black hat USA 2021

August 4-5, 2021

ARSENAL

Capture the Signal

Running Wireless IoT CTFs, Remotely!

Website: https://cts.ninja

GitHub Repo: https://github.com/capturethesignal

Twitter: @SignalCapture

Capture the Signal: A Blind Signal Analysis Challenge

Jonathan Andersson

The hardest thing about reversing radio signals...

...is that there are **many** challenging aspects of the process.

- We gained blind signal analysis skills from our research
- Experience bootstrapping Mobile P2O
- Discussed ideas with Dragos / Cansec / Pacsec community
- We envisioned an interactive, dynamic contest
- CTS was created to help educate the community

Ideas…

How do we make blind signal analysis into a fun and interesting contest?

- Allow contestants to create static or dynamic solutions
  - Static: using all-in-one tools
  - Dynamic: developing GNURadio flowgraphs
- Include radio reception and transmission as part of the CTS framework
- Allow beginners as well as more advanced contestants to participate

Challenges…

We needed a simple way to deploy the contest at events, and avoid:

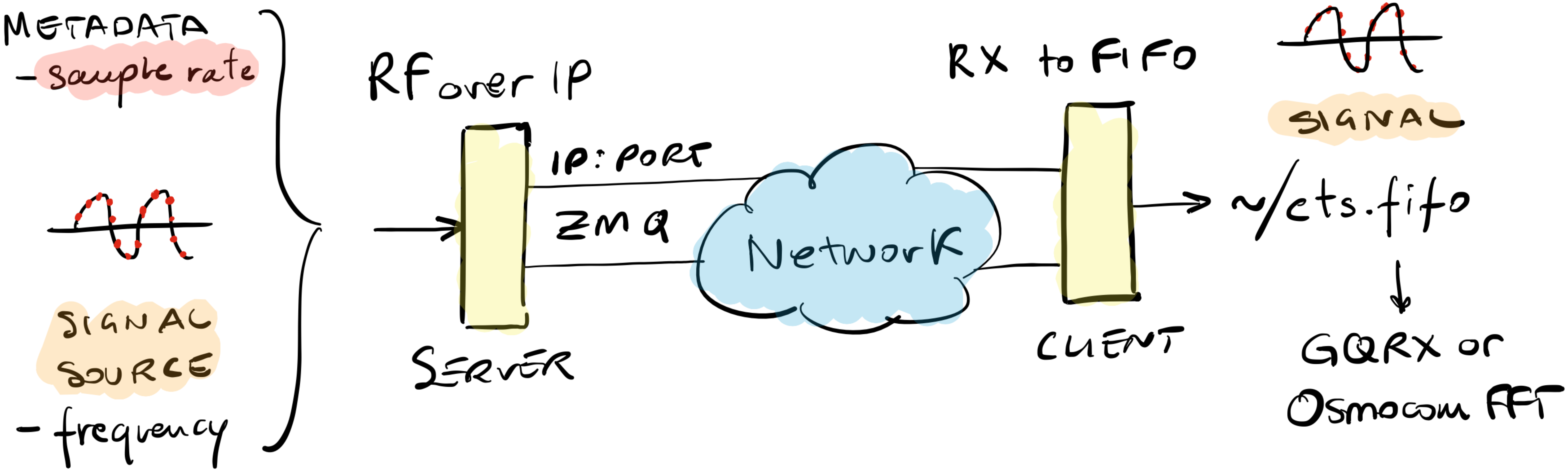
- Dealing with various international regulations and restrictions
- On site reception issues, radio interference, and intentional jamming
- Requiring contestants who may be new to radio to buy and bring SDRs

We envisioned a virtualized CTS framework

It needed to be:

- Simple and lightweight
- Able to stream multiple signals concurrently
- Bandwidth customizable
- Work transparently with GNURadio and other radio-hacking tools

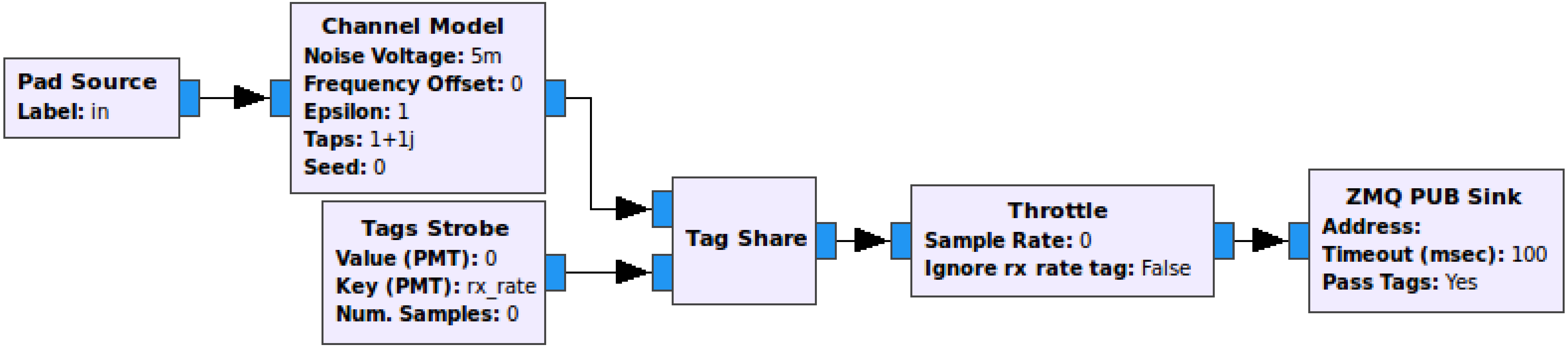
Our Solution: RF Over IP



Our Solution: RF Over IP

- Virtual frequency space divided into configurable, equally sized sections
- Neighboring sections assigned to consecutive ports
- ZeroMQ (TCP PUB/SUB) is used as the underlying transport
- Tuning accomplished by connecting to a specific port
- A simple channel model is inserted before TX to simulate OTA

GNURadio Transmission Sink (offset_tx.grc)



Pad Source
Label: in

Channel Model
Noise Voltage: 5m
Frequency Offset: 0
Epsilon: 1
Taps: 1+1j
Seed: 0

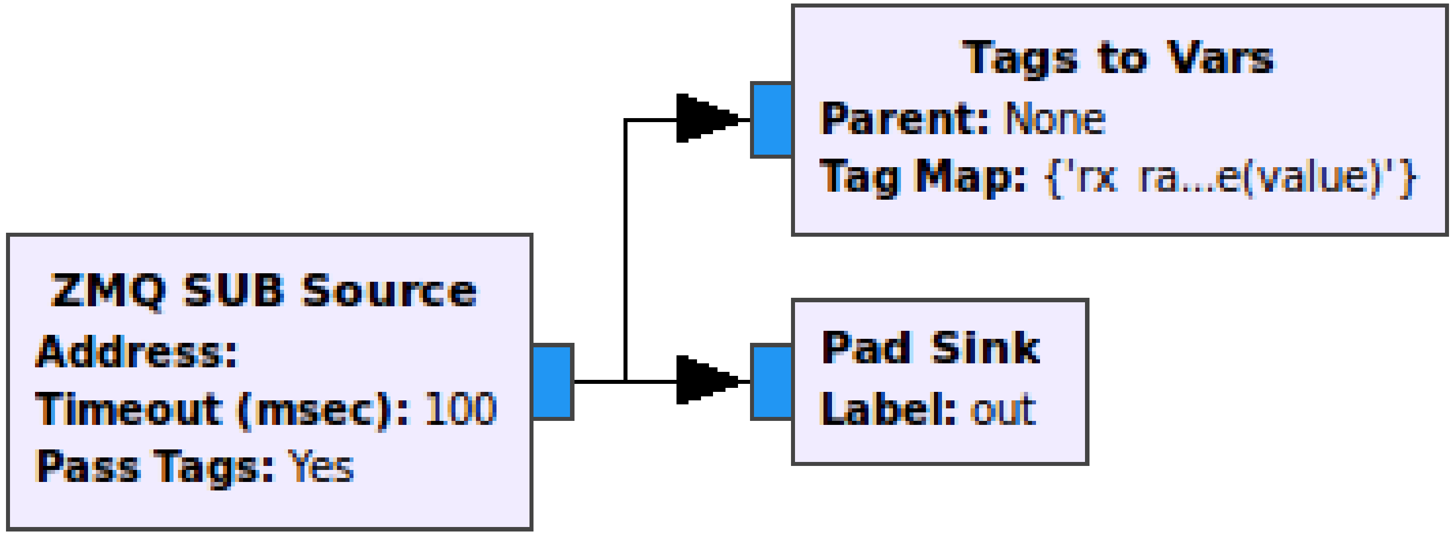
Tags Strobe
Value (PMT): 0
Key (PMT): rx_rate
Num. Samples: 0

Tag Share

Throttle
Sample Rate: 0
Ignore rx rate tag: False

ZMQ PUB Sink
Address:
Timeout (msec): 100
Pass Tags: Yes

GNURadio Reception Source (rf_over_ip_source.grc)

