:13:08
- 20:13:09
- 20:13:10
- 20:13:11
- 20:13:12
- 20:14:01
- 20:14:02
- 20:14:03
- 20:14:04
- 20:14:05
- 20:14:06
- 20:14:07
- 20:14:08
- 20:14:09
- 20:14:10
- 20:14:11
- 20:14:12
- 20:15:01
- 20:15:02
- 20:15:03
- 20:15:04
- 20:15:05
- 20:15:06
- 20:15:07
- 20:15:08
- 20:15:09
- 20:15:10
- 20:15:11
- 20:15:12
- 20:16:01
- 20:16:02
- 20:16:03
- 20:16:04
- 20:16:05
- 20:16:06
- 20:16:07
- 20:16:08
- 20:16:09
- 20:16:10
- 20:16:11
- 20:16:12
- 20:17:01
- 20:17:02
- 20:17:03
- 20:17:04
- 20:17:05
- 20:17:06
- 20:17:07
- 20:17:08
- 20:17:09
- 20:17:10
- 20:17:11
- 20:17:12
- 20:18:01
- 20:18:02
- 20:18:03
- 20:18:04
- 20:18:05
- 20:18:06
- 20:18:07
- 20:18:08
- 20:18:09
- 20:18:10
- 20:18:11
- 20:18:12
- 20:19:01
- 20:19:02
- 20:19:03
- 20:19:04
- 20:19:05
- 20:19:06
- 20:19:07
- 20:19:08
- 20:19:09
- 20:19:10
- 20:19:11
- 20:19:12
- 20:20:01
- 20:20:02
- 20:20:03
- 20:20:04
- 20:20:05
- 20:20:06
- 20:20:07
- 20:20:08
- 20:20:09
- 20:20:10
- 20:20:11
- 20:20:12
- 00:14:03

Bibliography

[1] Johannes K Becker, David Li, and David Starobinski. Tracking anonymized bluetooth devices. *Proc. Priv. Enhancing Technol.*, 2019(3):50–65, 2019.

[2] Nishant Bhaskar, Maxwell Bland, Kirill Levchenko, and Aaron Schulman. Please pay inside: Evaluating bluetooth-based detection of gas pump skimmers. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 373–388, Santa Clara, CA, August 2019. USENIX Association.

[3] Knowledge Center. How to set up a zebra printer for the check-in app. https://chms.pushpay.com/s/article/How-do-I-set-up-a-Zebra-printer-for-the-Check-In-App, 2017.

[4] Cesar Cerrudo. Hacking traffic control systems. https://youtu.be/_j9lELCSZQw, 2014.

[5] Matthew Haag. New jersey woman on oxygen dies after electric company shuts off her power. https://www.nytimes.com/2018/07/09/nyregion/woman-dies-oxygen-tank-electricity.html, 2018.

[6] Matthew A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[7] Angela M. Lonzetta, Peter Cope, Joseph Campbell, Bassam J. Mohd, and Thaier Hayajneh. Security vulnerabilities in bluetooth technology as used in iot. *Journal of Sensor and Actuator Networks*, 7(3), 2018.

[8] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C. Rye, and Dane Brown. A study of MAC address randomization in mobile devices and when it fails. *CoRR*, abs/1703.02874, 2017.

[9] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. Iot sentinel: Automated device-type identification for security enforcement in iot. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2177–2184, 2017.

[10] Ken Munro. Bluetooth + electrical switchgear. https://www.pentestpartners.com/security-blog/Bluetooth-electrical-switchgear/, 2018.

[11] Wesley Neelen and Rik van Duijn. Hacking traffic lights. https://youtu.be/L9UUD3a7xP4, 2020.

[12] Jorge Ortiz, Catherine Crawford, and Franck Le. Devicemien: Network device behavior modeling for identifying unknown iot devices. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, IoTDI '19, page 106–117, New York, NY, USA, 2019. Association for Computing Machinery.

[13] V. S. Rajkumar, A. Stefanov, S. Musunuri, and J. de Wit. Exploiting ripple20 to compromise power grid cyber security and impact system operations. In *CIRED 2021 - The 26th International Conference and Exhibition on Electricity Distribution*, volume 2021, pages 3092–3096, 2021.

[14] Inc. Schweitzer Engineering Laboratories. Manual (sel-2924/sel-2925 bluetooth® serial adapters). https://fccid.io/ANATEL/02908-13-07001/Manual/750F8E42-49B0-4BFC-AE70-0130C29FD1CD/PDF, 2017.

[15] Rahul Anand Sharma, Elahe Soltanaghaei, Anthony Rowe, and Vyas Sekar. Lumos: Identifying and localizing diverse hidden IoT devices in an unfamiliar environment. In *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, August 2022. USENIX Association.

[16] Chao Shen, Ruiyuan Lu, Saeid Samizade, and Liang He. Passive fingerprinting for wireless devices: A multi-level decision approach. In *2017 IEEE International Conference on Identity, Security and Behavior Analysis (ISBA)*, pages 1–6, 2017.

[17] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying iot devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, 2019.