Spectrum Vulnerability Meter

Priyanka Samanta, Ari Mermelstein

December 24, 2017

1 Introduction

In order to cater the growing spectrum demands of large scale future 5G Internet of Things (IoT) applications, Dynamic Spectrum Access (DSA) based networks are being proposed as a high-throughput and cost-effective solution. However, the lack of understanding of the DSA paradigms inherent security vulnerabilities on IoT networks might become a roadblock towards realizing such a spectrum aware 5G vision. One such inherent DSA vulnerability is Spectrum Sensing Data Falsification (SSDF) attacks. These attacks can be carried out by collaborative groups of selfish adversaries. Researchers have shown how SSDF attacks can impact the performance of spectrum aware IoT applications.

In this project, we have created a spectrum usage visualization for 4 different spectrum sub-bands, which enables us to we to observe and compare the vulnerabilities of channels across the four sub-bands. We define "vulnerability" as the absolute difference of observed power spectral density (in dB) of a particular channel from primary presence threshold, which is our case was experimentally found to be -90dB. That is, $V_t = |D_t - T|$, where V_t is the vulnerability of a channel at a particular time stamp, D_t is the power spectral density of a channel at a particular time stamp (in dB), and T is the primary user threshold. "Vulnerability" is a measure of how likely a particular channel is open to an SSDF attack.

2 Metadata

The data that we used for this experiment was collected by RWTH Aachen University's Department of Wireless Networks in Aachen, Germany. The particular data set that we used was collected in Maastricht, Netherlands, and was called the NE data set. This data was collected over a week-long period, and comprises approximately 250GB worth of .mat (matlab) files. Each .mat file consists of 1000 rows and 8192 columns, where each column represents a channel and each row represents a 1.8 second time interval. Each channel has an approximate bandwidth of 200kHz. Each .mat file represents 30 minutes of power spectral density data.

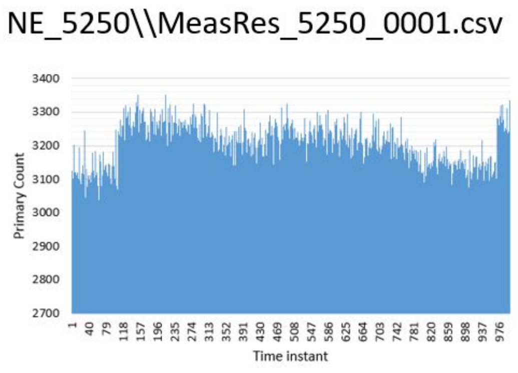

Figure 1: Time Vs. Primary Count

3 Prior Work

3.1 Conversion of Data into .csv Format

Since the data came to us in .mat format, and virtually all of the required Python libraries digest data in .csv format, we first had to write a Python script to convert a .mat file into a .csv file. We then wrapped that script in a Windows command line script to convert all of our files.

3.2 Preliminary Experiments

3.2.1 Reading in a .csv File into Python

Once our data was converted properly to .csv format, we wrote a Python script to load each .csv file into a Python dictionary. We were able to accomplish this easily using the pandas library and the read_csv() function.

3.2.2 Detecting the Primary Users

Once we figured out how to read in a file, we were able to look at each time stamp in the file and count the number of times that the primary user was present. The primary user is said to be present in the dB value measured is at least -90dB. We then created two Python lists: one for the time stamps and one for the counts. We then passed these lists to Matplotlib. Unfortunately, this type of visualization gave us no usable information, as the bars were very packed together. This gave us little to no information about the vulnerabilities. (See Fig 1)

3.2.3 Detecting Vulnerable Channels

After reading a file into Python, We eliminated those channels that were not vulnerable enough according to the above definition. We then decomposed the dictionary into two Python lists and passed those lists to Matplotlib. We then used a Matplotlib bar graph to attempt to visualize the vulnerable channels and their respective power spectral densities. These plots tended to be very sparse for individual 30 minute windows. (See Fig 2)

