Bluetooth Tracking Analysis

11/29/2017
Bluetooth Tracking
http://computerforensicsblog.champlain.edu/

175 Lakeside Ave, Room 300A
Phone: (802) 865-5744
Fax: (802) 865-6446

Table of Contents

Introduction

Almost everyone nowadays has some sort of Bluetooth device. Whether it is their smartwatch, wireless earphones, security cameras, TVs, or even just their phones, more and more devices are becoming connected. While Bluetooth is a useful protocol it is not particularly secure, which makes using Bluetooth a potential security risk. The goal of this project is to determine if it is possible to track a Bluetooth device around a given area. This could be useful in a corporate situation to track devices around the building and spot unusual activity. For instance, if a new Bluetooth device appears in a corner of an office and never moves, it might be worth investigating to determine what the device is.

Background

The LCDI has performed multiple projects researching Bluetooth technology. Most recently a LCDI team investigated Bluetooth hacking. Our project is the first of its kind at the LCDI as it aimed specifically at tracking Bluetooth devices. As Bluetooth is a wireless protocol, Bluetooth signals have a property called Received Signal Strength Indicator (RSSI). By using a signal's RSSI and other estimated values, we believe a distance can be estimated. If we further calculate the distance of a device emitting Bluetooth signals from three separate points, the device's position can be plotted on a 2D plane. These variables can be obtained by using tools such as Ubertooth and Blue-Hydra. However, there are no tools available to parse information from these two tools or calculate a distance value from them. Our goal is to make such a program to aggregate the required information and turn it into positional data about the Bluetooth device.

Purpose and Scope

This project's purpose is to identify and track discoverable and undiscoverable Bluetooth devices throughout a building using tools such as Ubertooth, BlueHydra, and our own Python scripts. We will be using Bluetooth devices with known addresses as our controls, and ODROIDs will serve as our nodes to perform the tracking. Testing was performed in the lab to prevent recording third party Bluetooth Devices wherever possible, and our analysis has been limited to our control devices.

Research Questions

1. Is it possible to see a Bluetooth device that is undiscoverable?
2. Is it possible to track a Bluetooth device throughout a building?
   a. Is it possible to track undiscoverable Bluetooth devices throughout a building?
3. Is it possible to use trilateration to determine approximate location or distance to an unknown Bluetooth device?

Terminology

- **Bluetooth**- A low-power wireless connectivity technology used to stream audio, transfer data, and broadcast information between devices (Bluetooth.com, 2017).

- **Ubertooth -** Project Ubertooth is an open source wireless development platform suitable for Bluetooth experimentation. Ubertooth ships with a capable BLE (Bluetooth Smart) sniffer and can sniff some data from Basic Rate (BR) Bluetooth Classic connections (Greatscottgadgets, 2017).
- **BlueHydra -** A Bluetooth device discovery service built on top of the bluez library. BlueHydra makes use of Ubertooth where available and attempts to track both classic and low energy (LE) Bluetooth devices over time (Pwnieexress, 2017).
- **RSSI: Received Signal Strength Indicator** - Measurement of the power present in a received radio signal (Wikipedia, 2017)
- **Odroid -** Single board computers capable of running Linux (Hardkernel, 2017).
- **BlueZ -** A official Linux Bluetooth protocol stack. It is an Open Source project distributed under GNU General Public License (bluez.org)
- **Bluetooth Address -** 48-bit address commonly abbreviated as BD_ADDR. Seen as a 12 digit hexadecimal value (Jimb0).
- **NAP** - Non-significant Address Portion, assigned to manufacturer (mxs).
- **UAP -** Upper Address Portion of the MAC, 1 byte in size, links NAP and LAP for full address (mxs).
- **LAP -** Lower Address Portion, lower segment of the MAC address used for Bluetooth Packet addressing (mxs).
- **Trilateration -** Method of calculating the coordinates of an object using points of reference (Trilateration).
- **Determinate -** Useful Values determined from matrix multiplication; in our case the X and Y coordinates (Determinant).
- **Cramer's Rule -** Algorithm for solving a system of equations (Staple).

