Classification of activities on TPLink router

Abhinav Narain

January 29, 2017

1 Study of EMI activity on TPLink router

This study is for EMI activity studied for router shown in figure 7.

2 Benchmarks

We want micro benchmarks to suggest how good or bad the technique we are using. These benchmarks will vary depending on the device if we want to do activity recognition (for example,camera might run different applications than a photo-frame.) but might remain the same for basic activities done by malware.

As suggested by Kyle, I have simulated malware activities, instead of running actual malware.

I have done classification for the following activities with the technique and the results in the section 3 and section 4

- DNS packet flood attack
- UDP flood attack with payload of 1460 bytes
- UDP flood attack with payload of 730 bytes
- Router with default firmware running (Idle)
- Computation in a loop

There are set of benchmarks I wanted to do but was unable to proceed and reasons that it was not possible.

- Experiments with measurements of L1, L2, L3 caches
  - Router does not have many levels of caches as dmesg shows 5.3
- copying a file (using Linux command *dd*)
  - Does not have enough memory to copy files for recording it

Figure 1 shows the change in EMI due to turning hardware switch on or off for the router 7. This is significant change due turning on/off the wireless subsystem which wasn't present in case of laptop. There is clearly a wide difference in the proportion of power consumption among the subsystems depending on the hardware of the device.

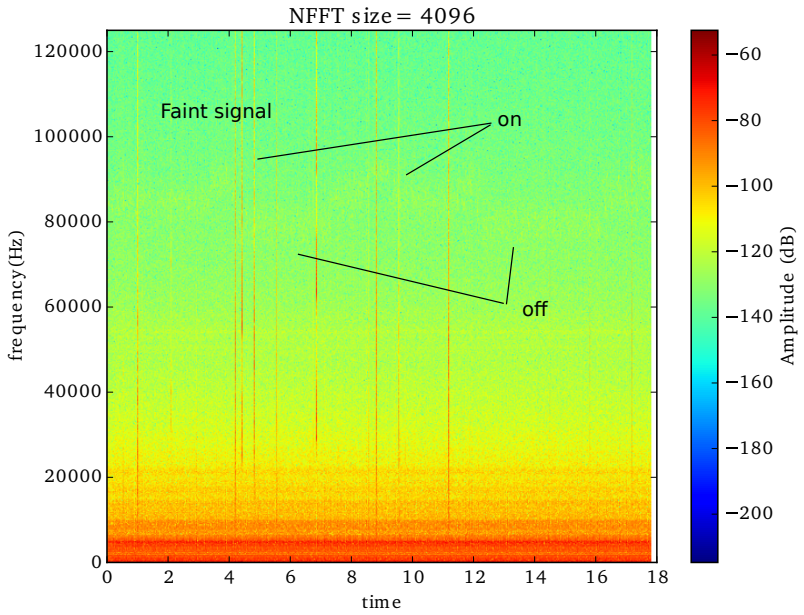

Figure 1: EMI trace of power-line when the switch powers the wireless subsystem on and off. Fs=500kHz

3 Technique

Although there is large spectrum of noise that is produced by the router, but using the whole spectrum might not be practical (although techincally correct in the isolated setting) to use in real setting. The WattsUpDoc(Kevin Fu) work can uses the whole frequency range as feature set because the setup isolates the device measurements from the channel measurement with the extra hardware between the socket and the device.

The EMI trace is segmented period of 10ms as shown in Fig 3 which is a basic unit of the EMI trace of the device 2 and decomposed using DWT