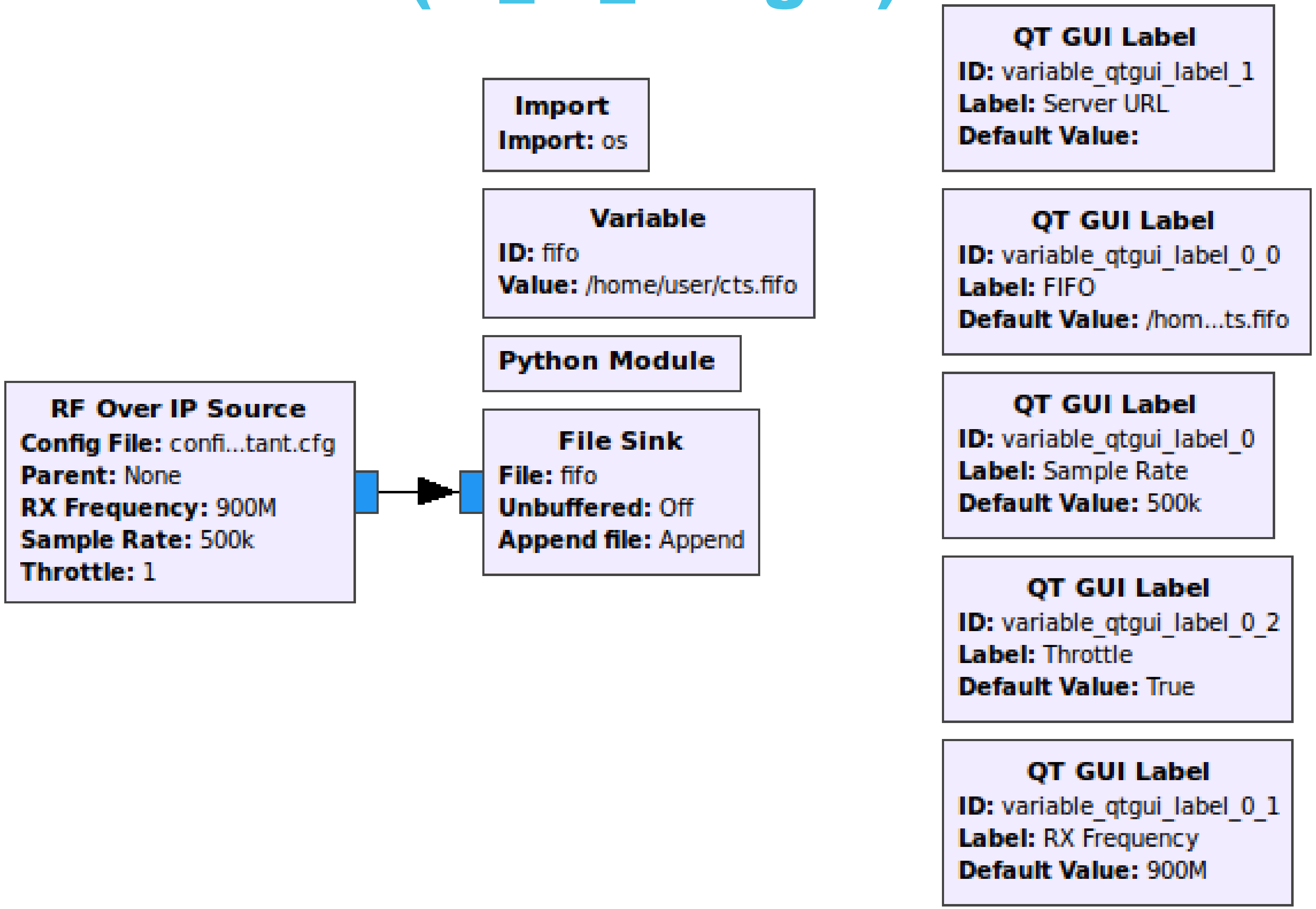


ZMQ SUB Source
Address:
Timeout (msec): 100
Pass Tags: Yes

Tags to Vars
Parent: None
Tag Map: {'rx ra...e(value)'}

Pad Sink
Label: out

Generic FIFO Source (rx_to_fifo.grc)



Import
Import: os

Variable
ID: fifo
Value: /home/user/cts.fifo

Python Module

RF Over IP Source
Config File: confi...tant.cfg
Parent: None
RX Frequency: 900M
Sample Rate: 500k
Throttle: 1

File Sink
File: fifo
Unbuffered: Off
Append file: Append

QT GUI Label
ID: variable_qtgui_label_1
Label: Server URL
Default Value:

QT GUI Label
ID: variable_qtgui_label_0_0
Label: FIFO
Default Value: /hom...ts.fifo

QT GUI Label
ID: variable_qtgui_label_0
Label: Sample Rate
Default Value: 500k

QT GUI Label
ID: variable_qtgui_label_0_2
Label: Throttle
Default Value: True

QT GUI Label
ID: variable_qtgui_label_0_1
Label: RX Frequency
Default Value: 900M

Presence at Security Conferences

The Radio Village

Marco Balduzzi (@embyte)

Hands-on at conferences

2018

Tokyo

PAC SEC

HITB

Dubai

2019

Tokyo

PAC SEC

Vancouver

CAN SEC WEST

HITB

Amsterdam

HITB

Abu Dhabi

2020-2021

Virtual

hardwear.io

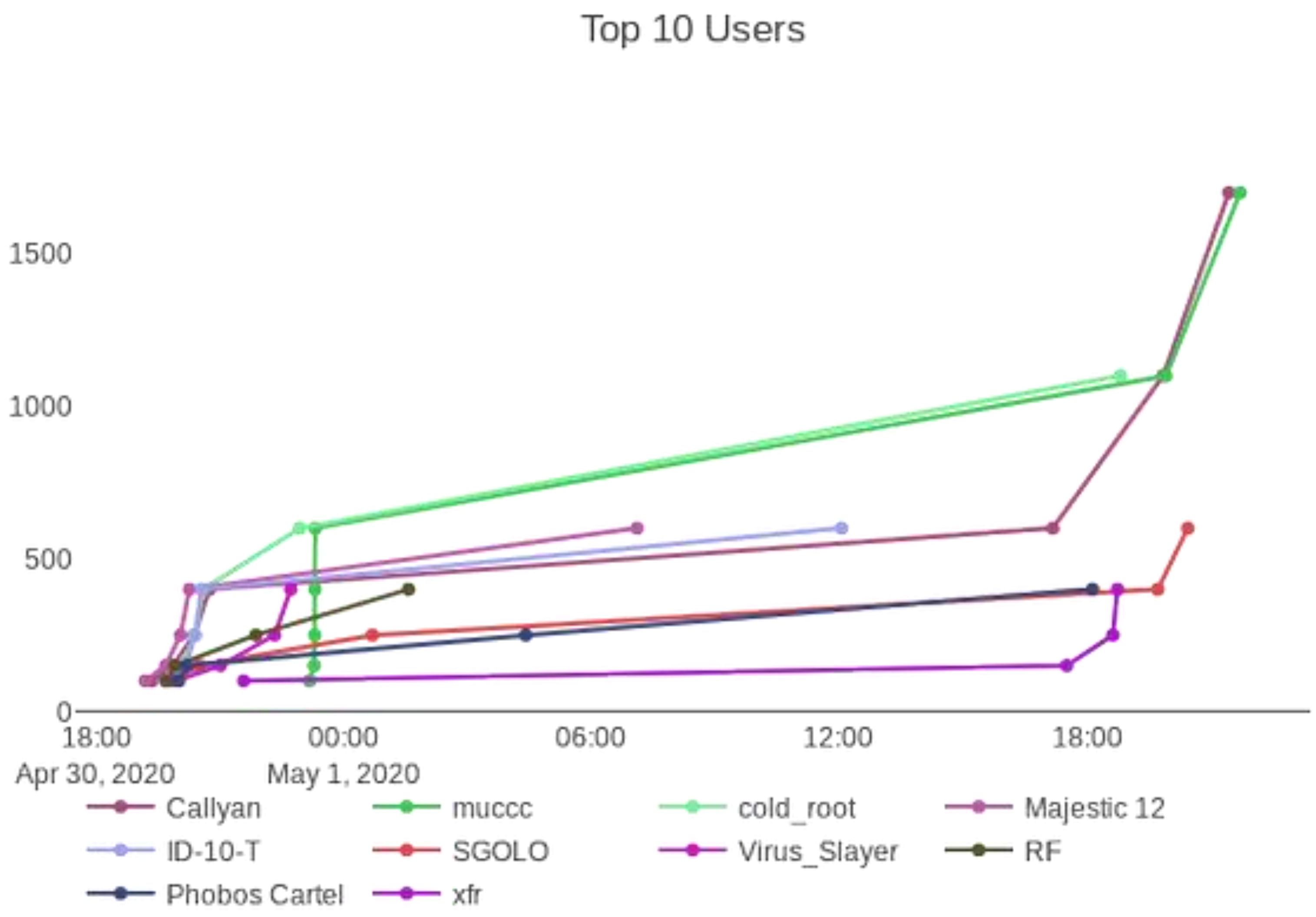
hardwear.io

HITB

Gadgets, radios, hackathon, fun

```
user@dell:~$ cat nano.save
        JOIN THE CAPTURE-THE-SIGNAL CHALL
             PREPARATION PHASE IS NOW
JOIN SLACK> https://tinyurl.com/HITB-AMS-CTS-2019
FOLLOW US > @SignalCapture @TrendMicroRSRCH
ww        > https://trendmicro.com/cts
user@dell:~$
```

But remember, it's a real competition...



337 users registered

223 teams registered

492 IP addresses

2860 total possible points

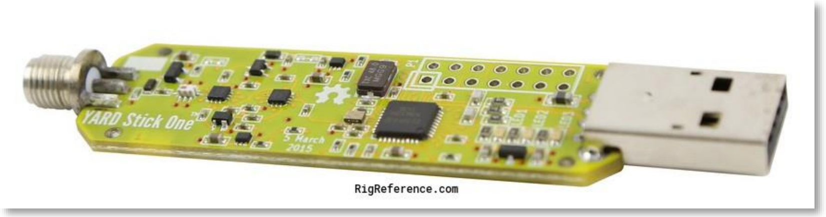
9 challenges

Signal 1 has the most solves with 31 solves

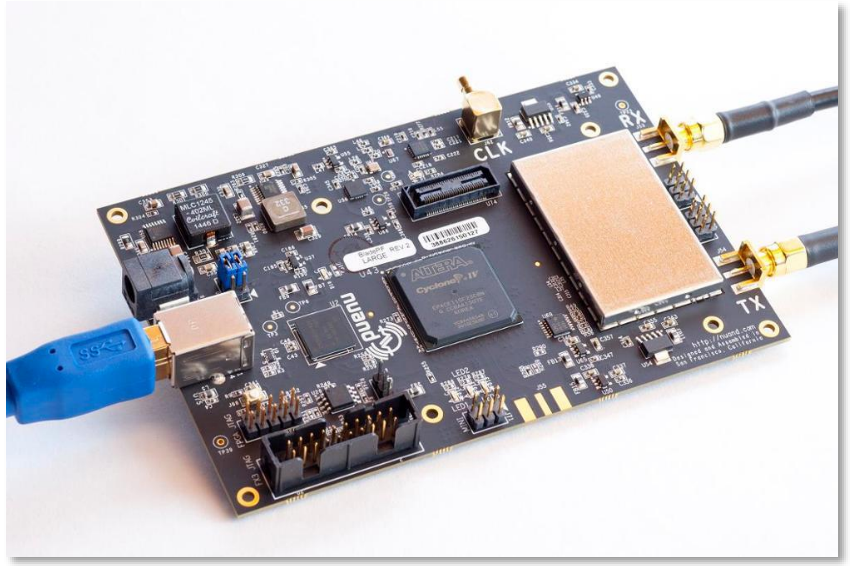
Signal 5 has the least solves with 8 solves

