

# RF Swift: a swifty toolbox for all wireless assessments

By Sébastien Dudek



11



# Founder of Penthertz

- Sébastien Dudek (@FIUxluS)
- CEO of Penthertz
  - Founded during COVID in 2020
  - Specialized in Wireless communications security
- > 10 years of experience in Software & Hardware security
  - Security researcher
  - Pentester & Red Team
  - Vulnerability researcher

**Perfect mix to make  
Penthertz!**

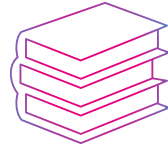


# Main activities



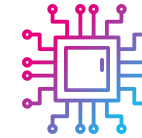
## Security assessments

- Wireless communications (RFID, Wi-Fi, Mobile communications, Bluetooth, etc.)
- Embedded devices
- Backend servers
- Red Team



## Trainings

- Software-Defined Radio Hacking
- Wi-Fi Red teaming
- RFID Hacking
- Mobile attacks (2G/3G/4G/5G), and more...



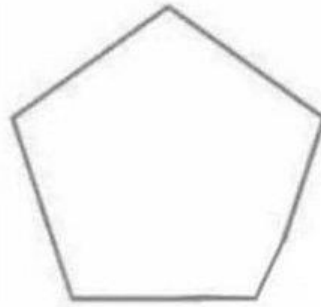
## Hardware security

- Firmware extraction
- Chip off
- Secrets extraction
- Library's analysis
- Vulnerability hunting

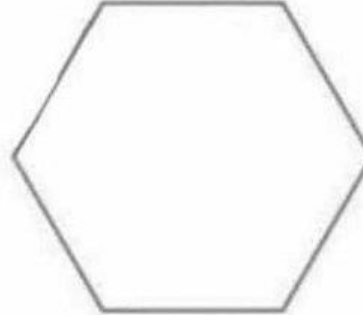
# RF Pentester 010: Having a good setup



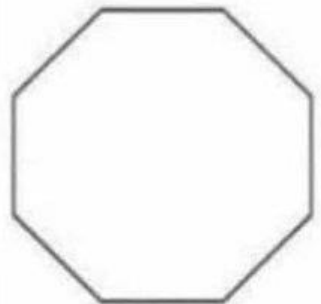
# A real lab -> follow the geometry...