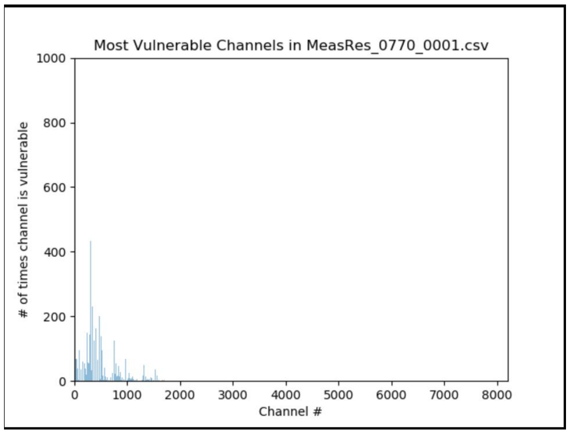

Figure 2: Number of Channel Vs. Number of time a channel get vulnerable

3.3 Coalescing .csv files into Periods

3.3.1 Creating One Day Long Files

We realized that for our purposes, analyzing one file at a time was not going to yield any large scale results. We then, therefore, wrote another Python script to open up a .csv using pandas and then appending that file to a master .csv file capable of storing the information for an entire day at a time. These files still wound up being infeasible, since they were approximately 18 GB in size.

3.3.2 Creating 6 Hour Files

Realizing that this too was unfeasible, we further borke down each file into quarter day length. These files became feasible to work with, but they took a very long time to run once passed to Plotly for visualization. We needed to figure out a way of compressing the data files so that we could efficiently run plots in real time.

4 Methods and Implementation

4.1 Compressing .csv Files Using NumPy

We learned how to compress .csv files using NumPy and .npz files. We first loaded an entire directory of .csv files using the glob Python module. We were then able to concatenate these files together and call NumPy's savez_compressed() function to compress the data into .npz files. These files, each containing one day's worth of data, only took up 1 GB on disk (rather than 18 GB before compression.)

4.2 Using NumPy to Initialize Data Structures

Now that we had our data set compressed into .npz files, we loaded then into NumPy Arrays. NumPy arrays, for our purposes are arranged as matrices, where each matrix data[0..N-1, 0..M-1] represents one day's worth of spectral power density data, N is the number of timestamps across

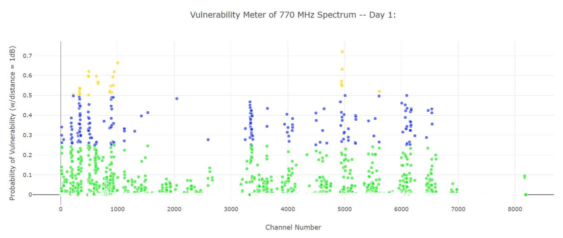

Figure 3: Probabilities of Vulnerability of channels across a particular Day with Respect to a Particular Sub-band

one day (approximately 48,000) and M is the number of channels (8192). We then created another matrix distance[0..N-1, 0..M-1] in which

$$distance[i, j] = |data[i, j] - (-90)|$$

We then calculate the probabilities of vulnerability with respect to a fixed distance threshold. NumPy transforms each column of the data matrix into a 1D array whose entries are either 1 or 0 depending on whether the distance is $\leq$ the threshold. The NumPy mean() function then calculates the probabilities for each channel by averaging each column of the transformed matrix, and returns these probabilities in a (1D) array of length M.

4.3 Using Plotly to Generate Dynamic Probability Scatter Plots

We then created scatter plots to visualize each channel's vulnerability over one day with respect to a particular sub-band. This was done as follows: once we have the array of probabilities of vulnerability handed back to us from NumPy, we mapped this array to an array of colors and sizes. We decided the bubbles corresponding to the channels with probability ≥ 0.9 should be colored in red, and their sizes should be bigger than the other channels. We discretized the colors based on a sliding scale of probabilities. We then used Plotly's scatter plot utility to plot channel numbers (x-axis) vs. probability of vulnerability. See Fig 3.

