DVB-S + MPEG-TS Arsenal Lab (Blackhat Europe 2021)

Overview

DVB-S (Digital Video Broadcasting for Satellite) is one of the most commonly used protocols for transmitting data from satellites to customers on Earth. DVB-S signals carry data ranging from TV stations to in-flight wireless internet connections.

In this lab, you will learn how to receive DVB-S traffic from a software defined radio or other hardware frontend, convert the raw radio signals to data, and interpret data sent by one of the most common satellite transport layer protocols: MPEG-TS.

Starting the Lab

Open a command terminal with `CTRL+ALT+T` and navigate to the lab directory by running the following command:

```
cd ~/satcoms_demo/
```

Next, connect to your virtual SATCOMs workstation by running the following command:

```
./reset_exercise.sh
```

After a few seconds, a command line should open within your lab machine.

Listen to Your Virtual Radio

Since we're inside (and can't see any satellites) we'll be using a simulated radio signal in the same format that you would get from an SDR hardware frontend outside.

Go ahead and check out the data you get from the simulated device by running this command:

```
nc localhost 8118 | xxd
```

Press `CTRL+C` to disconnect after a few seconds.

Understand Your Virtual Radio

Our task today is to convert all of that noise to something meaningful. To do this, we'll use an open-source tool called `leandvb`, part of the awesome `leansdr` toolkit from `pabr` (https://github.com/pabr/leansdr).

To pipe data from your virtual radio to `leandvb` run the following command:

```
nc localhost 8118 | leandvb --standard DVB-S2 --f32 -f 4e6 --gui > /dev/null
```

(NOTE: Don't worry too much about the --f32 and -f 4e6 parameters, they are specific to the radio hardware being used to receive the satellite signal and will likely change depending on your hardware.)

After a few seconds, you should see some graphs depicting the radio signal you have tuned to. Note the distinct circles on the constellation diagram, the arrangement of these circles relates to the specific modulation scheme used by the satellite to encode data into physical radio-waves. In this case, the satellite is using QPSK, but there are many different modulation schemes which offer tradeoffs between bandwidth usage, data rates, and reliability.

Let's go ahead and use `leansdr` to convert the raw radio data into a data stream and look at what we get. Run the following command, wait a 30 or so seconds, then press `CTRL+C` to stop recording data from the satellite. Don't worry if you see some warning messages.

```
nc localhost 8118 | leandvb --standard DVB-S2 --f32 -f 4e6 > my_recording.ts
```

Next, take a look at the first few bytes of your recording using a hex editor.

```
xxd -i 1000 my_recording.ts
```

Note the regular appearance of the uppercase "G" character in the right column of the hex dump. This is an easy way to recognize MPEG-TS encoded satellite traffic, where the letter G (hex 47) is used to delimit packets every 188 bytes.

Explore Your Recording

One of the best tools for interacting with MPEG-TS data is the open source `tsduck` (https://tsduck.io/). Let's use the `tsanalyze` tool here to take a closer look at our recording. Run the following command to see a summary of metadata from the stream you just captured. You can scroll through the data with `SPACE`.

```
tsanalyze my_recording.ts
```

You should see two programs in this satellite feed: `HackerSatTV One` and `HackerSatTV Two`. Both of these programs appear to have `AVC` encoded video data, suggesting that they might be video broadcasts. Let's see if we can watch them!

Watch A Feed

To watch the satellite feed, we are going to use the power `tsp` plugin from `tsduck` (see above). `tsp` allows us to create pipelines of plugins which work together to transform data from MPEG-TS streams.

TSP will also allow us to pipe this data directly to a media player which can support MPEG videos, but we need to know what program we want to tune to.

We can do this all in a single command and listen live to the data coming from `leandvb`. Run the following command to give it a spin, it may take a minute before you get enough traffic to display video - so don't worry.

```
nc localhost 8118 | leandvb --standard DVB-S2 --f32 -f 4e6 | tsp -P zap "HackerSatTV One" -O play
```

The last part of this command (`tsp -P zap "HackerSatTV One -O play`) tells `tsp` to remove all content from the incoming MPEG-TS stream that doesn't belong to the service named "HackerSatTV One" and then pipe that to the system's default media player.

If you've done everything right, you should be able to watch satellite TV through your own DVB-S signal processing pipeline. Congrats!

If you want to explore this more at home, I encourage you to check out MPE/ULE encapsulation - which isn't well supported by `tsduck` yet but is widely used over MPEG-TS feeds to transmit internet traffic and other general purpose data (such as TV receiver firmware updates).