Privacy Breaching Weather Application

Christopher Cartagena
University of Arkansas
Fayetteville, Arkansas 72762
Email:christophercartagena40@gmail.com

***Abstract*—Most phones today offer a wide amount of information that can be gathered, helped in part by their growing number of sensors, and a lack of administration for them. GPS is another tool that is now common place in phones that can, and is frequently targeted when trying to breach privacy. While the user does get notified when an app is going to attempt to get their location, and they must agree to this , attackers can still create legitimate looking applications that provide expected services, but also gather more data than they should. My project aims to address the problem by showing off an attack where I correlate a users college major by gathering sensor, and location data, analyzing it.**

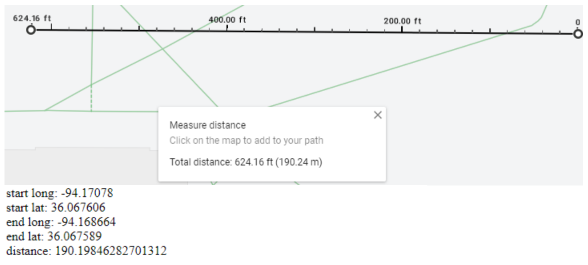
1. Introduction

I have devised a project where I will create a simple weather application that will retrieve the location information provided through GPS, as well as the accelerometer sensor readings to infer a targets major at the University of Arkansas. The two main points that will be analyzed using this data is first, what building the user has been in, and second how inactive have they been while there. I have evaluated my result in the following manner, I gather data on myself as I go through my classes, and then using my attack I check to see if I can correctly guess what major I am. I also go to all the other buildings to see if I can detect a user being inside of the building. My contributions are that I create a framework in which an attacker can retrieve both GPS and accelerometer data, and then plot the information as a report. This gives an attacker a way to interpret the data themselves in case my algorithm cannot decide on a major.

2. Background

Some basic background information is that the accelerometer sensor retrieves the current acceleration of your device in any given direction, it usually consists of a 3-axis reading in most phones. Each axis has a negative direction and a positive direction. Laying your phone down means that the acceleration that is acting on it is the approximate 9.8 meters per second squared that the earth generates. Changing the orientation of your phone redistributes the acceleration of the earth across these 3-axis, once the phone comes to a rest. GPS information is retrieved in terms of two coordinates and they are latitude and longitude. These coordinates can be positive or negative, in the context of Fayetteville, Arkansas the latitude will be positive with an approximate reading of around 36 and the longitude will be negative with an approximate reading of around -94.

I have decided to retrieve the location of the user using their network provider rather than GPS provider, this is because it uses less energy and is able to give you a faster result with the downside of having to sacrifice accuracy. The location that is reported will depend on what network the user has, the options being his cell phone companies network in which case the location will be retrieved with the help of cell towers, or an access point in which case the location retrieved will be that of the access point they are connected to. I use the haversine function to calculate the distance between two points, I have tested it out and it compares to the distance measured used by google maps with a very small difference which can be seen in figure 1.



Total distance: 624.16 ft (190.24 m)

```
start long: -94.17078
start lat: 36.067606
end long: -94.168664
end lat: 36.067589
distance: 190.19846282701312
```

Figure 1. Differences between using Google maps(left) and the haversine function(right).

3. Related Works

There has been a recurring amount of work done into the ability to infer a users privacy based on sensors and locations. A recent paper has shown an attack on users who have wearable health technologies. They can differentiate between different activities based on the accelerometer readings on their phone [1]. An attack on inferring a users password using onboard sensors shows the variety of inference attacks that are possible [2]. In this work an accelerometer was used to differentiate a tap event on the phone screen vs any other movement. A phones orientation sensor was used as well to create signatures of a user tapping on various parts of the phones screen. [3] showed an attack using location, as well as more context information such as the time, to predict a users activity.