With prizes

Bronze



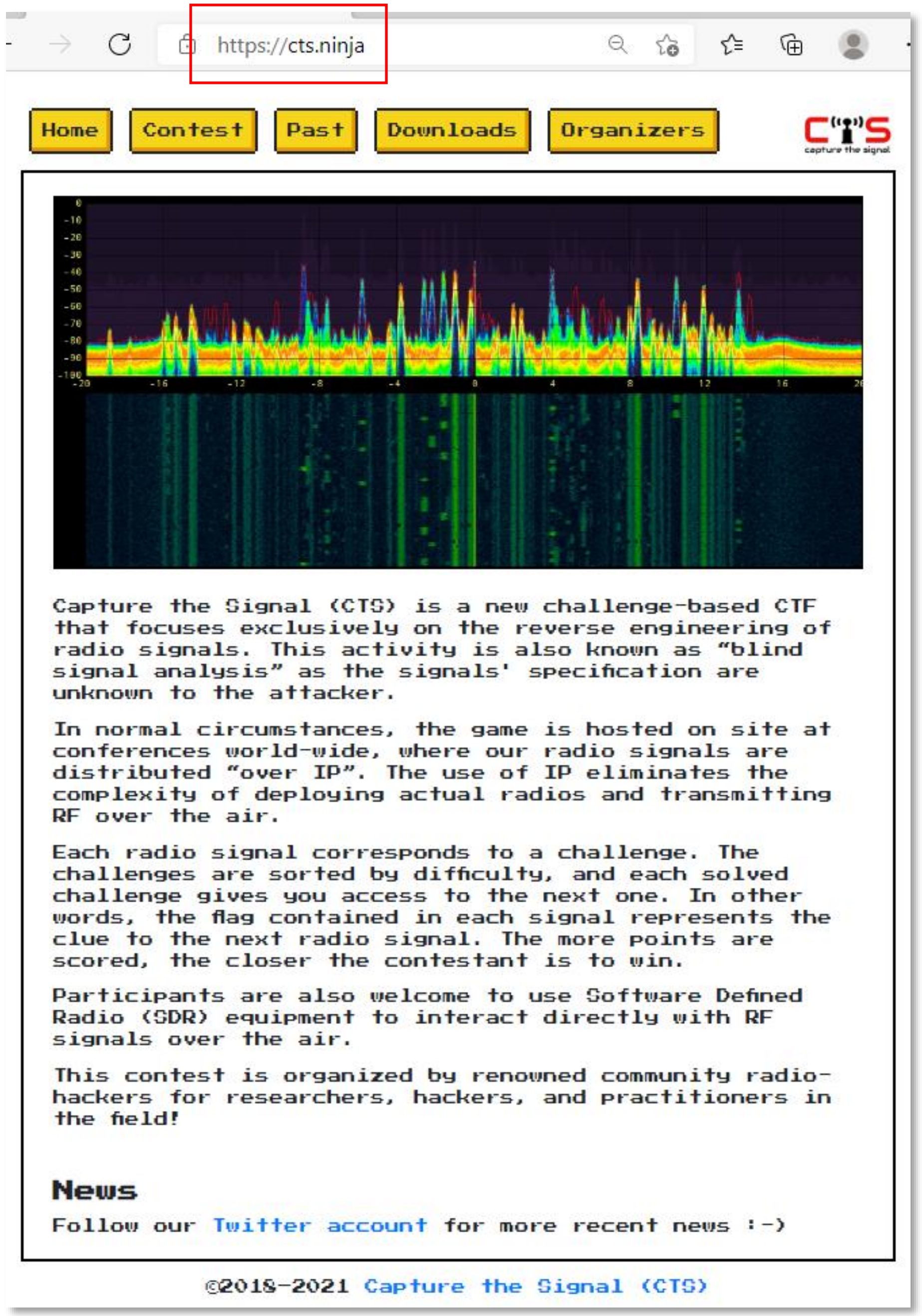
Silver



Gold

Awards Ceremony ☺

Open and community spirit



Capture the Signal (CTS) is a new challenge-based CTF that focuses exclusively on the reverse engineering of radio signals. This activity is also known as "blind signal analysis" as the signals' specification are unknown to the attacker.

In normal circumstances, the game is hosted on site at conferences world-wide, where our radio signals are distributed "over IP". The use of IP eliminates the complexity of deploying actual radios and transmitting RF over the air.

Each radio signal corresponds to a challenge. The challenges are sorted by difficulty, and each solved challenge gives you access to the next one. In other words, the flag contained in each signal represents the clue to the next radio signal. The more points are scored, the closer the contestant is to win.

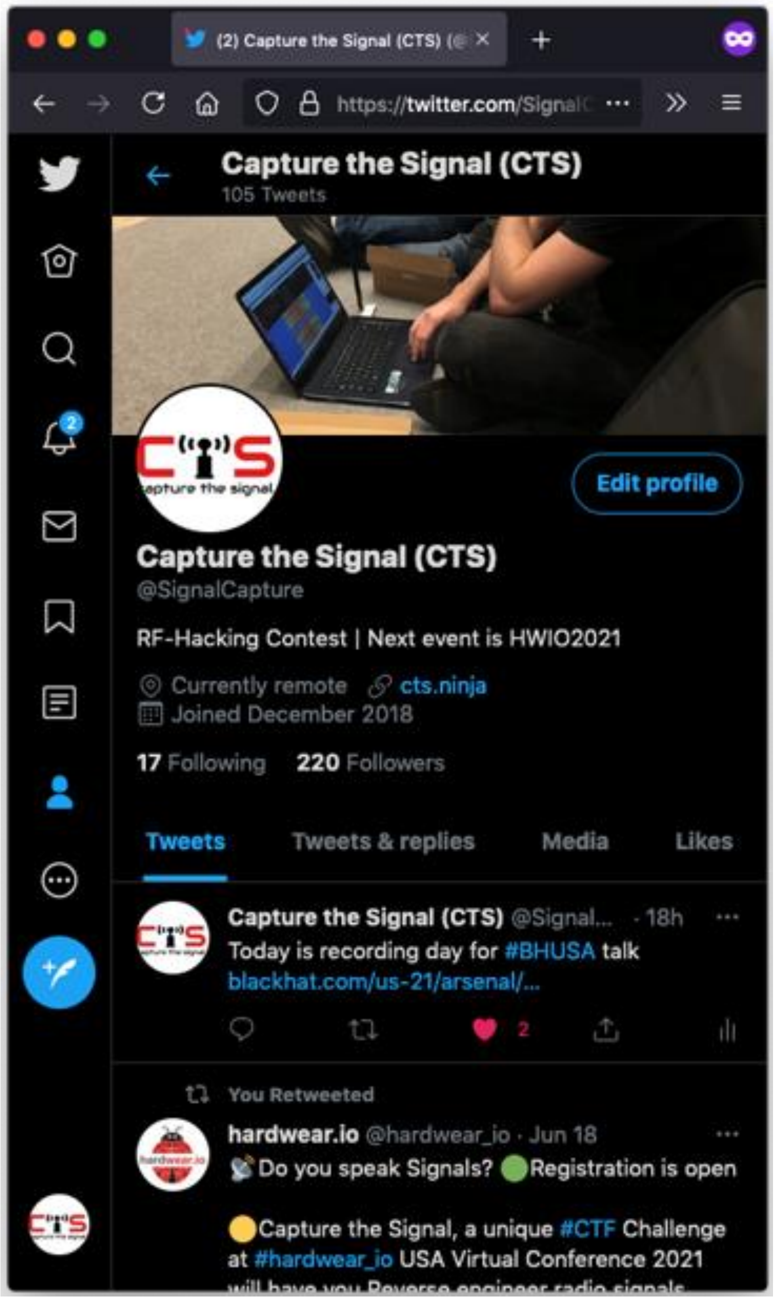
Participants are also welcome to use Software Defined Radio (SDR) equipment to interact directly with RF signals over the air.

This contest is organized by renowned community radio-hackers for researchers, hackers, and practitioners in the field!

News

Follow our Twitter account for more recent news :-)

©2018-2021 Capture the Signal (CTS)



Capture the Signal (CTS)
105 Tweets

Capture the Signal (CTS)
@SignalCapture

RF-Hacking Contest | Next event is HWIO2021

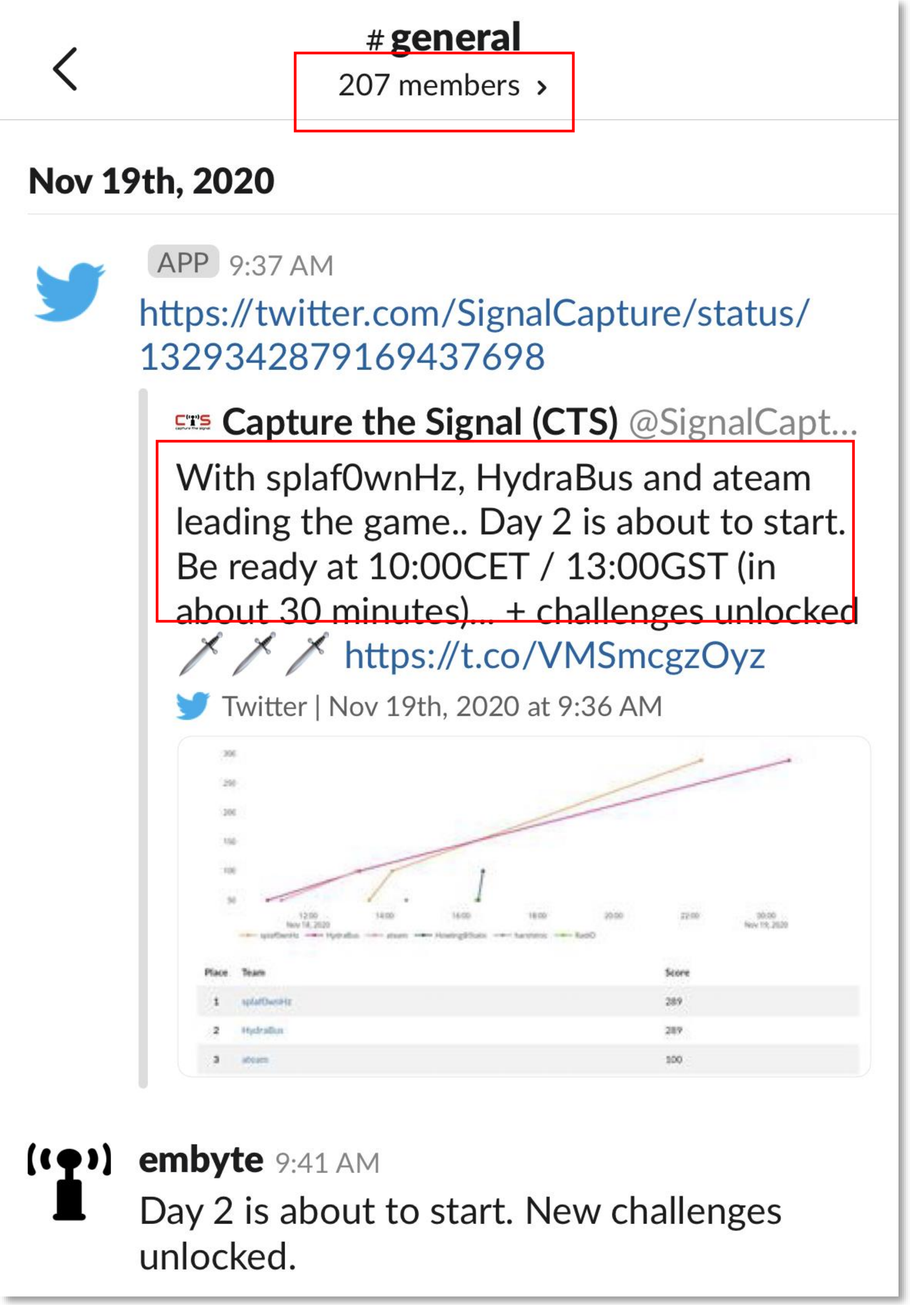
Currently remote   cts.ninja
Joined December 2018