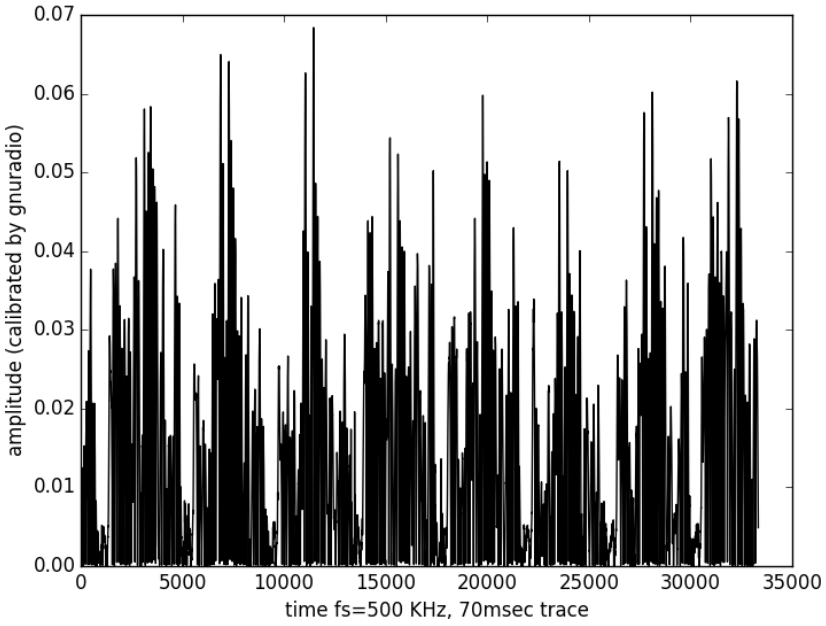

Figure 2: Signal trace of router EMI in time domain. This corresponds to the alternating bands (cyclo-stationary) in red parts of the spectrogram

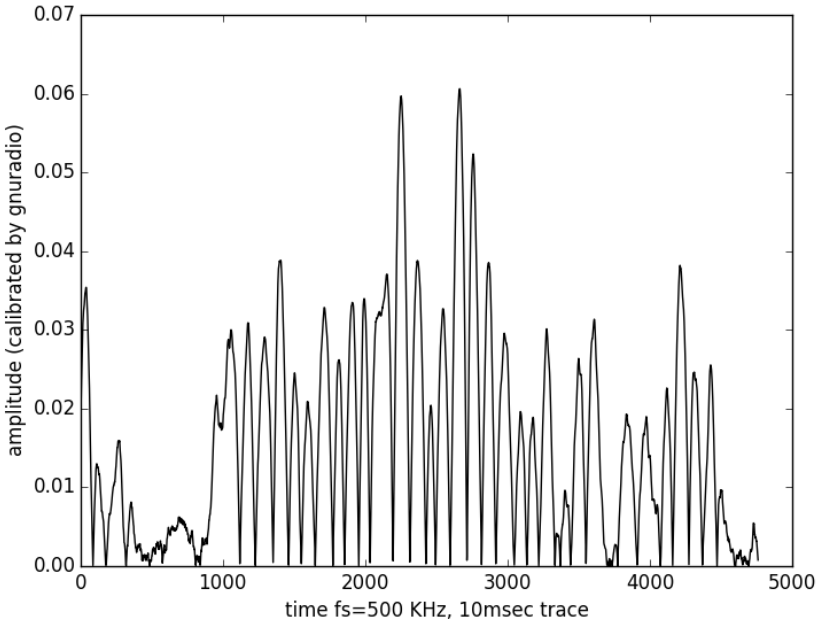

Figure 3: Signal trace of a single segment of router EMI, corresponding to one power cycle of the router. This segmented trace is given as a feature vector to the classifier

Each of the the segments in 3 are used for creating a feature for the supervised classifier.

To use the coefficients corresponding to the signal referring only to the specific activity, we have used wavelets as band pass filter to filter frequency using Discrete Wavelet Transform.

The difference between Discret Wavelet Transform (DWT) and Wavelet Packet Decomposition (WPT), WPT decomposes the signal space into a full binary tree, and the basis used are not orthogonal (that is some nodes will have redundant information). DWT only decomposes the left half with orthogonal basis tdhe left branch recursively.

While decomposition at every level, each of the algorithms (DWT and WPT) subsamples while passing through the Low Pass Filter and High Pass Filter, succesively reducing the number of coefficients by half and also partitioning the frequency content as shown in figure 4.

Sample rate used in experiments is 500 KHz and the interested frequecies (by visual inspection of spectrogram) are in 60.25KHz-125 KHz range, and hence we have selected the DWT coefficients of the second node of the second row.

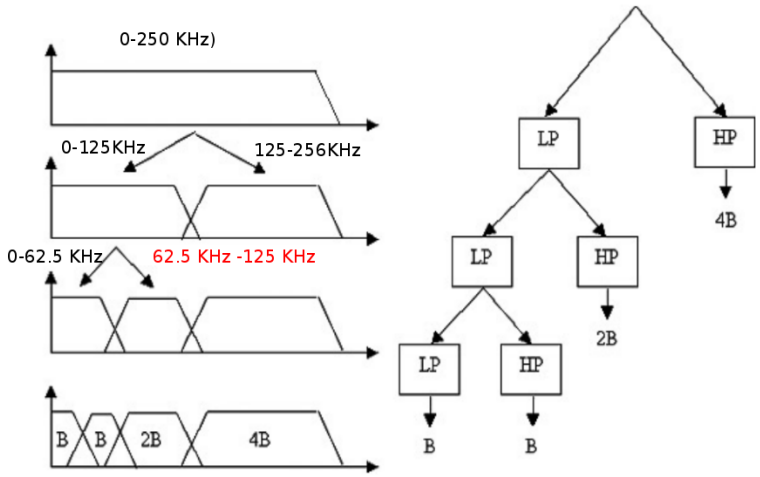

Figure 15. Decomposition of a signal through application of the DWT

Figure 4: Discrete Wavelet Transform. Decomposing the frequency range into ranges of values by half at every subsequent level of the tree. The sampling frequency is 500KHz and we decompose the maximum frequency (250 KHz) at every level.