4. System Model

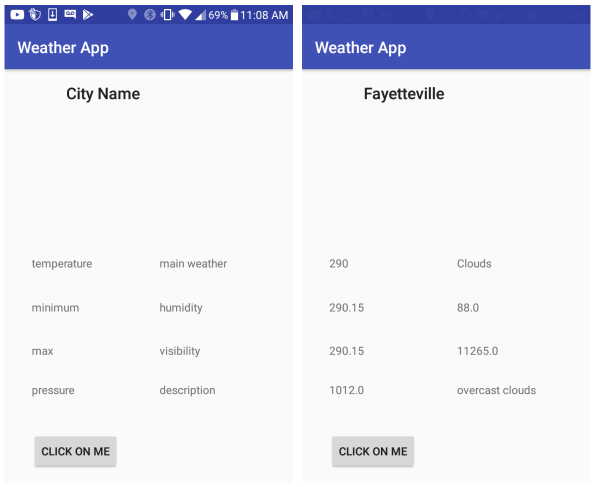
The attack is based on the ability for me to get a targets location, and accelerometer readings. The latter is not a problem to retrieve, the former needs permission from the users. I will try to achieve permission by designing an application that has an obvious reason to retrieve the current location. An assumption is made that the users does not force close the application after they are done, rather that they navigate away from it, which will allow the application to continue retrieving the location, and accelerometer reading. The user would also need to restart the application every time they restart their phone, however Im assuming this does not happen very often.

5. Overview

An abstract view of my infrastructure is a typical client server model to retrieve information from the user, the client in this case being an android phone, while the server will be my desktop. Data retrieved will be analyzed on the server, and a GUI will be provided that will show the results to the attacker in the form of a webpage.

6. Design

In the first step of my attack I created an android application that tells a user what the current weather in their city is figure 2. This is to have a valid reason to retrieve the users location, and for the user to have less suspicion when they see that their location is being retrieved, the weather information comes from OpenWeatherMap[6] .In the case of android phones this is shown in the form of a pin in the top section of the phone as shown in figure 3.



City Name

temperature	main weather
minimum	humidity
max	visibility
pressure	description

Fayetteville

290	Clouds
290.15	88.0
290.15	11265.0
1012.0	overcast clouds

Figure 2. Weather app interface layout, and live screenshots. All temperature readings are in kelvin

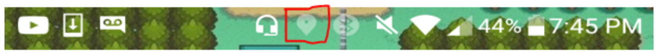
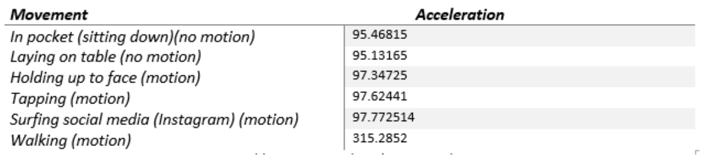


Figure 3. Android Reporting Location Retrieval

Once my application can retrieve the location of the user I set it to send the location information every minute to the server. This server gets the data and runs a php script to store that information into a MySQL database. The application will continue to retrieve the users location for as long as the user has not forcefully exited the application. Apart from the location, the application is also set to retrieve the accelerometer readings of the user, this is done in a different manner then the location, and it is because the sensitivity of the accelerometer can vary greatly.

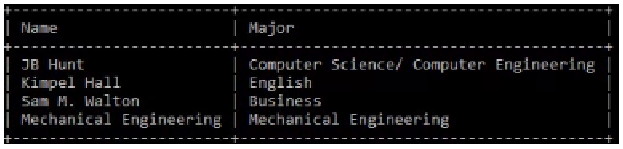
To make it easier to analyze later I programmed the accelerometer to give back a different reading only if the total movement exceeds a threshold of 1.4 m/sˆ2. The choice of 1.4 m/sˆ2 is made after testing the accelerometer, it was the value that was the smallest in terms of going from a no motion to an active motion activity. The results of these test are in table 1, the accelerometer data was put through the formula M = (Xˆ2+Yˆ2+Zˆ2), where X, Y, and Z correspond to the acceleration in that axis, the values are squared to make them all positive and are added together to get an absolute value of the movement detected. After this filtering, the readings are sent to the server approximately every 20 seconds and stored using a similar procedure as the location.