17 Following   220 Followers

Capture the Signal (CTS) @Signal... · 18h
Today is recording day for #BHUSA talk blackhat.com/us-21/arsenal/...
2

You Retweeted
hardwear.io @hardwear_io · Jun 18
📡Do you speak Signals? 🟢Registration is open

🟡Capture the Signal, a unique #CTF Challenge at #hardwear_io USA Virtual Conference 2021 will have you Reverse engineer radio signals



Nov 19th, 2020

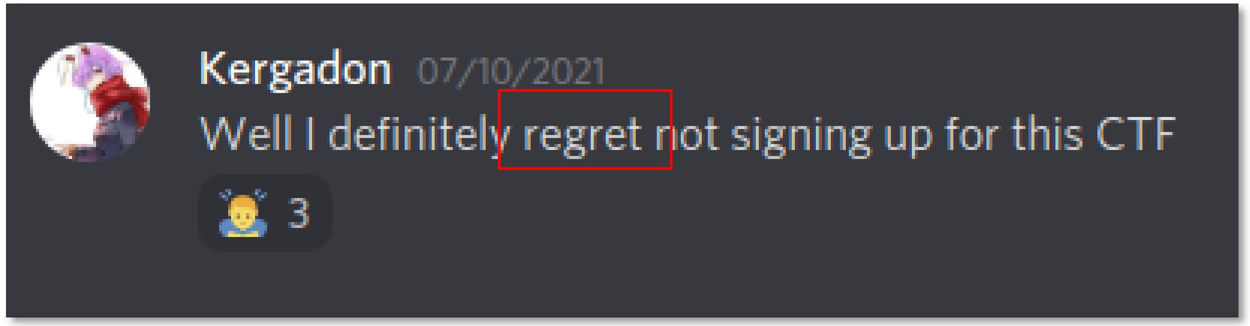
APP 9:37 AM
https://twitter.com/SignalCapture/status/1329342879169437698

Capture the Signal (CTS) @SignalCapt...
With splaf0wnHz, HydraBus and ateam leading the game.. Day 2 is about to start. Be ready at 10:00CET / 13:00GST (in about 30 minutes)... + challenges unlocked 🗡️🗡️🗡️ https://t.co/VMSmcgzOyz
Twitter | Nov 19th, 2020 at 9:36 AM

Place	Team	Score
1	splaf0wnHz	289
2	HydraBus	289
3	ateam	100

embyte 9:41 AM
Day 2 is about to start. New challenges unlocked.

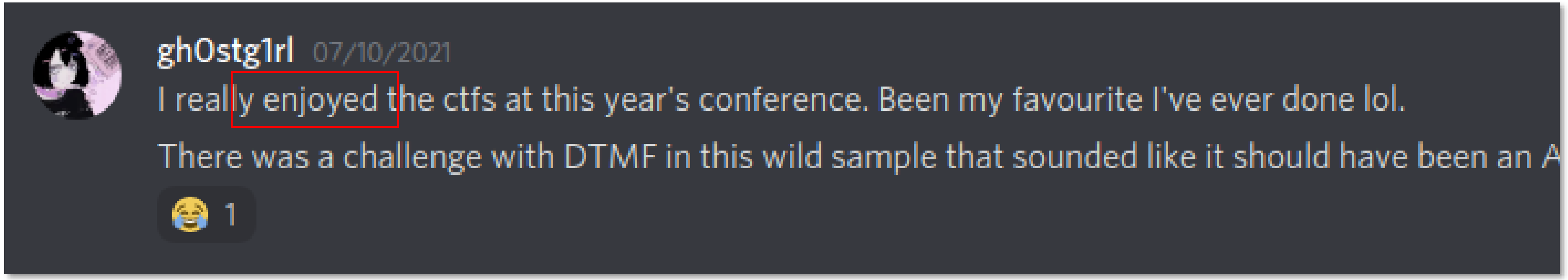
Fun!



Kergadon 07/10/2021

Well I definitely regret not signing up for this CTF

🙇 3

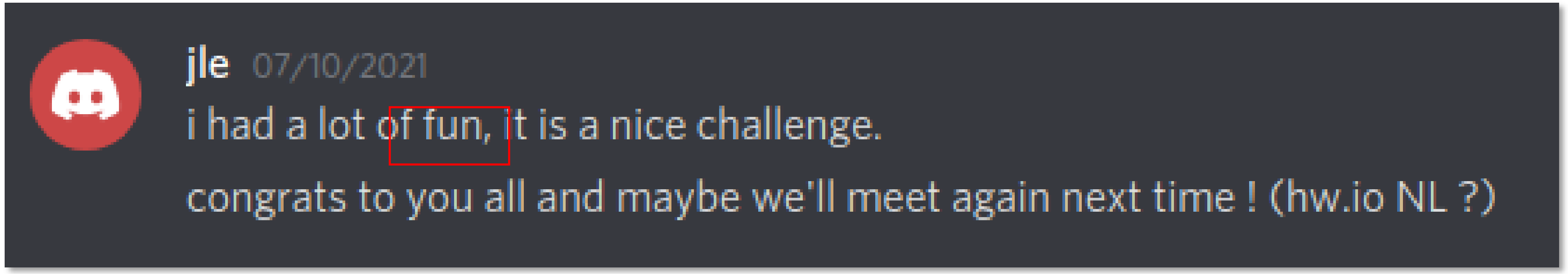


gh0stg1rl 07/10/2021

I really enjoyed the ctfs at this year's conference. Been my favourite I've ever done lol.

There was a challenge with DTMF in this wild sample that sounded like it should have been an A

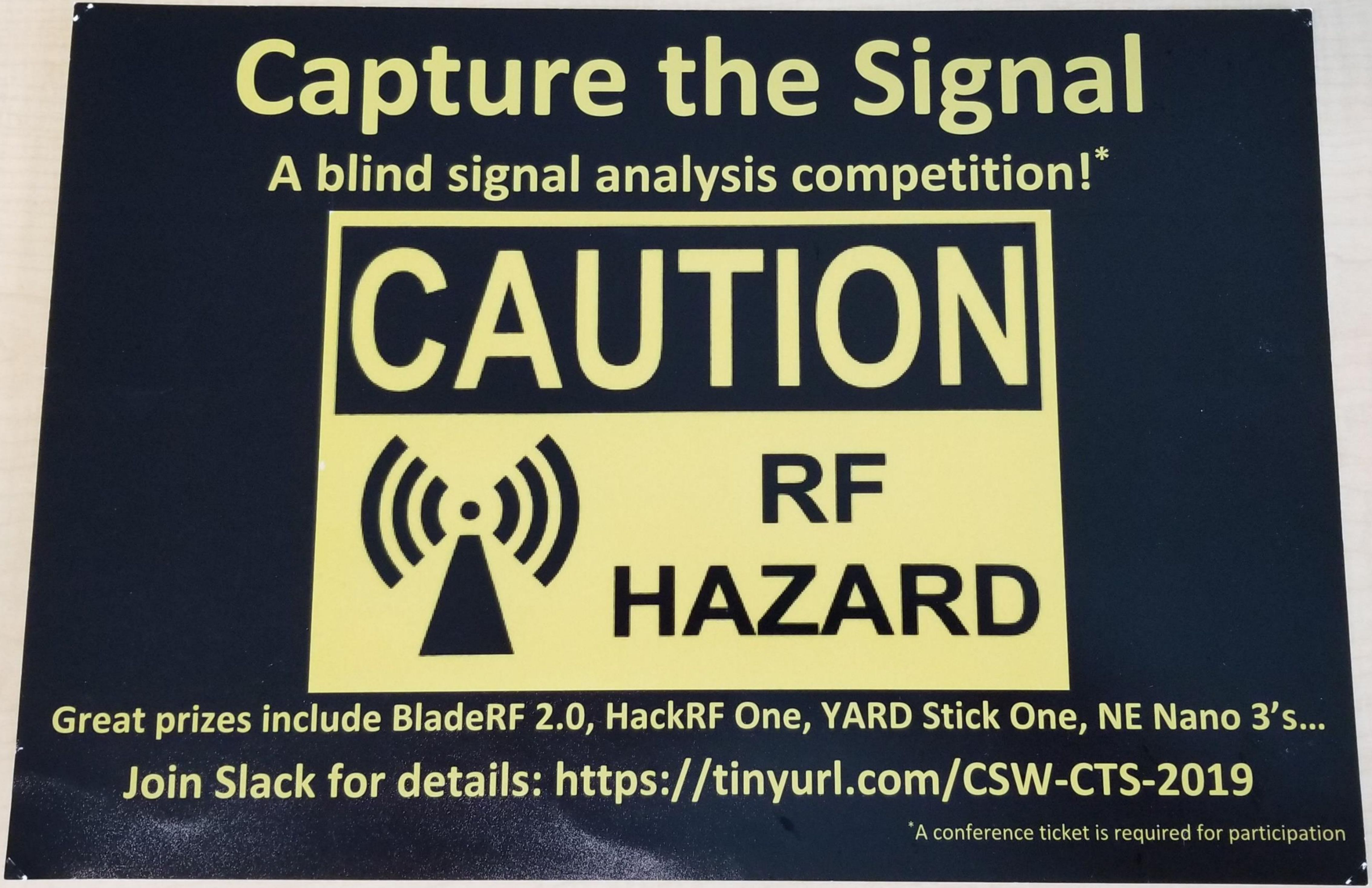
😭 1



jle 07/10/2021

i had a lot of fun, it is a nice challenge.

congrats to you all and maybe we'll meet again next time ! (hw.io NL ?)



