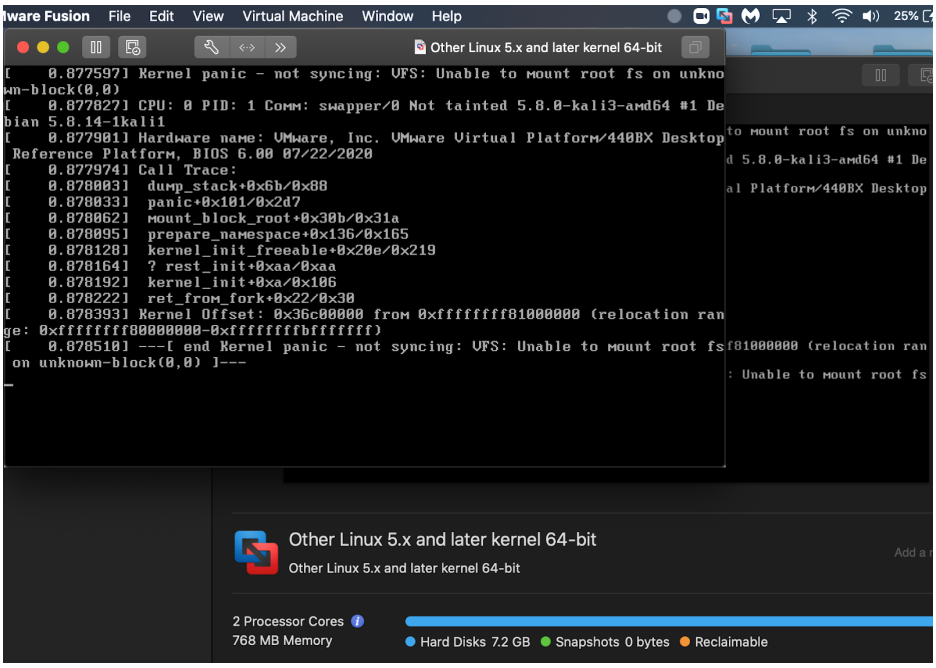
Nicolas Barraclough
11/1/2020
CS373 – Defense Against the Dark Arts

Lab 3: Software Defined Radio

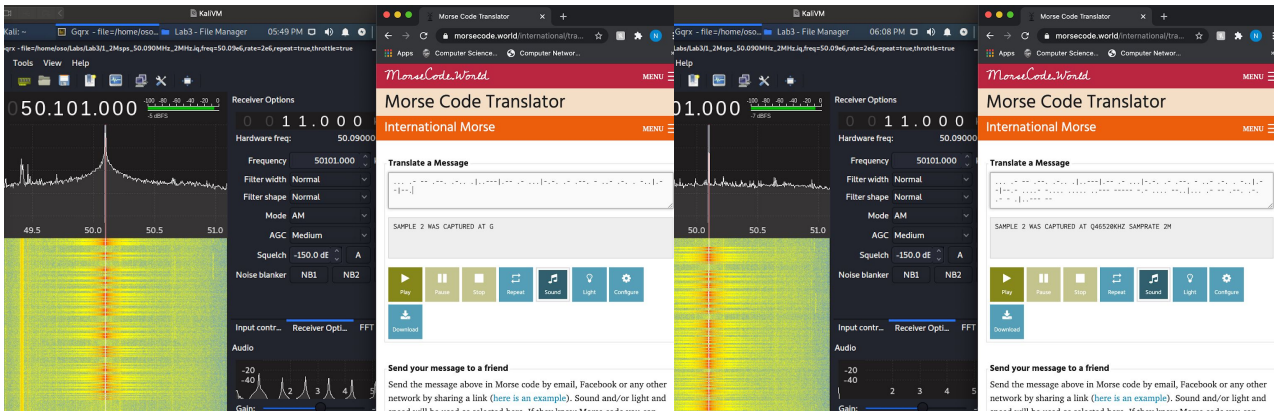
This lab was rough for me to set up. I had to use a different computer than the one I used for the first lab, so I had to set up the VM from square 1. I ran into quite a few issues while trying to get the VM situated. When I finally got the Kali VM to boot up, there were quite a few issues related to the script that set up the VM for the class material. After running the script for the first time, the VM had a kernel panic. Luckily, I took a snapshot before running the script and I was able to try it again pretty quickly. I think the issue was due to my not paying attention to the install prompts and the VM suspended itself mid-installation.