Movement	*Acceleration*
In pocket (sitting down)(no motion)	95.46815
Laying on table (no motion)	95.13165
Holding up to face (motion)	97.34725
Tapping (motion)	97.62441
Surfing social media (Instagram) (motion)	97.772514
Walking (motion)	315.2852

Figure 4. Testing activities that phone records

Furthermore, I chose only a subset of all the buildings at the University of Arkansas and associated them with the most prevalent major in that building these being shown in figure 4.



```
+---------------------+--------------------------------------+
| Name                | Major                                |
+---------------------+--------------------------------------+
| JB Hunt             | Computer Science/ Computer Engineering |
| Kimpel Hall         | English                              |
| Sam M. Walton       | Business                             |
| Mechanical Engineering | Mechanical Engineering            |
+---------------------+--------------------------------------+
```

Figure 5. Building and Majors

I also needed to retrieve the coordinates of the buildings, I first thought about associating only a single point and comparing the users location with this point. This however brought up the issue of at what distance is the user considered to be in the building, and the problem of having to compare it against a circular distance in which the building is represented by. Since building are most often shaped in a rectangular shape when seen from above I decided to instead gather 4 coordinates that would represent the borders of what I would consider to be inside. I did this using google maps as shown in figure 5.

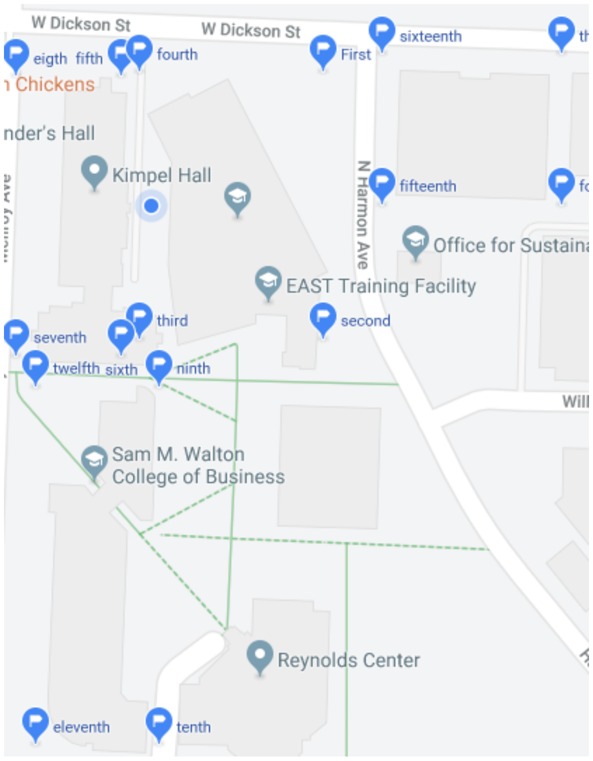
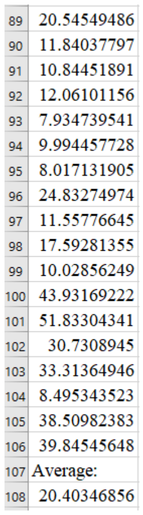


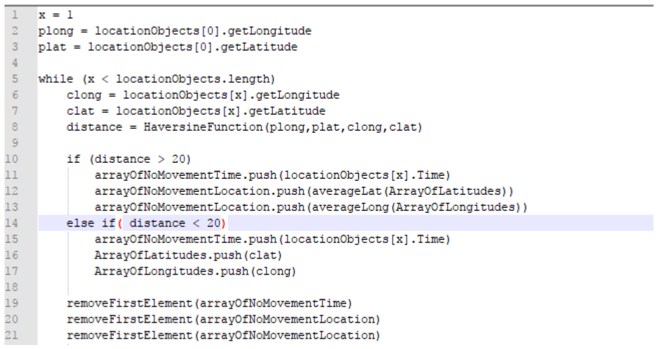
Figure 6. xample is JB Hunt represented by First, Second, Third and Fourth.

After the retrieval of user data and mapping the buildings, I then go into the process of querying the database and analyzing the data. My first step is determining if the user has moved, I do this with the algorithm depicted in Figure 7. What it does is that it checks the distance between two coordinates using the Haversine function, and if it is greater than 20 meters than it will store the average latitude and longitude, and this will be considered a no movement reading. I chose 20 meters as my threshold because after some testing I found that when a user stayed in the same location the amount the distance would vary is about 20.4034, so I rounded down. Figure 6 shows the actual readings.