Capture the Signal

A blind signal analysis competition!*

CAUTION

RF HAZARD

Great prizes include BladeRF 2.0, HackRF One, YARD Stick One, NE Nano 3’s…

Join Slack for details: https://tinyurl.com/CSW-CTS-2019

*A conference ticket is required for participation

Learning experience

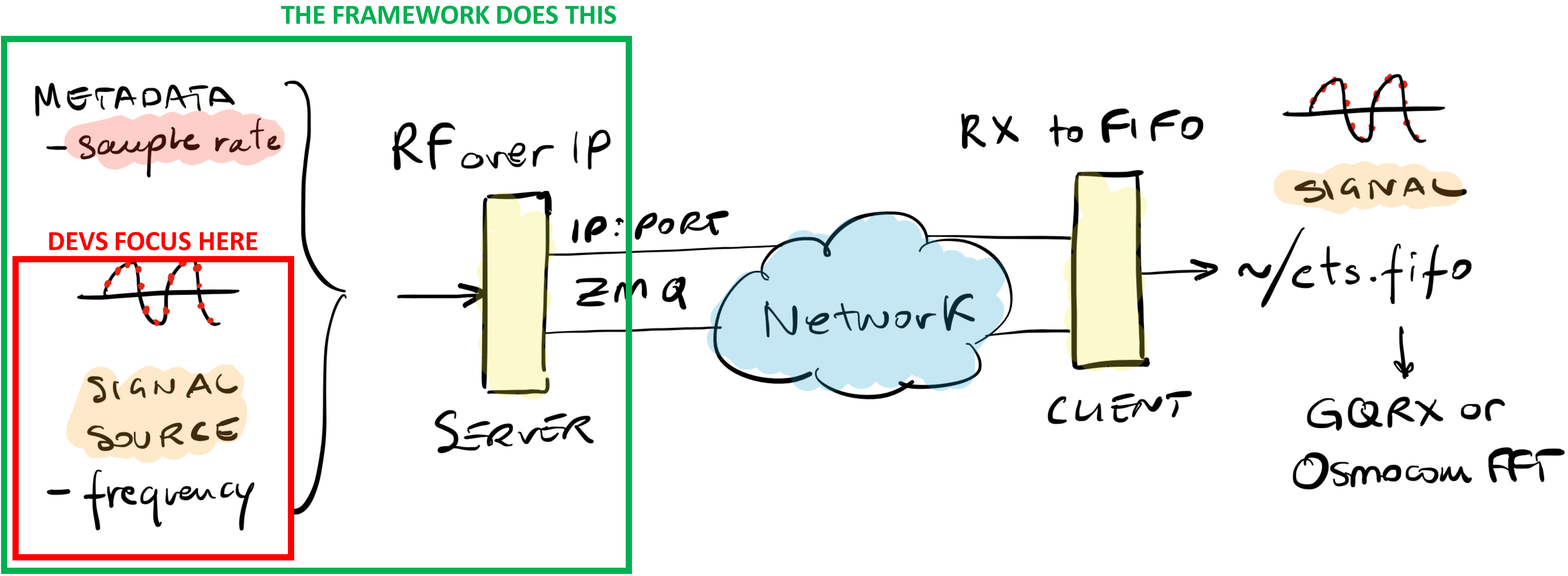
How to Scale: From a radio village to a global contest

Federico Maggi

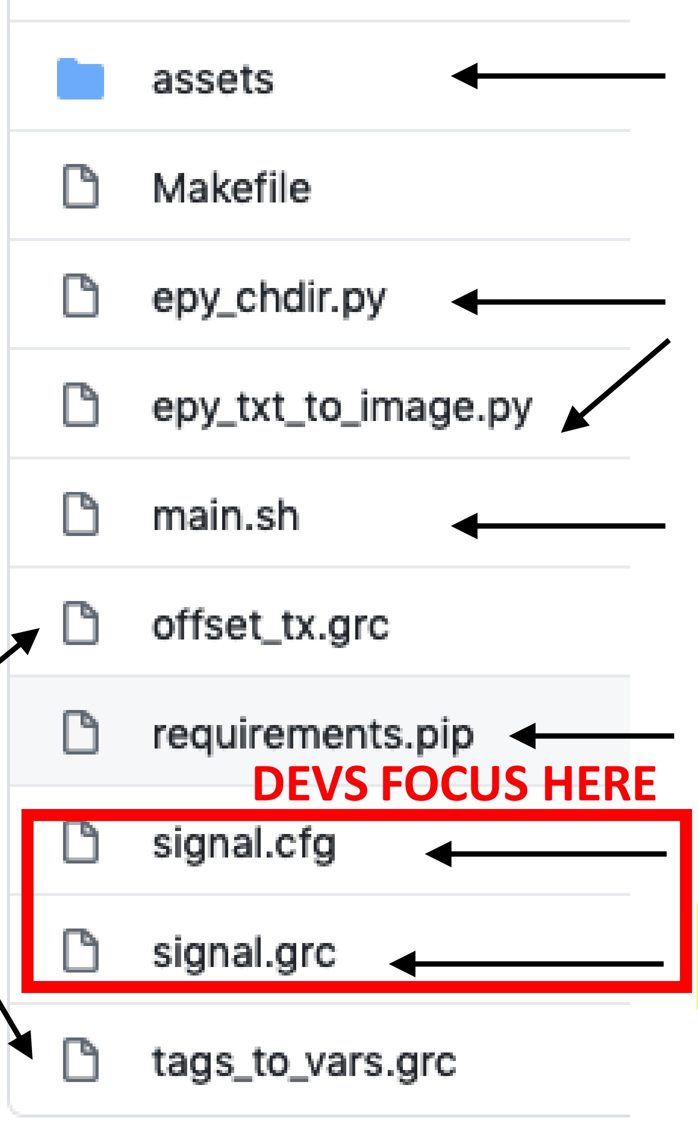
From GNURadio Flowgraphs to Headless Python Scripts

- No **GUI** (unless you want to burn memory to run Xvfb over Docker)
- Make flowgraphs as **parametric** as possible
- Use a **configuration file** if possible
- Plan for **unattended** compilation and deployment

RF Over IP: Backend



Our recipe to package a challenge



- assets
- Makefile
- epy_chdir.py
- epy_txt_to_image.py
- main.sh
- offset_tx.grc
- requirements.pip
- signal.cfg
- signal.grc
- tags_to_vars.grc

Resource files (e.g., png, wav, mp3)

Additional Python files

The entry point for this challenge

Python requirements (used by GR too)

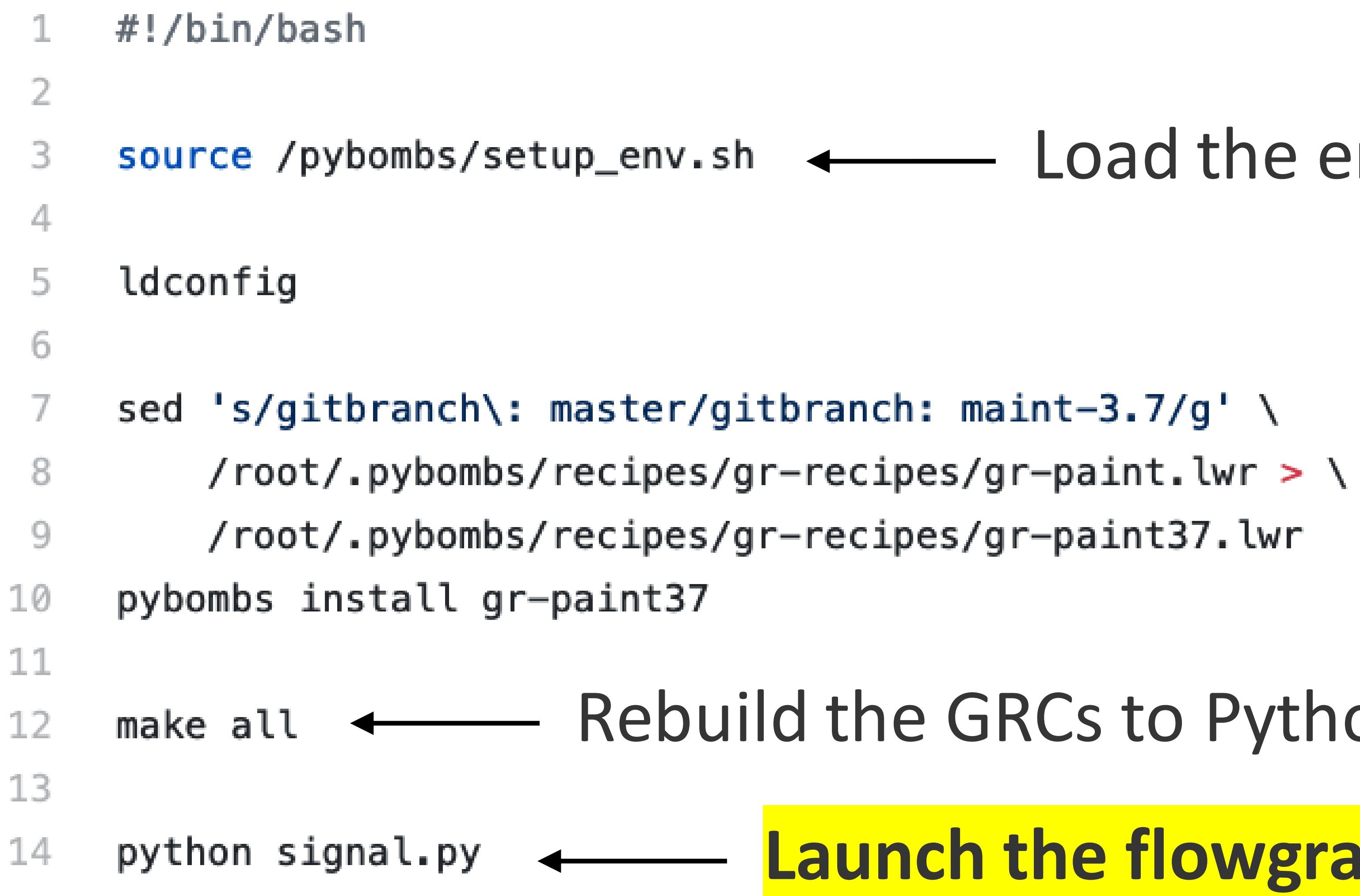
DEVS FOCUS HERE

The challenge's configuration file

The flowgraph's GRC file

Additional GRC files (e.g., blocks)