Methodology and Methods

Our first step was to conduct extensive research into the Bluetooth protocol, Ubertooth hardware, and various Linux tools. We discovered Kali Linux to be the ideal operating system, as it already had some of the tools we were looking for. Additional dependencies for the Ubertooth and methods for installing tools that take advantage of it were recorded. These research steps constituted our first few weeks of the project.
Our next phase was to gather our physical resources to build our project. We started out with a single laptop running Kali Linux and began to install the Bluetooth dependencies. We installed BlueHydra, Ubertooth Tools, and Kismet, and began testing our ability to pick up discoverable, undiscoverable, and low energy devices.

Once sufficient data was collected, we decided that using Ubertooth Tools and BlueHydra would be the most efficient. This is because Ubertooth Tools is able to collect RSSI values from both discoverable and undiscoverable devices. However, Ubertooth Tools fails at collecting low energy device information. BlueHydra, on the other hand, collects low energy device information, but fails at collecting RSSI values for undiscoverable devices. An example of a BlueHydra output can be seen in Figure 2: BlueHydra Output Database. To connect both tools, we wrote Python scripts to run each program at intervaled times, merging the information that each tool gathers so as to have a more complete picture of Bluetooth devices around us. Figure 1: Tech Jam Output , shows an example of a working earlier version of our code.

We then wrote our own Python scripts to parse out information from Ubertooth Tools. Once we had the laptop fully functional, we received 3 ODROID microcomputers to simulate a real usage scenario. The ODROIDs became our Primary and Secondary nodes for tracking Bluetooth devices in a room using our trilateration Python scripts. An example of our Python scripts connecting Primary and Secondary nodes can be found in Figure 4: Python Script Output.

Equipment Used

Table 1: Equipment Used

Device	OS Version/Manufacturer	Comments
Toshiba Satellite C55T-E511	Kali 2017.2	Used for Master Node & Analysis
Ubertooth One (3X)	2017-03-R2	Used to analyze Bluetooth Communication
ODROID C4	Ubuntu 16.04 Minimal	Secondary Node
ODROID C4	Ubuntu 16.04 Minimal	Secondary Node
ODROID XU4	Ubuntu 16.04 Mate	Primary Node
Micro SD Card	SanDisk	8 GB
Micro SD Card	Patriot	16 GB
Micro SD Card	Samsung EVO	32 GB
Bluetooth Adapter (3X)	Kinivo BTD-400	USB Adapters

Data Collection

Data collection was done via the Ubertooth Tools and through our own handwritten Python scripts. The Ubertooth Tools allowed us to gather information about discoverable and undiscoverable devices such as the Bluetooth addresses UAP, LAP, RSSI, and last seen time. This information was collected in real time, and parsed out by our various Python scripts. By using the RSSI, a sample TX Power variable, and average interference on each Bluetooth channel, we determined approximate distance from each node. These distances range from near, intermediate, and far. Our Primary determinant for distance is derived from the equation $D = 10^{((Po\text{-}Fm\text{-}Pr\text{-}10*N*\log10(f)+30*N\text{-}32.44)/(10+N))}$ simplified to $D = 10^{((txPower\text{-}RSSI)/20)}$ for local variables and testing. Then using Cramer's rule, the X and Y coordinates are calculated from the estimated distances and device coordinates.

Analysis

Scenarios that were used in this experiment involved three nodes, in our case comprised of the ODroid micro-computers, each equipped with Bluetooth adapters and Ubertooth One's. Each node would be placed around a building with Ubertooth, BlueHydra, and our Python scripts running. Each node was placed 30 feet (10 meters) away from the Primary Node, the maximum distance that Bluetooth adapters can operate on. The Primary Node sent out "Ready" commands to the two Secondary nodes and all three began running the Ubertooth commands, BlueHydra commands and Python scripts. The nodes each recorded any observed devices and sent their acquired information to the Primary Node. The Primary Node then calculated an approximate distance where each device was seen from each node.