89	20.54549486
90	11.84037797
91	10.84451891
92	12.06101156
93	7.934739541
94	9.994457728
95	8.017131905
96	24.83274974
97	11.55776645
98	17.59281355
99	10.02856249
100	43.93169222
101	51.83304341
102	30.7308945
103	33.31364946
104	8.495343523
105	38.50982383
106	39.84545648
107	Average:
108	20.40346856

Figure 7. Live Distance Readings average



```
x = 1
plong = locationObjects[0].getLongitude
plat = locationObjects[0].getLatitude

while (x < locationObjects.length)
    clong = locationObjects[x].getLongitude
    clat = locationObjects[x].getLatitude
    distance = HaversineFunction(plong,plat,clong,clat)

    if (distance > 20)
        arrayOfNoMovementTime.push(locationObjects[x].Time)
        arrayOfNoMovementLocation.push(averageLat(ArrayOfLatitudes))
        arrayOfNoMovementLocation.push(averageLong(ArrayOfLongitudes))
    else if( distance < 20)
        arrayOfNoMovementTime.push(locationObjects[x].Time)
        ArrayOfLatitudes.push(clat)
        ArrayOfLongitudes.push(clong)

    removeFirstElement(arrayOfNoMovementTime)
    removeFirstElement(arrayOfNoMovementLocation)
    removeFirstElement(arrayOfNoMovementLocation)
```

Figure 8. Algorithm detecting user movement

In my experimentation with test data I noticed that sometimes there would be a glitch in the location data that would end up calculating that the user had moved when in fact they had not. I attempted to fix this by gathering up the data where movement was detected and checking the accelerometer reading. If for any span the accelerometer had not detected any changes then I would gather up these time spans in an array, and re-analyze the original no movement results. As it stands I only re-analyze the time spans immediately ending an original no movement, if any are found then the time span is increased to the ending of the new no movement calculation done with the accelerometer. The average latitude and longitude are not changed in this process because as stated above these new timestamps had a glitch in their location. After merging the original no movement detections with the new ones, I collapse the original array with itself, this being because of the possibility that the new array was able to fill holes in the original. The results are stored into the MySQL database to more easily pass into the next process.

The analysis was done over the previous calculated no movement data. I first filtered out the results that yielded no movement for over 50 minutes this is because classes at the University of Arkansas are usually at least 50 minutes long. For each one I first check to see if the reported location corresponds to a building, if so I query the accelerometer readings that corresponded with that time span. For each entry of accelerometer reading I calculate the total movement with the formula M=(X^2+Y^2+Z^2). I take an average of all the accelerometer readings, and if they show an average movement of below 97 m/s^2, this because the movement activities were all above 97 m/s^2, then I assert that they are likely in a class, I plot the accelerometer readings on a line graph using plotly[5]. I then analyze the rest of the no movement data that was below the 50-minute threshold. I add up the length of times that correspond to each building and plot them on a bar graph [5]. If the potential classes building, and the building with the longest time both are the same, then I will infer that their major is the one that is housed by that building. Otherwise nothing is asserted, but based on the graphs an attacker could decide on the major.

7. Evaluation

To evaluate my attack, I took two test cases. In the first test case I used my own data, which was to say that I opened the application at 8:30 A.M. and went through my usual day, and then at 5:30 P.M. I force closed the application. After analysis, my attack was correctly able to assert that my major was in the CSCE department, although it did not know which of the two it was. The results of my analysis are shown in figure 8, at the top of the figure it shows my inferred major. The first part of the image shows a line graph of a time span where it was determined it correlated with a class, which is 50 minutes of being in the same location. The second attribute that is asserted is whether there was less than a 97 in movement, which corresponds to all the acceleration in each axis being squared and added together. The result on the graph shows that my acceleration in this timespan was about 96.425. So, with the timespan being greater than 50 minutes, and the acceleration being less than 97, then it is asserted that a class was detected. Before the graph, the building corresponding to the timespan is shown, as well as the average latitude and longitude the user maintained, and the starting and ending time periods.