Pentagon



Hexagon

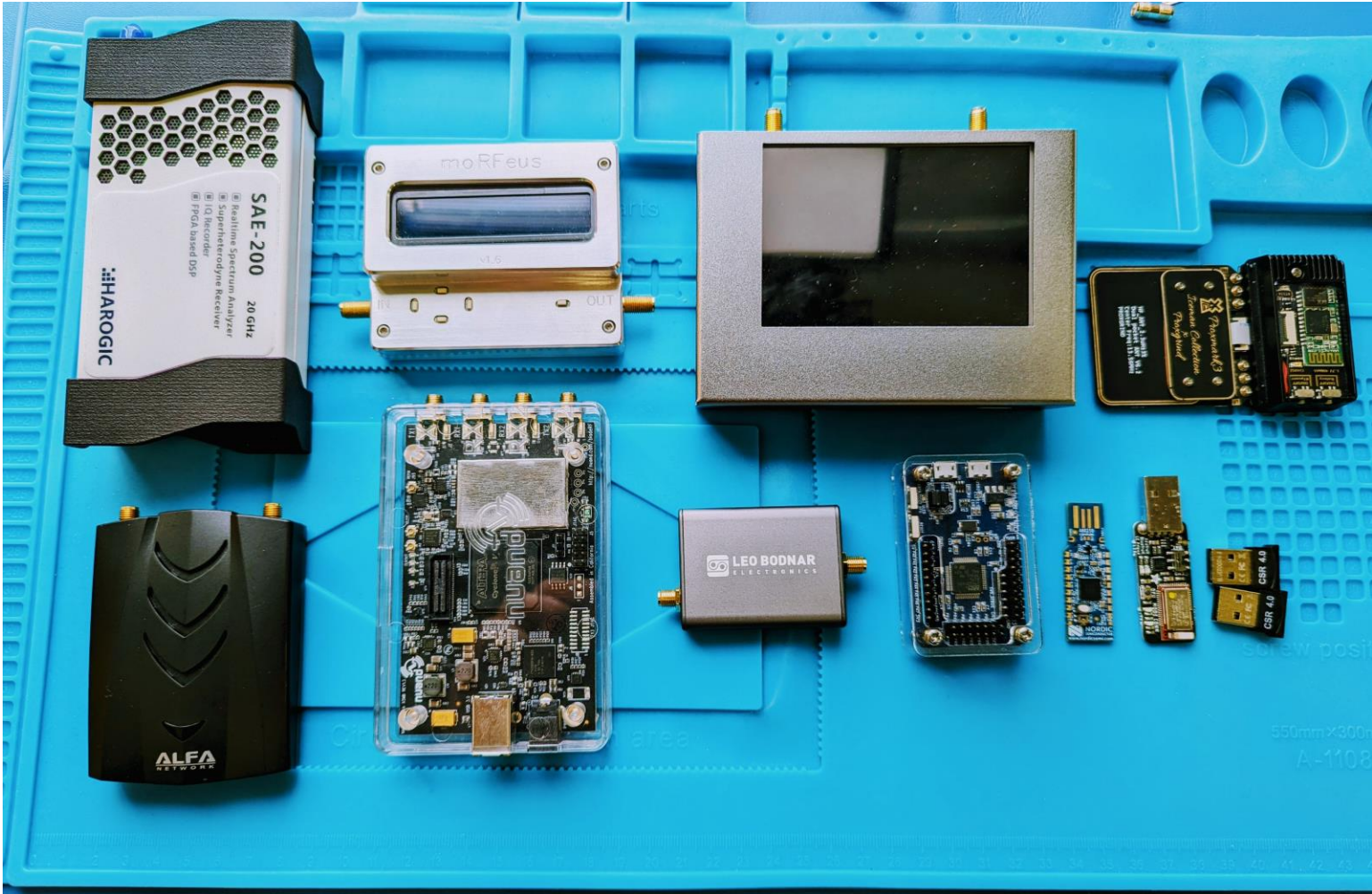


Octagon



**Moneygone**





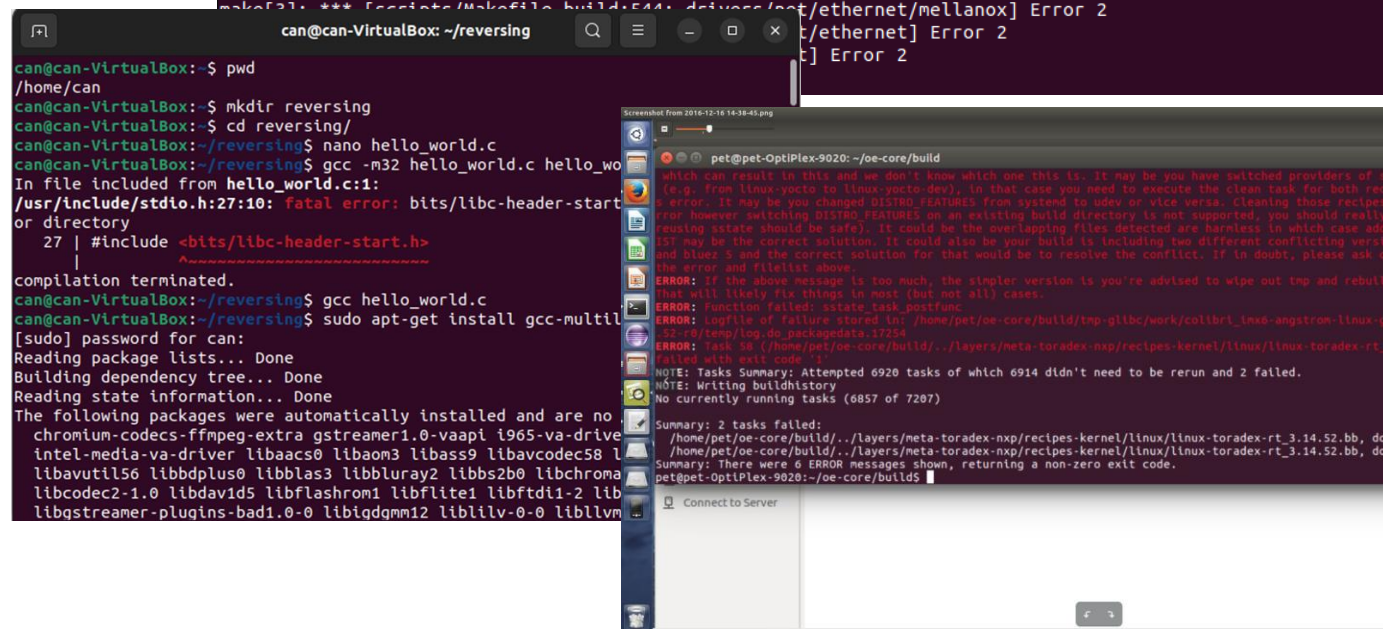
# Software setup

- We need all required pentests tools for different context:
  - Wi-Fi
  - RFID
  - Bluetooth Classic & LE 4/5
  - Telecom
  - And even exotic communications
- In addition: report generator, common network tools, web tools, etc.
- But: takes at least 1-5 days to setup properly (depending on number of tools)

# Compile your tools

- Need to deal with:
  - Compilation issues
  - Dependencies
  - Collisions/conflicts
- A good setup can take a day to a week depending on needed tools
- Time is running
- Not good when rushing on an assessment...

```
CC [M] drivers/net/ethernet/mellanox/mlx5/core/dev.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/wq.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/lib/gid.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/diag/fs_tracepoint.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/diag/fw_tracer.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_main.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_common.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_fs.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_ethtool.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_tx.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_rx.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_dim.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_txrx.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_xdp.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_stats.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_selftest.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en/port.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_arfs.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_fs_ethtool.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_dcbnl.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_port_buffer.o
CC [M] drivers/net/ethernet/mellanox/mlx5/core/en_rep.o
gcc: fatal error: Killed signal terminated program cc1
compilation terminated.