4 Results

Spectrograms in Fig 8 and Fig 9 show the difference in the change in frequency when DNS flood attack is performed. The en

Using wider range of frequencies give much better results as indicated by the precision-recall curve below. Differentiation in packet sizes might not just be attributed to the change in energy consumption by the wireless card but also the difference in code generating it. Although this effect is mitigated in the experiments when the same code generated UDP packets of different sizes.

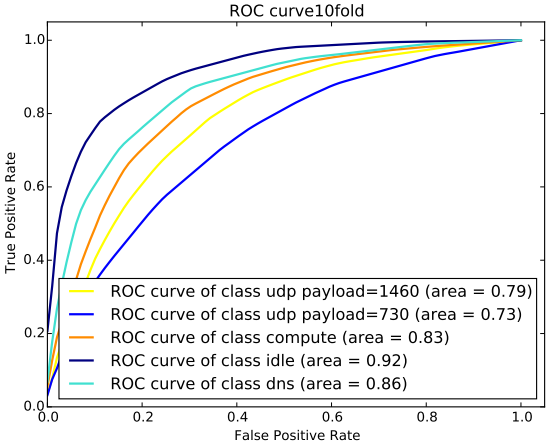

Figure 5: Random Forest Multiclass classifier. This is done using Wavelet coefficients at level 2 (on the level marked red in Fig 4.

If we take coefficients between between the frequency range of 60-120 KHz as suggested by the spectrogram, we get poor precision-recall curves, unfortunately there is a lot of information in lower frequencies which needs to be used for better results.

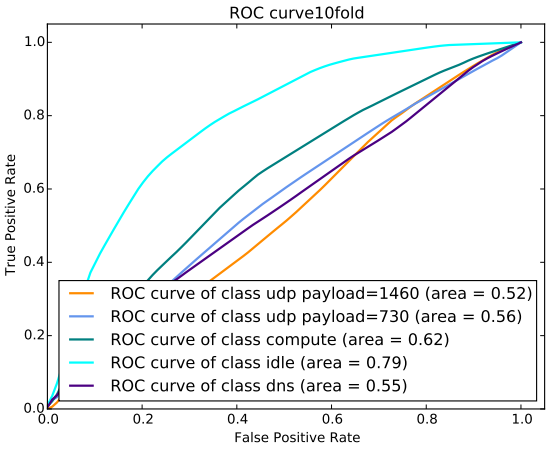

Figure 6: Random Forest Multiclass classifier. This is done on the data after running a bandpass filter for 60-120KHz to extract the frequencies which were visible in spectrogram. The wavelet coefficients corresponding to the node marked red in Fig 4.

4.1 Comments

1. The results are dependent on the hardware. Raspberry Pi did not show any activity using our hardware, while the Netgear Router shows significant EMI variation for wireless component turned on and the packet sizes

2. EMI is not very effective in detecting stealthy malware, which has been established in last report on mirai, except when it is actively in attack mode. A malware can *sleep* for random time period (for example in *mirai*) which would evade signature detection. I don't think malware would be written to especially fool such a detector, but there would be many true negatives if the current malware do so already

3. This technique is highly dependent on the hardware configuration. Unlike Shwetak's work, where television definitively showed correlation between the EMI and movies, we might have to test many IoT/computing devices to show on which ones this works

4. I also think, the rate of change of EMI is very slow as the television screen and most components are not as fast switching as computing devices

5. One might use devices which already use *OpenWrt* to for experiments as one can cross-compile and run custom workloads on them

6. We can study privacy invasion by measuring EMI when one watches a movie. There has been study of packets bitrate coding leaking the information about the movie watched on a slingbox (*Usenix Sec 2007: Devices That Tell On You: Privacy Trends in Consumer Ubiquitous Computing*). This is similar to experiments where Fu et al. have used their system to show the web pages visited on a laptop (although we do it on a router)

7. We don't know the actual power (in terms of dBm, which is something anyone will be interested in knowing), which we don't have any calibration device to measure

8. I had also pursued finding if we could use some bulbs (suggested by Kyle in our last meeting) etc. to do a distributed attack from bunch of IoT devices together but I was not able to find anything useful