The second part of the figure shows a bar graph of the aggregation of all the time spans that were less than 50 minutes, and the buildings they corresponded to, in my case it was JB Hunt. It also shows some time in none, this corresponds with anytime spent anywhere else besides these four buildings. In the results it did not detect the right time span when I had class that day because that was from 12:55 1:45, the class it detected was me sitting still and working on an assignment.

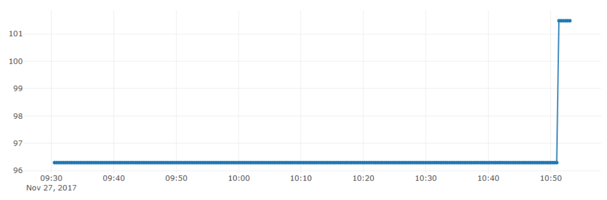
In my second case I again turned my application on from 8:30 A.M. to 5:30 P.M., but this time instead of going through my normal day I went to all the buildings I look for, these being Walton, Kimpel, Mechanical Engineering, JB Hunt and sat there for more than 50 minutes to see if it detected that I was in a class. The results of this case are shown in figure 9. At the very top of the image you can see that it shows my major as being undetermined, this is because the building it detected as having the most classes, was not the same building with the maximum amount of aggregate time. The former being JB Hunt and the latter being the None option which meant I spent more accumulated time in none of the buildings.

In figure 9 the first class it detected was in mechanical engineering, the spikes in the line graph are due to me getting up and walking away. The second class it detected was in JB Hunt, however this result is wrong because I was in Kimpel Hall. While in Kimpel Hall I checked my location in google maps, and it too asserted that I was in JB Hunt, Im not entirely sure why it thought this. Overall Kimpel Hall gave me the worst results in terms of being able to detect that I was in the building. When I was connected to the UARK Wi-Fi it would show I was in JB Hunt, and when on my providers network it would show I was next to Kimpel Hall. The next class time span that was detected was in JB Hunt, this one however was correct, but it shows that a class was not detected. This having to do with the average acceleration exceeding the threshold of 97 m^2/s^4. It should have detected class, so this is an error in my class detection, although an attacker would be able to determine that the victim is most likely in class because the acceleration stays at the same level. I also went into Walton, but it couldnt keep the entire 50 minutes that I spent in a single time stamp, it ended up splitting it up. So, it doesnt show up as a class, but rather as part of the aggregated less than 50-minute time bar graph at the very bottom.

Predicted major is: Computer Science/ Computer Engineering

Start: 2017-11-27 09:30:13,End:2017-11-27 10:53:27
Coordinate: (36.066331756443,-94.174071839935)
Building: JB Hunt
Major: Computer Science/ Computer Engineering

2017-11-27 09:30:13 - 2017-11-27 10:53:27 Class Detected, Average Acceleration: 96.42571081012625



Less than 50 Minutes Total (in hours)

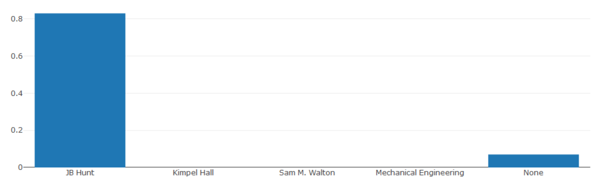


Figure 9. My own results

Rest of Building Evalutations

Predicted major is: Undetermined

Start: 2017-11-29 07:46:46,End:2017-11-29 09:43:22
Coordinate: (36.066502567998,-94.1728048886)
Building: Mechanical Engineering
Major: Mechanical Engineering

2017-11-29 07:46:46 - 2017-11-29 09:43:22 Class Detected, Average Acceleration: 94.9722921951952

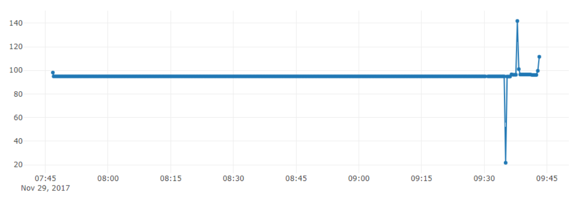


Figure 10. Mechanical Engineering

Start: 2017-11-29 10:51:53,End:2017-11-29 11:44:52
Coordinate: (36.066161313657,-94.173826866319)
Building: JB Hunt
Major: Computer Science/ Computer Engineering