The Python script that calculates distance and trilateration took the RSSI value and approximated a distance from it using our simplified equation. Using Cramer's rule, it then calculated the X and Y coordinates of the device relative to each node's locations. Our expectations were to get a rough distance measurement from each node to the Bluetooth devices in the area, and then based on trilateration, create a more defined location based on the calculations of all 3 nodes. The Target Device moved along known coordinates within the trilateration area while its approximate coordinates were calculated. The actual vs. approximated coordinates were taken and used to improve the algorithms for triangulation.

Once the program ran and completed, we intended to see all Bluetooth devices, discoverable or not, within range. We also aimed to be able to estimate their individual distances and approximate locations relative to each node when plotted on a coordinate plane. An example of results for a single node are found in Figure 5: Node Results.

Results

Feasibility of Tracking Bluetooth Devices

We were able to gain positive results from this investigation. The largest variation of actual distance from predicted distance was 5 feet. This means that our system can limit the position of the device to a five foot radius, making tracking devices feasible due to a reasonably low margin of error.

Area of Tracking

This testing also provided an idea of the ranges in which we could expect to detect both classic and low energy Bluetooth devices. Because BlueHydra required a traditional Bluetooth dongle to detect low energy devices, the detection range for these devices is limited to 10 meters (32.8 feet). For classic Bluetooth devices, we have a more theoretical range of 100 meters (328 feet). Although we did not test the full range of the Ubertooth, we did confirm that we were seeing longer distances from classic devices than we were from low energy devices.

Accuracy of Calculations

While the current accuracy is within 5ft of the actual range, this can be further reduced. To further optimize the calculation, three nodes would need to be implemented. The mode of the RSSI on each LAP over a short period of time, no more than a few seconds in duration, would be used to calculate a more accurate distance. The short monitor time will allow for timely updates on position in the case of movement.

Conclusion

Is it possible to see a Bluetooth device that is undiscoverable?

Contrary to popular belief, it is actually possible to pick up signals from devices already connected in a piconet. In order to do so, functions of Ubertooth Tools need to be utilized in conjunction with an Ubertooth device. Additionally, an undiscoverable device can only be seen if it is actively transmitting data from inside the piconet.

Is it possible to track a Bluetooth device throughout a building?

Yes, it is very much possible to track Bluetooth devices throughout a building. While it cannot find pinpoint locations, RSSI values can be calculated for each device seen and used as a reference for a change in distance. By using short, timed scans, the change in signal strength can be calculated in order to indicate a general movement.

Is it possible to track undiscoverable Bluetooth devices throughout a building?

Yes, but to track undiscoverable devices they need to be actively streaming data. When they are streaming data, Ubertooth Tools is able to pull the RSSI and our scripts calculate an approximate distance. In cases where the device was not transmitting information, the location could not be determined.

Is it possible to use trilateration to determine approximate location or distance to an unknown Bluetooth device?

Yes, while our time constraint prevented us from working through all the bugs in our code, a proof of concept was successfully demonstrated. Although this proof of concept does not provide the greatest level of accuracy we believe is possible, the 5 meter range is precise enough to conclude that trilateration can be used for Bluetooth tracking. By utilizing three distributed nodes configured to gather signal strength, a trilateration computation can be done to acquire a fairly accurate approximate device location as it moves throughout an area. A visualization of this concept can be seen in Figure 3: Ubertooth Diagram.