The entry point : main.sh



```
#!/bin/bash

source /pybombs/setup_env.sh

ldconfig

sed 's/gitbranch\: master/gitbranch: maint-3.7/g' \
    /root/.pybombs/recipes/gr-recipes/gr-paint.lwr > \
    /root/.pybombs/recipes/gr-recipes/gr-paint37.lwr
pybombs install gr-paint37

make all

python signal.py
```

Load the environment variables

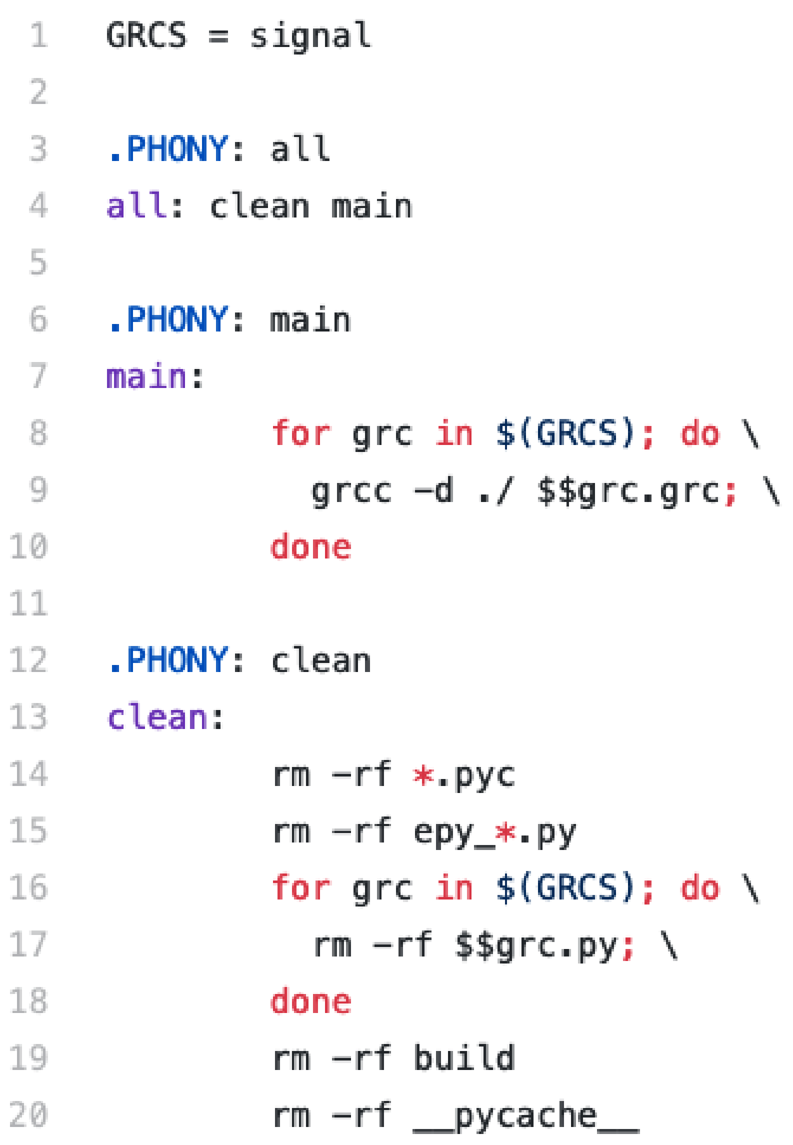
Anything else you need to do

Rebuild the GRCs to Python

Launch the flowgraph in foreground

The Makefile

THE FRAMEWORK DOES THIS

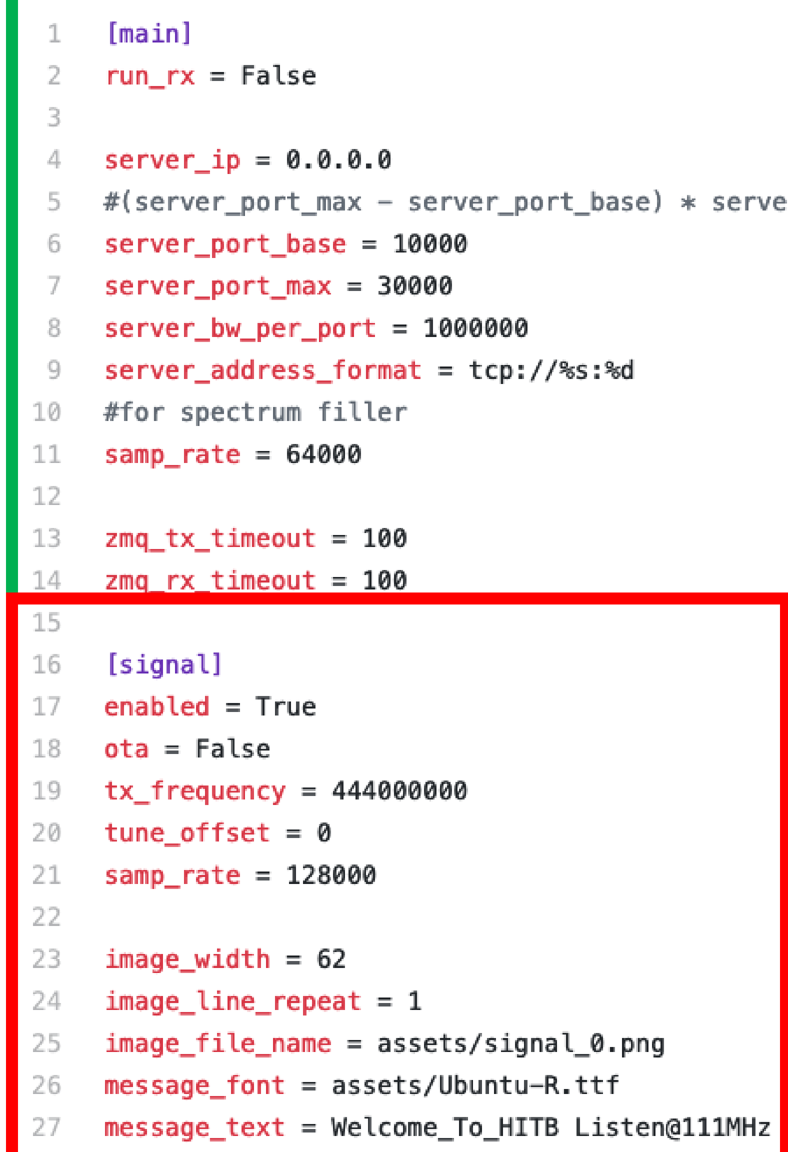


```
GRCS = signal

.PHONY: all
all: clean main

.PHONY: main
main:
	for grc in $(GRCS); do \
	  grcc -d ./ $$grc.grc; \
	done

.PHONY: clean
clean:
	rm -rf *.pyc
	rm -rf epy_*.py
	for grc in $(GRCS); do \
	  rm -rf $$grc.py; \
	done
	rm -rf build
	rm -rf __pycache__
```

Compile all GRCs into Python files (yes, GNURadio has a CLI compiler)

Cleanup

Configuration file: signal.cfg



```
[main]
run_rx = False

server_ip = 0.0.0.0
#(server_port_max - server_port_base) * server_bw_per_port = 20GHz
server_port_base = 10000
server_port_max = 30000
server_bw_per_port = 1000000
server_address_format = tcp://%s:%d
#for spectrum filler
samp_rate = 64000

zmq_tx_timeout = 100
zmq_rx_timeout = 100

[signal]
enabled = True
ota = False
tx_frequency = 444000000
tune_offset = 0
samp_rate = 128000

image_width = 62
image_line_repeat = 1
image_file_name = assets/signal_0.png
message_font = assets/Ubuntu-R.ttf
message_text = Welcome_To_HITB Listen@111MHz
```

FRAMEWORK

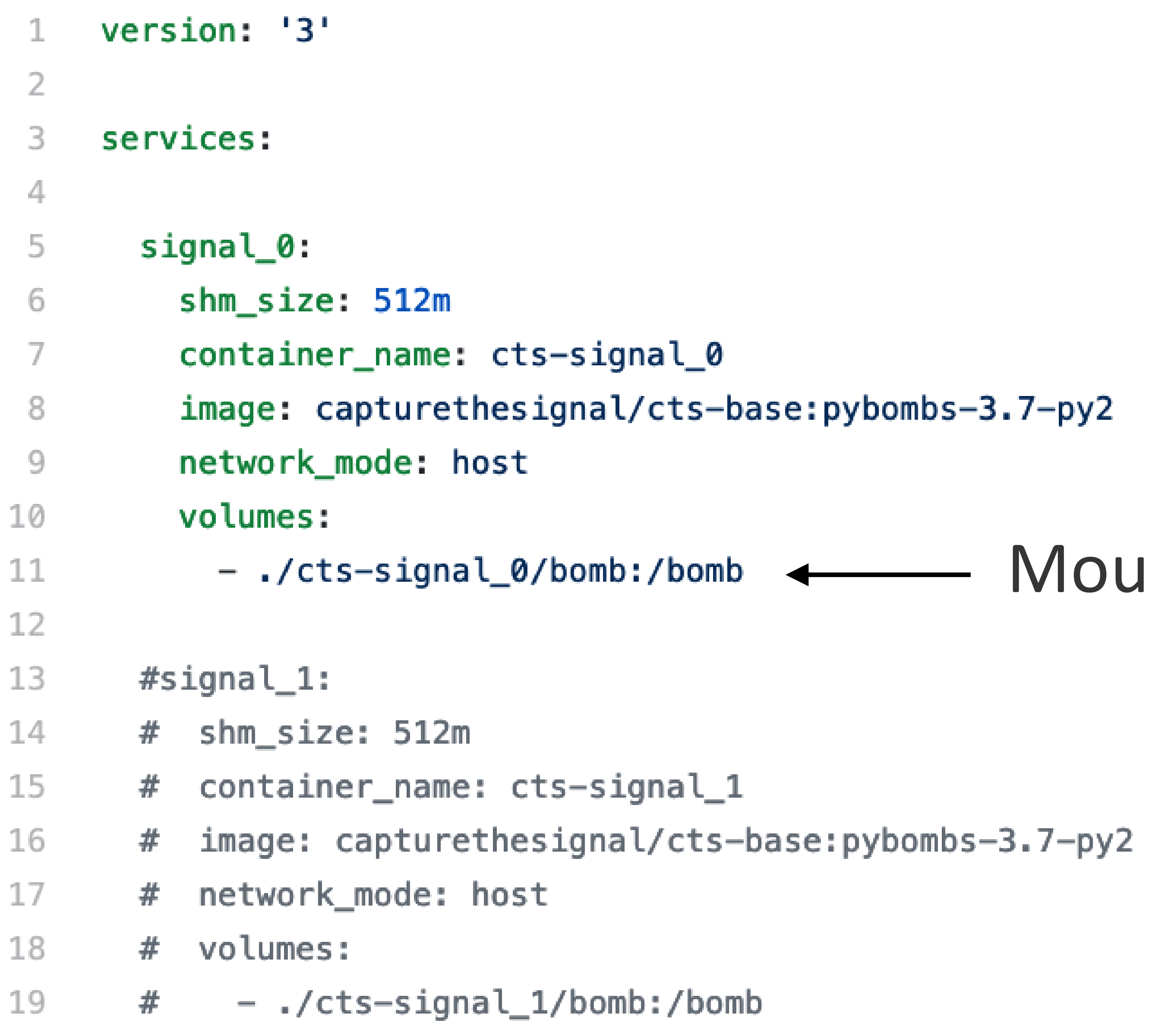
Challenge-specific options

DEVS

How to create new challenges from this template

- Make a copy of a signal that is known to work
- Work on your `signal.grc` (e.g., using GNURadio Companion) and bare in mind that everything should be self-contained
- Test that your signal works in GNURadio
- Remove/disable any GUI element