2017-11-29 10:51:53 - 2017-11-29 11:44:52 Class Detected, Average Acceleration: 96.29429399999978

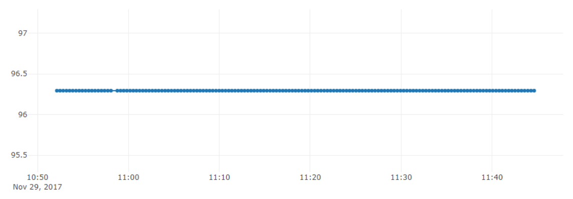


Figure 11. J.B. Hunt

Start: 2017-11-29 11:51:52,End:2017-11-29 12:48:20
Coordinate: (36.066190532769,-94.173994883572)
Building: JB Hunt
Major: Computer Science/ Computer Engineering

2017-11-29 11:51:52 - 2017-11-29 12:48:20 No Class Detected, Average: 97.73190600000042

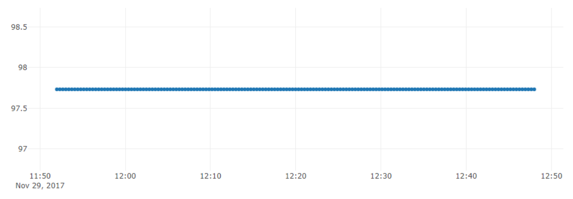


Figure 12.

Less than 50 Minutes Total (in hours)

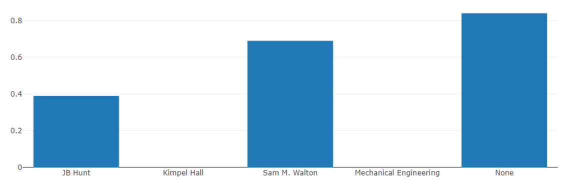


Figure 13.

8. Conclusion

I was able to correlate my own data with the correct department which then gave me computer engineering or computer science as the potential major. However, there are many problems, and cases that I can see in hindsight that would cause my approach to not provide the right inference or no inference at all. For example, buildings usually house more than one major, in the case of Bell engineering its most of engineering. Also, there could be classes that have nothing to do with the buildings major that take place in the building. The fluctuation in the coordinates also became a problem as sometimes they would show that somebody had moved when they really hadnt. Some potential future work that could make this approach better is to have a list of all the classes and their times that take place in a building this way you have a better idea of when someone might have a class, as opposed to relying on the accelerometer movements.

References

[1] Basbug, Mehmet Emin, et al. Accelerometer based Activity Classification with Variational Inference on Sticky HDP-SLDS. IEEE PERSON-CENTERED SIGNAL PROCESSING FOR ASSISTIVE, REHABILITATIVE, AND WEARABLE HEALTH TECHNOLOGIES, Sept. 2015, pp. 110.

[2] Zhi Xu , Kun Bai , Sencun Zhu, TapLogger: inferring user inputs on smartphone touchscreens using on-board motion sensors, Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks, April 16-18, 2012, Tucson, Arizona, USA

[3] X. Chen, A. Mizera and J. Pang, ”Activity tracking: A new attack on location privacy,” 2015 IEEE Conference on Communications and Network Security (CNS), Florence, 2015, pp. 22-30. doi: 10.1109/CNS.2015.7346806

[4] Veness, Chris. ”Calculate distance, bearing and more between Latitude/Longitude points.” Movable Type Scripts. Accessed November 20, 2017. https://www.movable-type.co.uk/scripts/latlong.html.

[5] ”Plotly.js The open source JavaScript graphing library that powers Plotly.” Plotly. Accessed November 30, 2017. https://plot.ly/javascript/.

[6] OpenWeatherMap. ”We Deliver 1 Billion Forecasts Per Day.” Accessed October 30, 2017. http://openweathermap.org/.