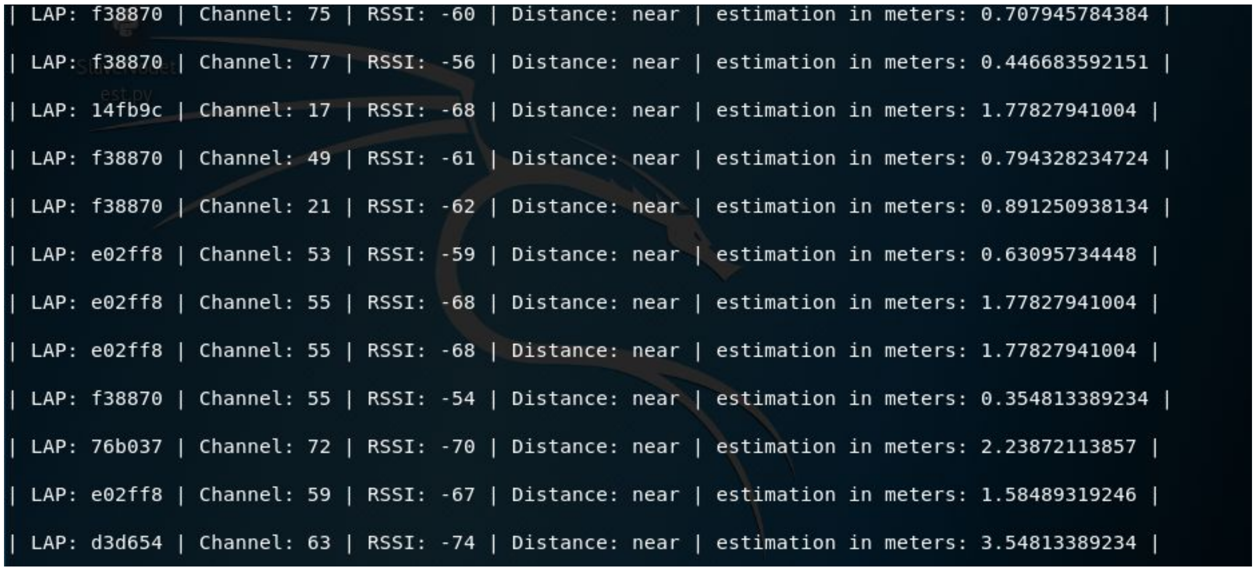
Further Work

Further work could include the sniffing of discoverable and undiscoverable Bluetooth packets, as well as performing man in the middle attacks on Bluetooth targets. Being able to sniff Bluetooth packets could result in the user gaining knowledge to what their target is using their Bluetooth connected devices for, and that they can find sensitive information in the process.

When analyzing broadcasts using the Ubertooth Tools, something of interest was discovered. It was noted that should a device attempt to connect to a previously paired but offline device, the first device would actively broadcast the LAP of the offline device. This would occur regardless of the secondary device's location. We believe this is due to Ubertooth picking up the paired LAP somewhere inside of the packet sent to search for it. It would be worth investigating if the contents of the Bluetooth packet provide more insight into the actual reason behind this. Understanding the contents of the Bluetooth packet could provide insight into successfully identifying these broadcasts as false positives.

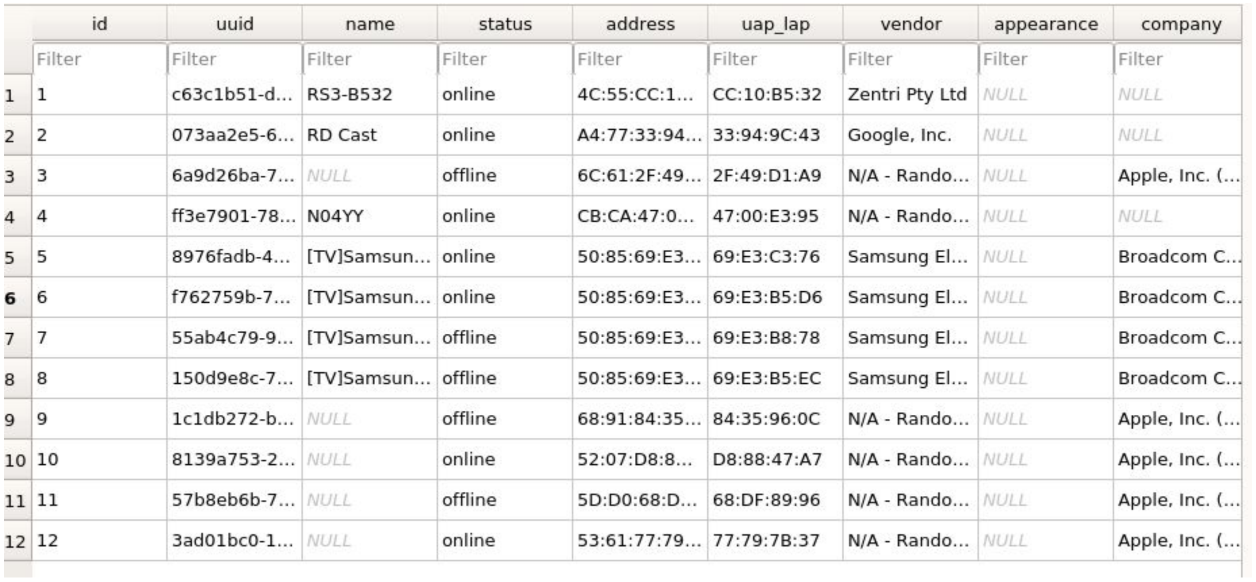
With more testing and time, algorithms for guessing a device's TX-power could be developed. As the max range is known and analysis can be done from several points, an algorithm could be developed that factors in interference between devices, max distance, and known distances to create an estimation TX-power, making the coordinates more accurate.