4.4 Using Plotly, NumPy, and glob to Create Heat maps

Another important utility that we have created is a heat map that allows us to visualize the probability distribution of a channel and sub-band over an entire week. To do this, we read in all the files for the week using glob, concatenate them together, and pass this to our drawHeatmap() function. Our function isolates data[0..7N-1, j] and the calculates the associate probabilities with low, medium, and high thresholds. This calculation is done as described in §4.2. We pass these values to Plotly's Heatmap() function to plot the probabilities of vulnerability over each threshold and day of the week. See Fig 4.

5 Dashboard

Our dashboard is an HTML file that contains 4 parts: Individual View, Comparison View, Metadata View, and the Heat Map Generator Tool.

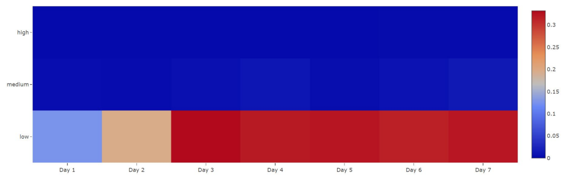

Figure 4: Heatmap Representing the Probability Distribution for a particular Channel with Respect to a Particular Sub-band for an Entire Week

5.1 Individual View

The Individual View allows the user to select a particular sub-band, day of the week, and whether they would like to see the high, medium, or low vulnerability thresholds, and displays the corresponding Plotly scatter plot. These plots allow for dynamic resizing and zooming, so that the user can easily isolate any particular set of channels that are of interest.

5.2 Comparison View

The Comparison View allows the user to pick a particular day of the week, and vulnerability level and displays the corresponding scatter plots for all 4 sub-bands at the same time.

5.3 Metadata

We have provided the metadata PDF for easy reference so that the user can look up the details on how the data was gathered, where it was gathered, when the data was gathered, and the measurement technique used to acquire the data.

5.4 Heat Map Generator Tool

Our HTML file provides radio buttons and a text box so that the user can enter a particular channel number and sub-band, and a heat map is generated in real time using the procedure described in §4.4. The HTML file and Python script described in §4.4 were hosted on an XAMPP server. The HTML and Python files communicate using the post method and CGI.

6 Conclusion

This project is about the visualization of 7 days used spectrum data of Aachen Germany. Our objective functions behind this visualization was:

1. Is there any specific day when channels are most vulnerable?
2. Is there any specific sub-band where channels are consistently vulnerable ?
3. What is the probability of vulnerability for a particular channel under a particular sub-band given varying thresholds for vulnerability?

4. What is the probability distribution of a particular channel of interest over the course of a week?

From the Individual View, we can see the probability of vulnerability for all channels under a specific sub-band for all 7 days. So we can easily visualize if there is any specific day when the channels are getting more vulnerable under a specific sub-band and we also can visualize if there is any specific sub-band where channels are consistently vulnerable.

From the comparison view as we are able to visualize all 4 sub-bands together, so we can easily find the probability of vulnerability for a particular channel with varying threshold under all sub-bands at the same time.

From the Heat Map Generator Tool, the user is able to choose a particular channel under a particular sub-band of interest and can see the probability distribution for that channel over a week, where it is easy to predict that how much that channel can contribute in SSDF attack in future.

Behind the creation of this visualization, our motivation was if we can predict the vulnerability nature for different sub-bands so that it will be easy to choose a particular sub-band for dynamic spectrum access in future 5G architecture for any government. Though this visualization only based on a particular location data, but we strongly believe this spectrum usage data visualization approach can be executed for future research of DSA and SSDF attack for more predictive aspect.

7 References

1. NumPy: http://www.numpy.org/
2. pandas: https://pandas.pydata.org/
3. Plotly: https://plot.ly/python/
4. Matplotlib: https://matplotlib.org/
5. Metadata: https://download.mobnets.rwthaachen.de/fileadmin/Documentation/Metadata.pdf