make[5]: *** [scripts/Makefile.build:304: drivers/net/ethernet/mellanox/mlx5/core/en_rep.o] Error 1
make[5]: *** Deleting file 'drivers/net/ethernet/mellanox/mlx5/core/en_rep.o'
make[4]: *** [scripts/Makefile.build:544: drivers/net/ethernet/mellanox/mlx5/core] Error 2
make[3]: *** [scripts/Makefile.build:544: drivers/net/ethernet/mellanox] Error 2
make[2]: *** [scripts/Makefile.build:544: drivers/net/ethernet] Error 2
make[1]: *** [scripts/Makefile.build:544: drivers] Error 2
```





# Alternative distributions

- Existing alternative distributions:
  - Kali: packages for Wi-Fi, Bluetooth, RFID, SDR and many other pentest tools
  - Pentoo: Like Kali with extra GNU Radio tools and modules, SDR tools as well (<https://github.com/pentoo/pentoo-overlay/tree/master/net-wireless>)
  - Dragon OS: Really focusing on radio tools and much more complete than other distributions
  - Others
- **But nothing specific to RF & Hardware security with hardware accesses (USB, mPCIe, GPU, sound, etc.)**



# Alternative distributions (2)

- **Pros:**

- Packages as much tools as possible --> reducing installation time
  - Tools not yet package can be installed after
- Less troubleshooting during our setup --> tools are ready to be used
- Perfect for less experienced people

- **Cons:**

- Need to reinstall the computer with the specialized distribution
  - And also to complete it with missing tools
- Dependencies issues with new installed tools --> breaking the setup
- No scalability

# Alternative distributions

- Existing alternative distributions:
  - Kali: packages everything you need for pentesting, including SDR, SD, and many other pentesting tools.
  - Pentoo: Like Kali with extra tools for wireless modules, SDR tools as well. (<https://github.com/pentestmonkey/pentoo-overlay/tree/master>)
  - Dragon OS: Really good for SDR tools and more complete toolsets for wireless communications.
  - Others
- But nothing specific to RF & Hardware security with hardware accesses (USB, mPCIe, GPU, sound, etc.)**



# Breaking the setup

- **Need to reinstall everything! Sometimes until 5am right before a pentest...**
- **You morning starts like that:**