Appendix



```
| LAP: f38870 | Channel: 75 | RSSI: -60 | Distance: near | estimation in meters: 0.707945784384 |

| LAP: f38870 | Channel: 77 | RSSI: -56 | Distance: near | estimation in meters: 0.446683592151 |

| LAP: 14fb9c | Channel: 17 | RSSI: -68 | Distance: near | estimation in meters: 1.77827941004 |

| LAP: f38870 | Channel: 49 | RSSI: -61 | Distance: near | estimation in meters: 0.794328234724 |

| LAP: f38870 | Channel: 21 | RSSI: -62 | Distance: near | estimation in meters: 0.891250938134 |

| LAP: e02ff8 | Channel: 53 | RSSI: -59 | Distance: near | estimation in meters: 0.63095734448 |

| LAP: e02ff8 | Channel: 55 | RSSI: -68 | Distance: near | estimation in meters: 1.77827941004 |

| LAP: e02ff8 | Channel: 55 | RSSI: -68 | Distance: near | estimation in meters: 1.77827941004 |

| LAP: f38870 | Channel: 55 | RSSI: -54 | Distance: near | estimation in meters: 0.354813389234 |

| LAP: 76b037 | Channel: 72 | RSSI: -70 | Distance: near | estimation in meters: 2.23872113857 |

| LAP: e02ff8 | Channel: 59 | RSSI: -67 | Distance: near | estimation in meters: 1.58489319246 |

| LAP: d3d654 | Channel: 63 | RSSI: -74 | Distance: near | estimation in meters: 3.54813389234 |
```

Figure 1: Tech Jam Output



	id	uuid	name	status	address	uap_lap	vendor	appearance	company
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	c63c1b51-d...	RS3-B532	online	4C:55:CC:1...	CC:10:B5:32	Zentri Pty Ltd	NULL	NULL
2	2	073aa2e5-6...	RD Cast	online	A4:77:33:94...	33:94:9C:43	Google, Inc.	NULL	NULL
3	3	6a9d26ba-7...	NULL	offline	6C:61:2F:49...	2F:49:D1:A9	N/A - Rando...	NULL	Apple, Inc. (...
4	4	ff3e7901-78...	N04YY	online	CB:CA:47:0...	47:00:E3:95	N/A - Rando...	NULL	NULL
5	5	8976fadb-4...	[TV]Samsun...	online	50:85:69:E3...	69:E3:C3:76	Samsung El...	NULL	Broadcom C...
6	6	f762759b-7...	[TV]Samsun...	online	50:85:69:E3...	69:E3:B5:D6	Samsung El...	NULL	Broadcom C...
7	7	55ab4c79-9...	[TV]Samsun...	offline	50:85:69:E3...	69:E3:B8:78	Samsung El...	NULL	Broadcom C...
8	8	150d9e8c-7...	[TV]Samsun...	offline	50:85:69:E3...	69:E3:B5:EC	Samsung El...	NULL	Broadcom C...
9	9	1c1db272-b...	NULL	offline	68:91:84:35...	84:35:96:0C	N/A - Rando...	NULL	Apple, Inc. (...
10	10	8139a753-2...	NULL	online	52:07:D8:8...	D8:88:47:A7	N/A - Rando...	NULL	Apple, Inc. (...
11	11	57b8eb6b-7...	NULL	offline	5D:D0:68:D...	68:DF:89:96	N/A - Rando...	NULL	Apple, Inc. (...
12	12	3ad01bc0-1...	NULL	online	53:61:77:79...	77:79:7B:37	N/A - Rando...	NULL	Apple, Inc. (...