After updating apt-get and manually retrieving the lab files from dmcgrath's setup repo (because the wget command in the provided script hung no matter how long I waited), I was finally able to start the lab, days after I intended to. For some reason, the third Lab3 file won't extract, but I will have to solve that issue when I am ready to work on the third file.

Having no clue where to start in this lab even after the 2-hour long SDR video, I took to the assignment hints for guidance. I installed Gqrx and opened the first lab file with it. I had quite a bit of trouble at first getting the file to open and to hear anything other than my own audio input or jumbled noise. Scouring Slack, I finally got recognizable audio to come from Gqrx. It was morse code (as Slack hinted). Decoding the message was difficult because the heat map that

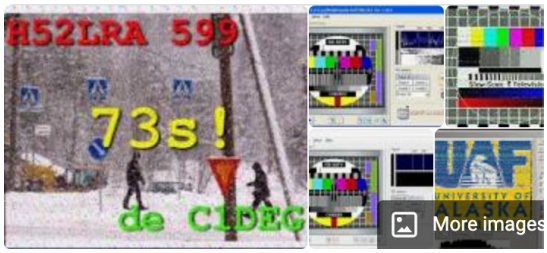
the audio made did not make the distinct dots or lines apparent. I have 0 experience with morse code so I couldn't understand even the respective letters/words in the message. I found a morse code translator online and did my best to map the heat map to the dots and lines.



The translation I ended up with was "`SAMPLE 2 WAS CAPTURED AT Q46520KHZ SAMPRATE 2M`" which seems to make sense aside from the 'Q' in the KHz number. I had a pretty hard time finding a number from that heat signature segment, so instead of finding what that number is and since it is just one number that I need to replace, I will try out various frequencies until I find the correct one for sample 2. Sample 1 complete!

Using the message from sample 1 and through trial and error with the first number of the frequency, I was able to figure out that the 'Q' in my decoded message was supposed to be a '1'. Using 146.52MHz frequency with a rate of 2, I successfully received audio from the second sample. It was a man's voice reading off, "The third file was captured at 7.171MHz at (10 or 2).5Msps." Pretty self-explanatory and a fairly easy sample to work through! I wasn't completely sure if he said 10.5Msps or 2.5Msps, but 10.5 seems kind of high so I will try 2.5 first in sample 3. Sample 2 done.

I was finally able to get the 3rd sample by retrying the "wget" command. I threw it into Gqrx and got an output of a jumbled mess of noise. Looking on Slack for hints, I see that googling the frequency will provide a clue as to what tool I can use for this sample. 7.171MHz seems to be an Slow-scan television (SSTV) frequency. SSTV's Wikipedia page says that this is a picture transmission method. Perhaps that jumbled noise I was hearing in Gqrx was actually a picture!