## Breaking the setup (2)

- **How the client sees you during the assessment:**



# Each mission is different

- A mission needs a dedicated container:
  - More reproducibility + scalability
  - Avoid mixing "client 1" traces with "client 2"
  - You can mess inside a container -> destruct it there after





# Let me introduce you RF Swift!



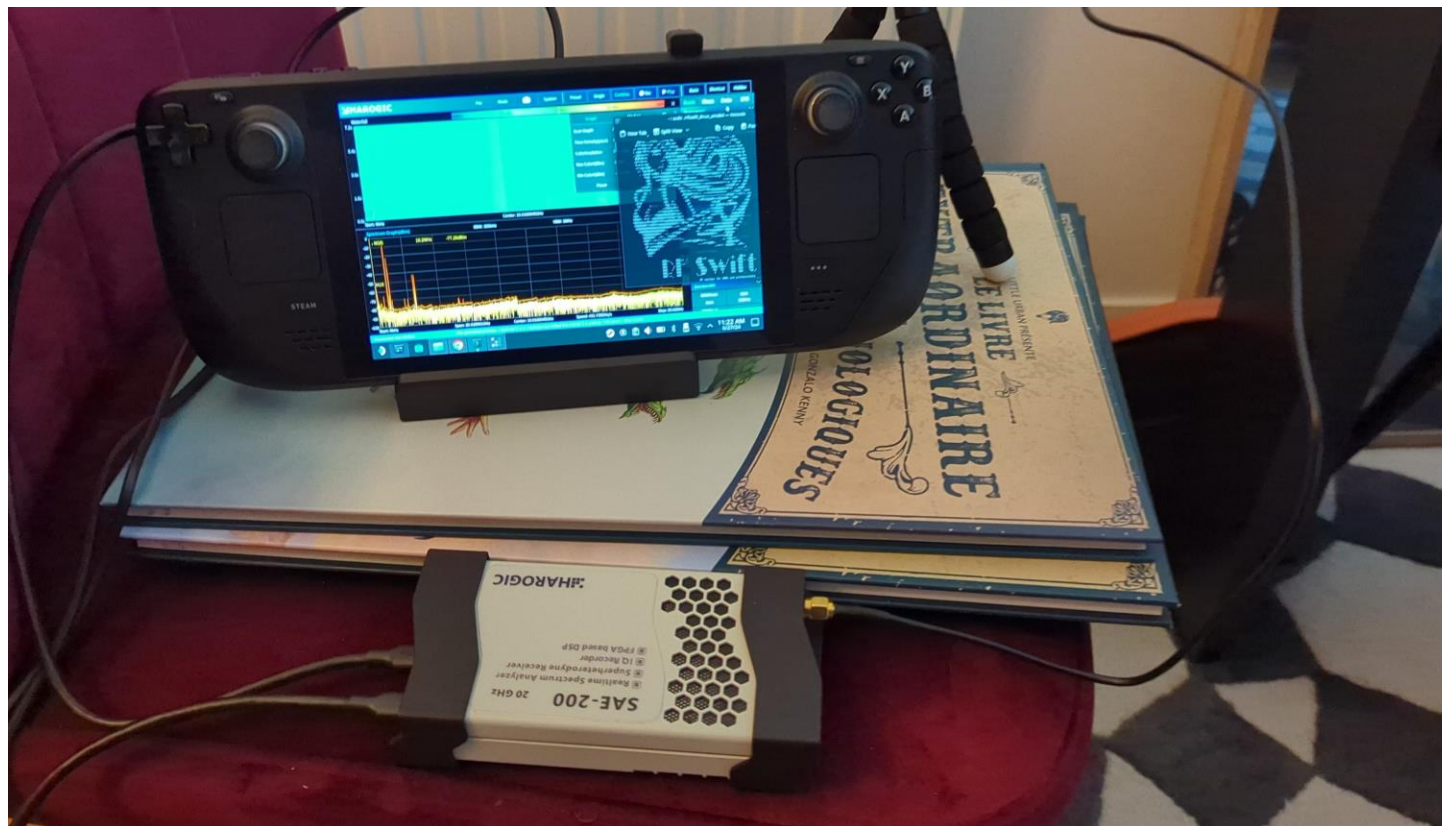
# What is it?