Figure 2: BlueHydra Output Database

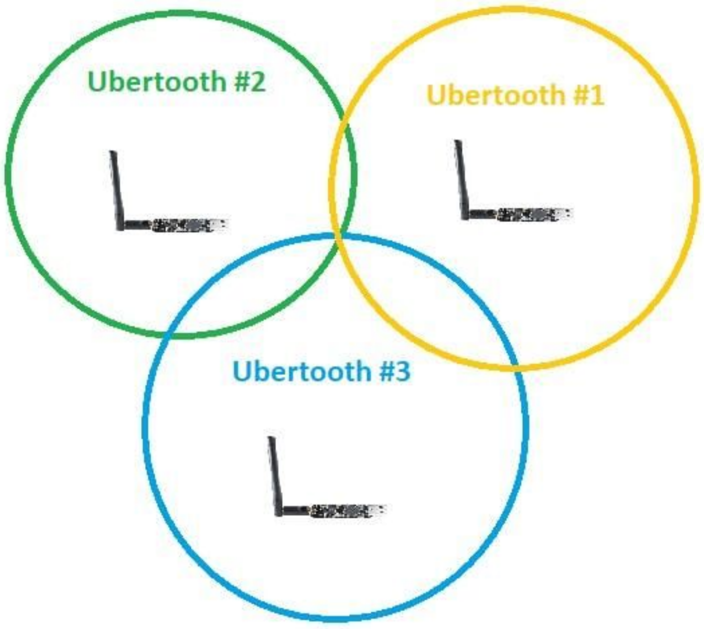
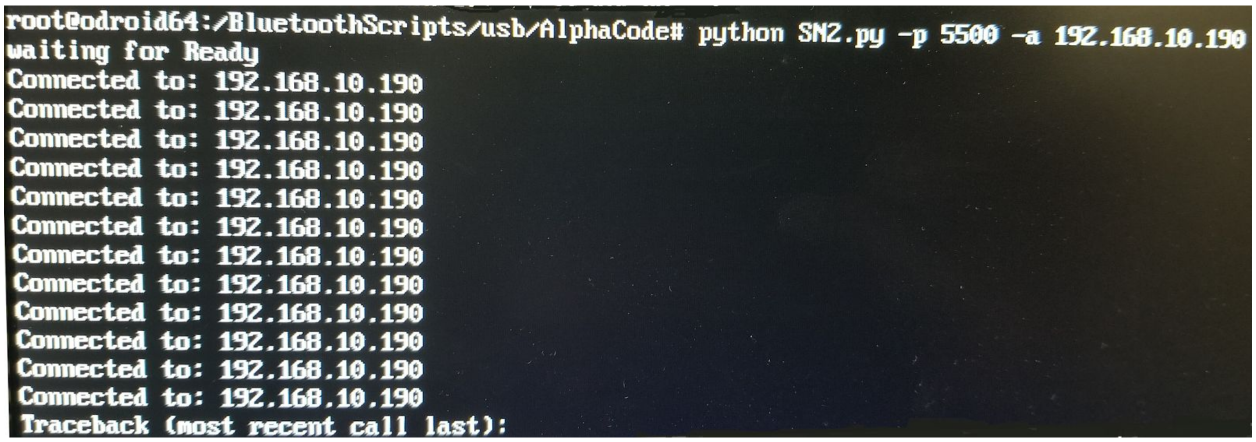
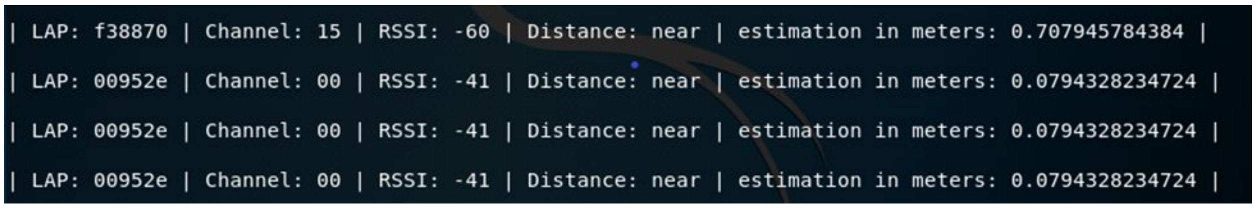