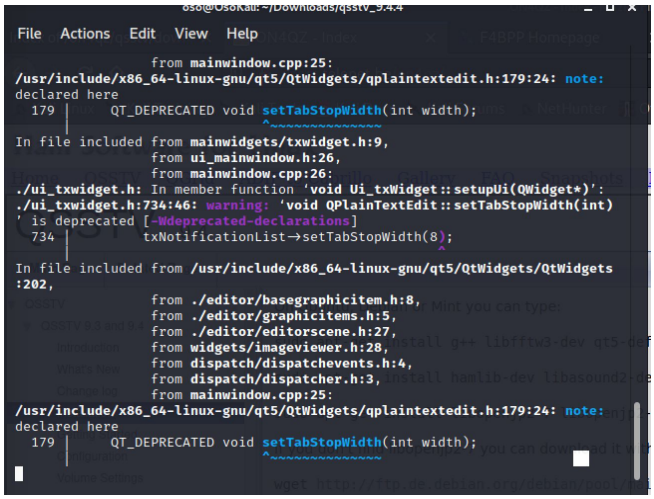


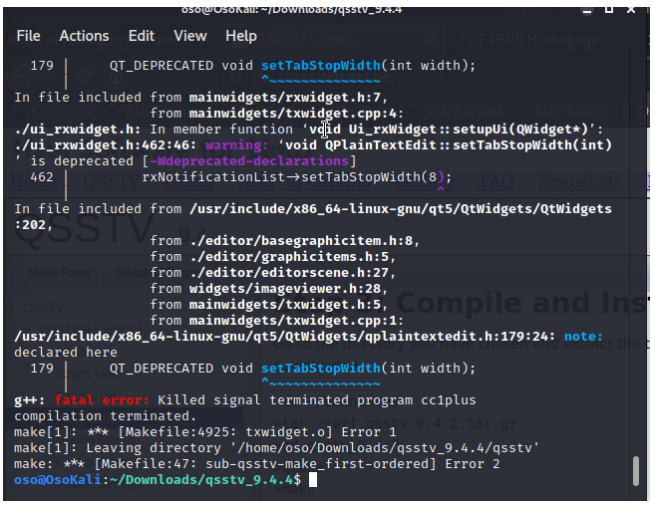
Slow-scan television

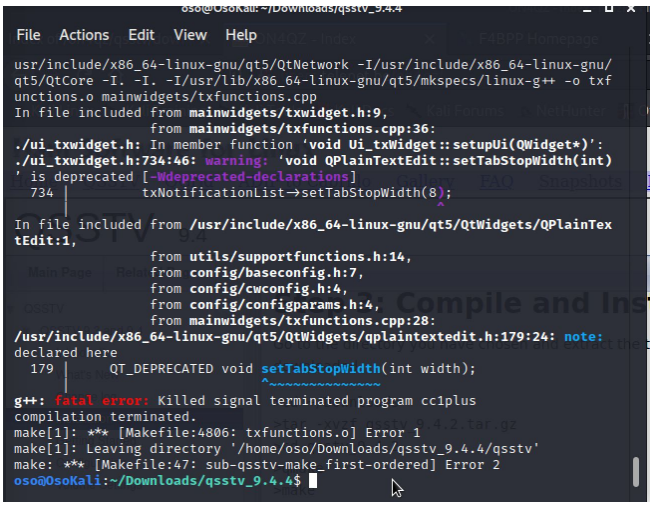
Slow Scan television is a picture transmission method used mainly by amateur radio operators, to transmit and receive static pictures via radio in monochrome or color. A literal term for SSTV is narrowband television. Wikipedia

Under the Modem Software for SSTV, there's a tool for Linux systems called QSSTV. This must be the "'q' tool" that was mentioned in Slack.

While trying to get QSSTV installed, I was running into an issue where the "hamlib-dev" package was unable to be located. Changing the package to "libhamlib-dev" fixed the error







Now, I am waiting for the “make” process of qsstv to finish. It’s taking quite a while. The terminal is showing this warning over and over again.
←

Update: 2 hours later and the make is still running…

Okay annnd it’s now 3 and a half hours since I began trying to get QSSTV installed… The screenshots to the right show the fatal errors that I got whenever the program stopped proceeding for more than an hour or so.. This was extremely frustrating.

If it’s worth any points at all for the last sample, I do know that the output was supposed to be a picture, and I would have been able to see and show that picture if QSSTV had worked or if I had known to try a different method sooner.