5 Supplemtary

Figure 7: router used for the study

5.1 Spectrograms of the traces

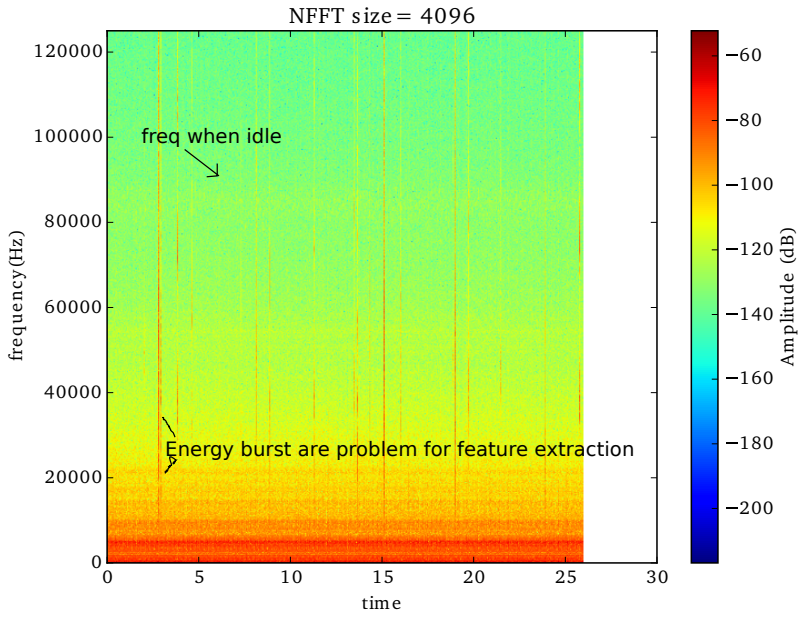

Figure 8: EMI trace when the router is idle, which means, it is not executing any special application/binary installed on it. EMI is at 80KHz. This might be a harmonic for a lower frequency, but it is most clear to us and hence indicated in the spectrogram.

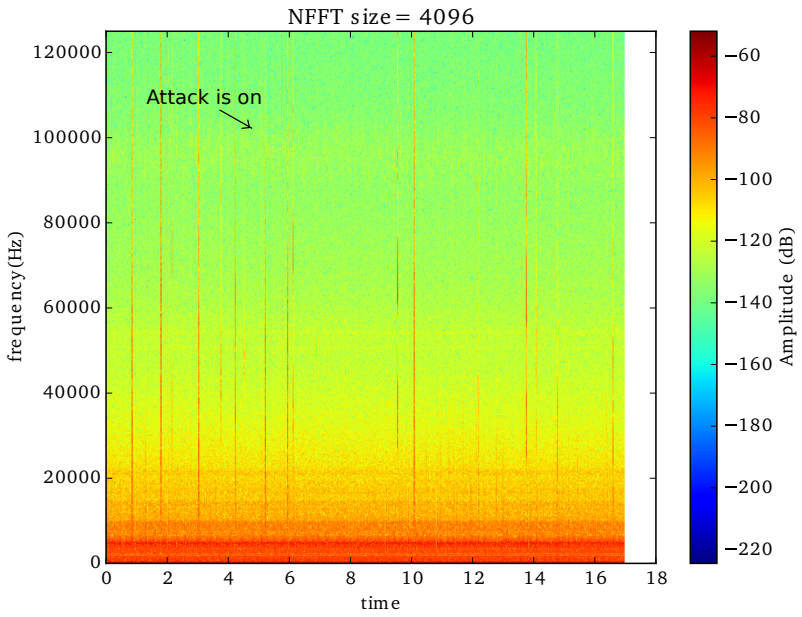

Figure 9: EMI trace when the router is doing DNS flood attack. The frequency changes to 100KHz. The same behavior happens when we do UDP flood attack.

5.2 Time traces of the samples

5.3 dmesg output

dmesg output suggesting only L1 cache on router. *lscpu* command was not found even after going through whole of menconfig/defconfig files while building *OpenWrt*.

```
Cached:             6284 kB
[    0.000000] Primary instruction cache 64kB, VIPT, 4-way, linesize 32 bytes
[    0.000000] Primary data cache 32kB, 4-way, VIPT, cache aliases, linesize 32 bytes
[    0.000000] Dentry cache hash table entries: 16384 (order: 4, 65536 bytes)
[    0.000000] Inode-cache hash table entries: 8192 (order: 3, 32768 bytes)
[    0.086171] Mount-cache hash table entries: 1024 (order: 0, 4096 bytes)
[    0.093221] Mountpoint-cache hash table entries: 1024 (order: 0, 4096 bytes)
```