Glue all together with Docker (Compose)



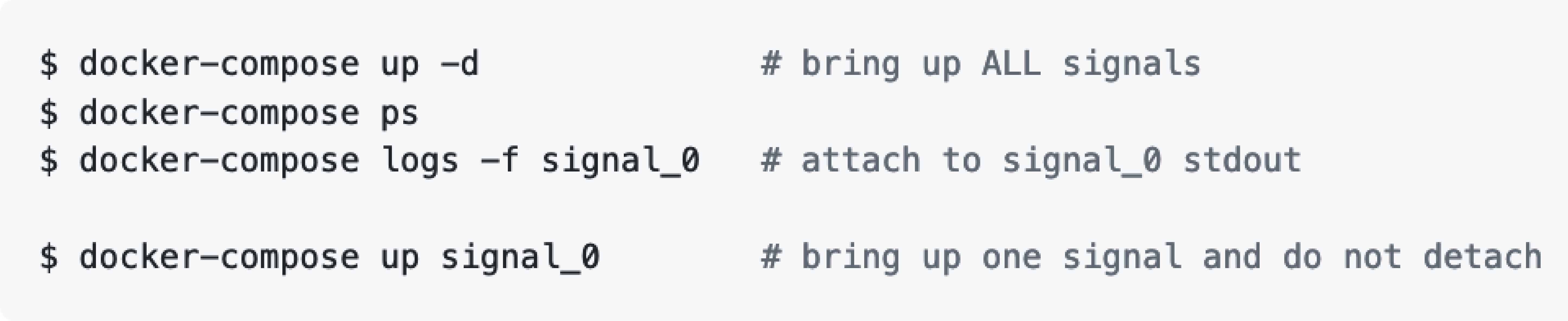
```
version: '3'

services:

  signal_0:
    shm_size: 512m
    container_name: cts-signal_0
    image: capturethesignal/cts-base:pybombs-3.7-py2
    network_mode: host
    volumes:
      - ./cts-signal_0/bomb:/bomb

  #signal_1:
  #  shm_size: 512m
  #  container_name: cts-signal_1
  #  image: capturethesignal/cts-base:pybombs-3.7-py2
  #  network_mode: host
  #  volumes:
  #    - ./cts-signal_1/bomb:/bomb
```

← We have ready-to-use images

← Mount the challenge's path

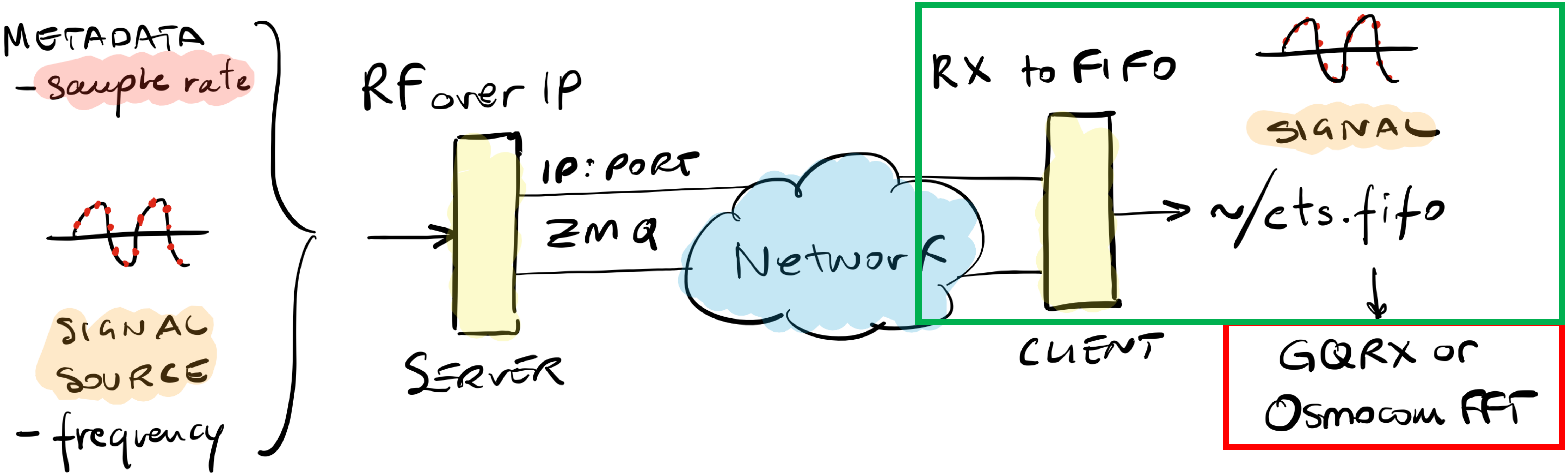
Replicate for each challenge

Launch!



```
$ docker-compose up -d                  # bring up ALL signals
$ docker-compose ps
$ docker-compose logs -f signal_0       # attach to signal_0 stdout

$ docker-compose up signal_0            # bring up one signal and do not detach
```

RF Over IP: Client side



THE FRAMEWORK DOES THIS

CONTESTANTS PLAY HERE

Contestants: Receive RF streams over IP

FRAMEWORK

```
$ git pull https://github.com/capturethesignal/cts-tools
$ cd cts-tools/cts-cli
$ python rx_to_fifo.py                                         \
      --server-ip=<this is given by the organizers>            \
      --rx-frequency=<each challenge has its own frequency>
Tags to Var imported
Tags to Var initialized

Created FIFO: /path/to/cts.fifo
```

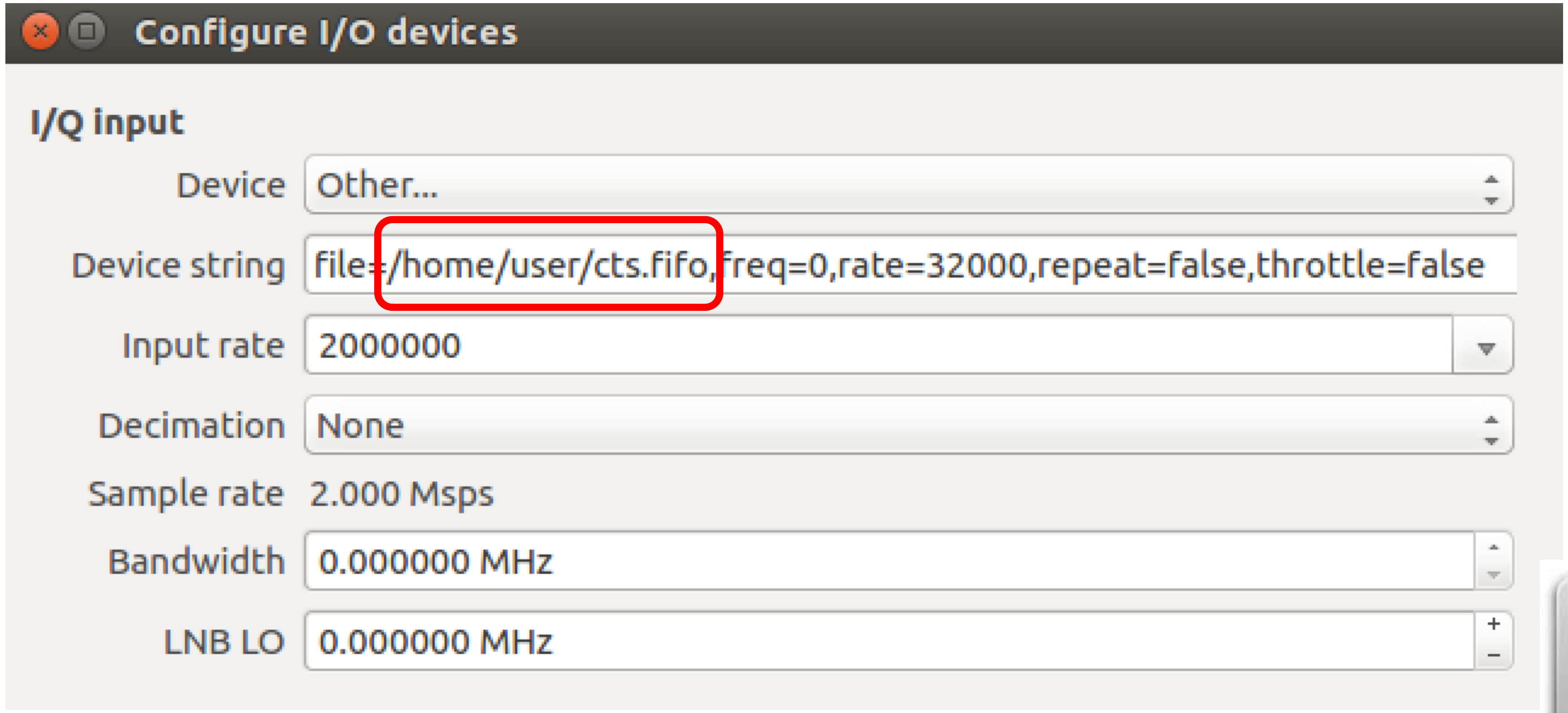
Contestants: What’s the sample rate?