Figure 3: Ubertooth Diagram



```
root@odroid64:/BluetoothScripts/usb/AlphaCode# python SN2.py -p 5500 -a 192.168.10.190
waiting for Ready
Connected to: 192.168.10.190
Connected to: 192.168.10.190
Connected to: 192.168.10.190
Connected to: 192.168.10.190
Connected to: 192.168.10.190
Connected to: 192.168.10.190
Connected to: 192.168.10.190
Connected to: 192.168.10.190
Connected to: 192.168.10.190
Connected to: 192.168.10.190
Connected to: 192.168.10.190
Connected to: 192.168.10.190
Traceback (most recent call last):
```

Figure 4: Python Script Output



```
| LAP: f38870 | Channel: 15 | RSSI: -60 | Distance: near | estimation in meters: 0.707945784384 |

| LAP: 00952e | Channel: 00 | RSSI: -41 | Distance: near | estimation in meters: 0.0794328234724 |

| LAP: 00952e | Channel: 00 | RSSI: -41 | Distance: near | estimation in meters: 0.0794328234724 |

| LAP: 00952e | Channel: 00 | RSSI: -41 | Distance: near | estimation in meters: 0.0794328234724 |
```

Figure 5: Node Results

References

BlueZ. (n.d.). Retrieved October 30, 2017, from http://www.bluez.org/faq/common/

"Determinant of a Matrix." *Determinant of a Matrix*, www.mathsisfun.com/algebra/matrix-determinant.html. October 30,

Encyclopedia. (n.d.). Retrieved October 30, 2017, from https://www.pcmag.com/encyclopedia/term/58396/2-4-ghz-band

G. (2017, June 24). Greatscottgadgets/ubertooth. Retrieved October 23, 2017, from https://github.com/greatscottgadgets/ubertooth

How it works | Bluetooth Technology Website. (n.d.). Retrieved October 23, 2017, from https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works

Jimb0. "Bluetooth Basics." *Bluetooth Basics*, learn.sparkfun.com/tutorials/bluetooth-basics/how-bluetooth-works.October 23, 2017

mxs. "Bluetooth: Defining NAP + UAP + LAP." *Nccgroup.trust*, 13 Feb. 2012, www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2012/february/bluetooth-defining-nap-uap-lap/.October 23, 2017

Our Most Advanced Penetration Testing Distribution, Ever. (n.d.). Retrieved October 30, 2017, from http://www.kali.org/

P. (2017, October 12). Pwnieexpress/blue_hydra. Retrieved October 23, 2017, from https://github.com/pwnieexpress/blue_hydra

Stapel, Elizabeth. "Cramer's Rule." *Cramer's Rule*, www.purplemath.com/modules/cramers.htm

"Trilateration." *Dictionary.com*, Dictionary.com, www.dictionary.com/browse/trilateration.

Welcome to Python.org. (n.d.). Retrieved October 23, 2017, from https://www.python.org/

WiFi Lessons. (n.d.). Retrieved October 30, 2017, from https://www.metageek.com/training/resources/understanding-rssi.html