FRAMEWORK

```
...continued from above...

$ cat /path/to/cts.fifo > /dev/null  # will trigger RX of first sample

rx_rate = 128000.0
```

Take note of the **sample rate**

Contestants: Test RX (no tuning needed!)



Configure I/O devices

I/Q input

Device: Other...

Device string: file=/home/user/cts.fifo,freq=0,rate=32000,repeat=false,throttle=false

Input rate: 2000000

Decimation: None

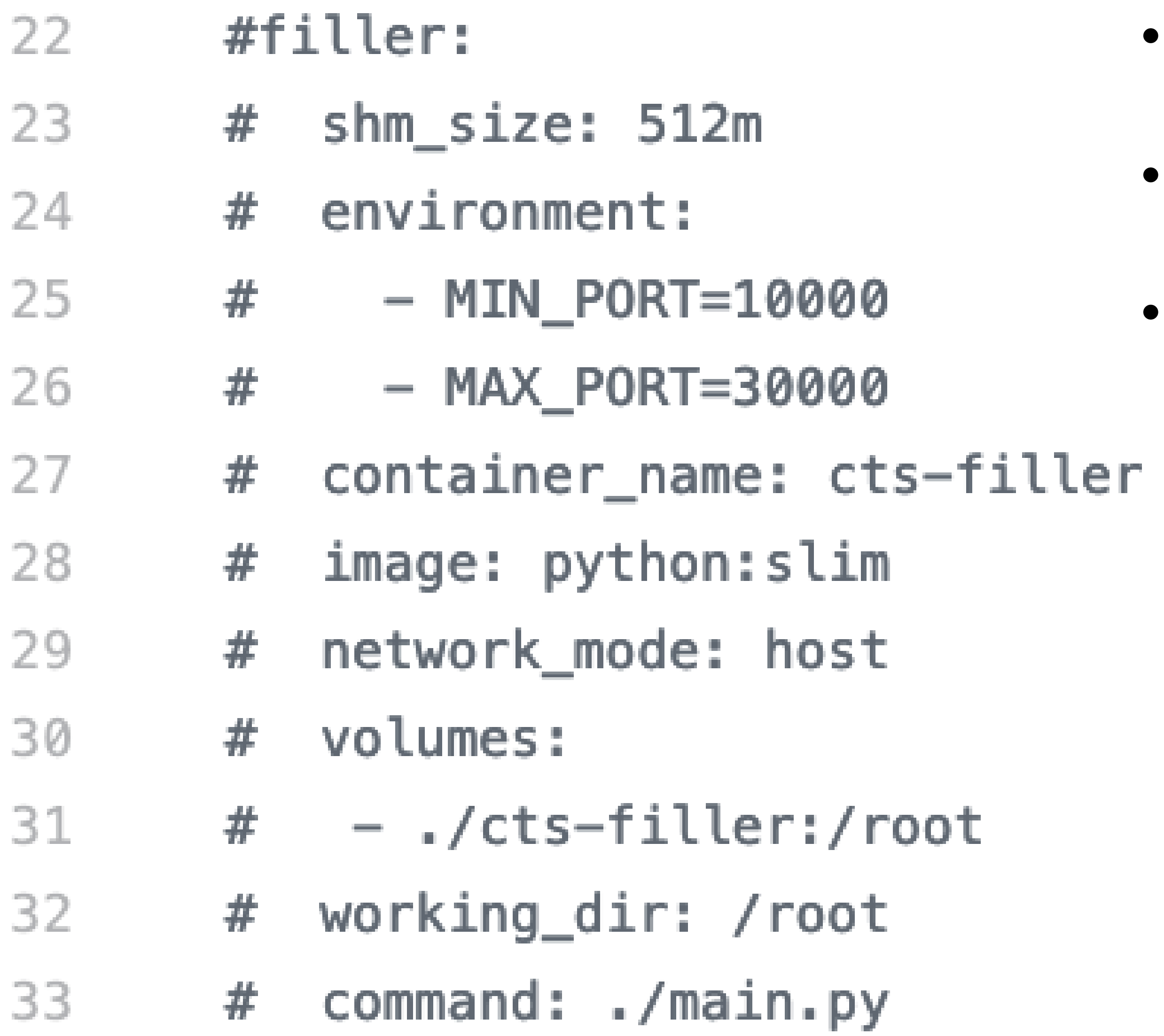
Sample rate: 2.000 Msps

Bandwidth: 0.000000 MHz

LNB LO: 0.000000 MHz

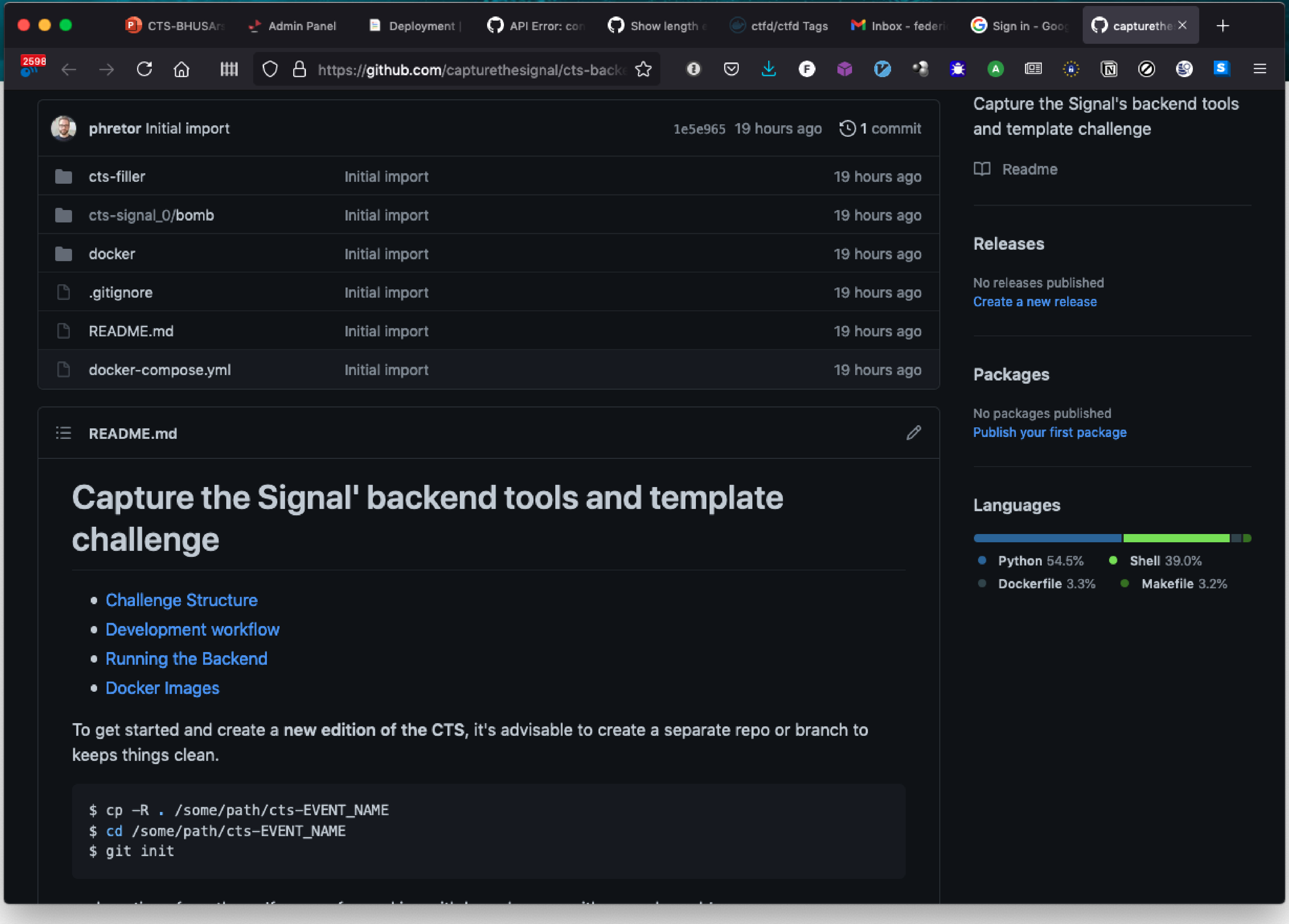
```
$ osmocom_fft -F \
    --args="file=/home/user/cts.fifo,freq=0,rate=32000,repeat=false,throttle=false"
```

Make it slightly harder to cheat (cheaters will still cheat)



```
#filler:
#  shm_size: 512m
#  environment:
#    - MIN_PORT=10000
#    - MAX_PORT=30000
#  container_name: cts-filler
#  image: python:slim
#  network_mode: host
#  volumes:
#   - ./cts-filler:/root
#  working_dir: /root
#  command: ./main.py
```

- Ideally, this should broadcast a random but valid signal
- Here, just keep ports occupied
- Cheaters will still port-scan and do recon to find true signals (and spoil the fun)



phretor Initial import 1e5e965 19 hours ago 1 commit

cts-filler	Initial import	19 hours ago
cts-signal_0/bomb	Initial import	19 hours ago
docker	Initial import	19 hours ago
.gitignore	Initial import	19 hours ago
README.md	Initial import	19 hours ago
docker-compose.yml	Initial import	19 hours ago

README.md

Capture the Signal' backend tools and template challenge

- Challenge Structure
- Development workflow
- Running the Backend
- Docker Images

To get started and create a **new edition of the CTS**, it's advisable to create a separate repo or branch to keeps things clean.

```
$ cp -R . /some/path/cts-EVENT_NAME
$ cd /some/path/cts-EVENT_NAME
$ git init
```

Capture the Signal's backend tools and template challenge

Readme

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Python 54.5% Shell 39.0%
Dockerfile 3.3% Makefile 3.2%

QUICK DEMO

Requirements for a successful contest

- 1 VM for the scoreboard (we use CTFd)
- 1 bigger VM to stream signals (Docker Engine + Docker Compose)
- We keep everything in a virtual private cloud
- 1+ external-facing load balancer(s)
- …
- Prizes and stickers to motivate your attendees

Wishlist: what we hope the community will do

- **Run small contests locally** (why not, use real devices to stream)
- **Clone the repo and give back with PRs and challenges**
- **Run yearly contests at larger events**
- **Continue publishing writeups**

black hat USA 2021

August 4-5, 2021

ARSENAL

Thanks for tuning in! Questions?

***** REFERENCES *****

Website: https://cts.ninja

GitHub Repo: https://github.com/capturethesignal

Twitter: @SignalCapture