- Tool made in Go --> Instrumenting Docker + host
  - Inspired from Exegol project ;)
- Docker files "recipes"
- Registry with built images
- Scripts for automating installations of various tools
- Supported and tested architectures: x86\_64, ARM64, and RISC-V 64
- Supported and tested OSes: Linux and Windows

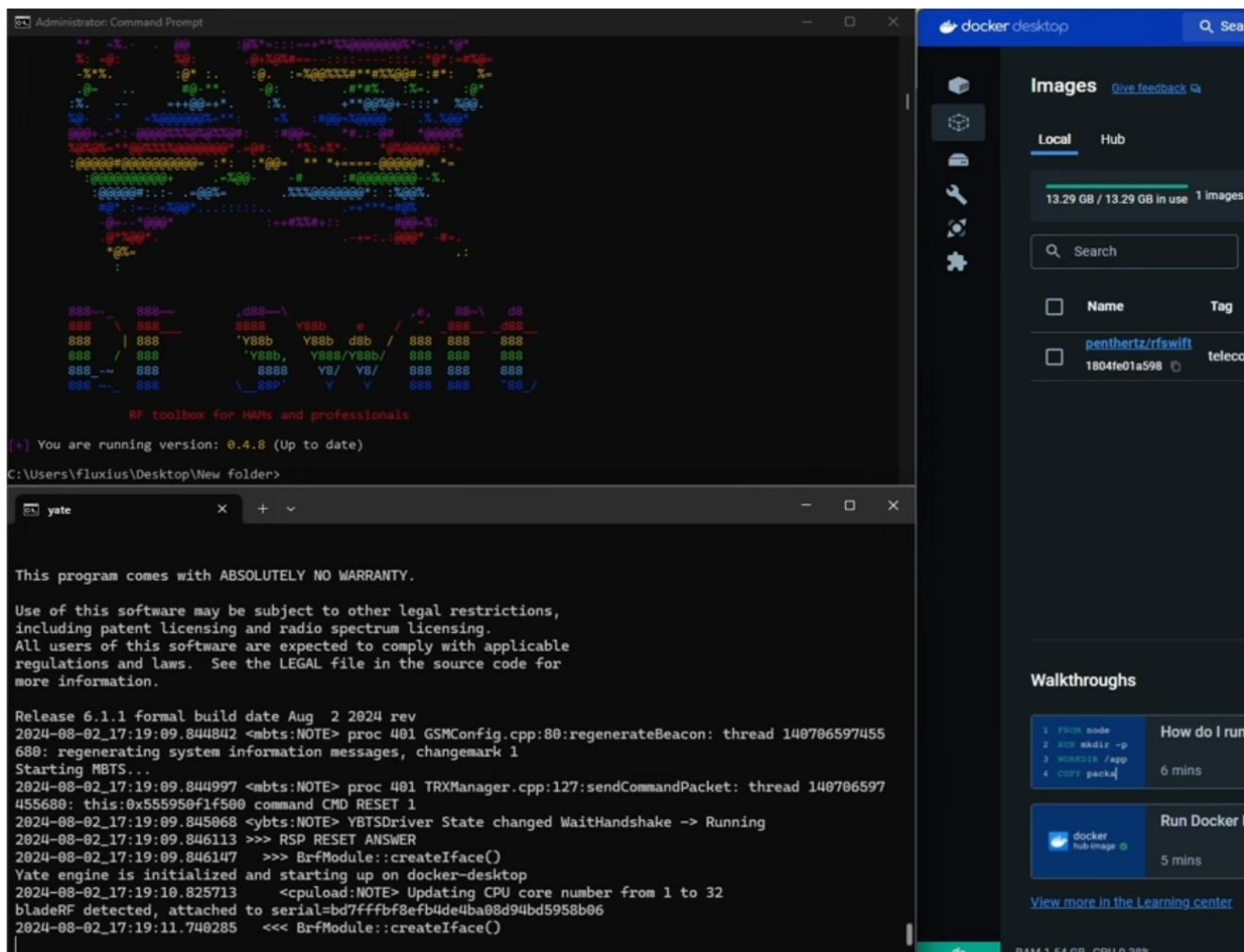




# Assessments on a Steam Deck

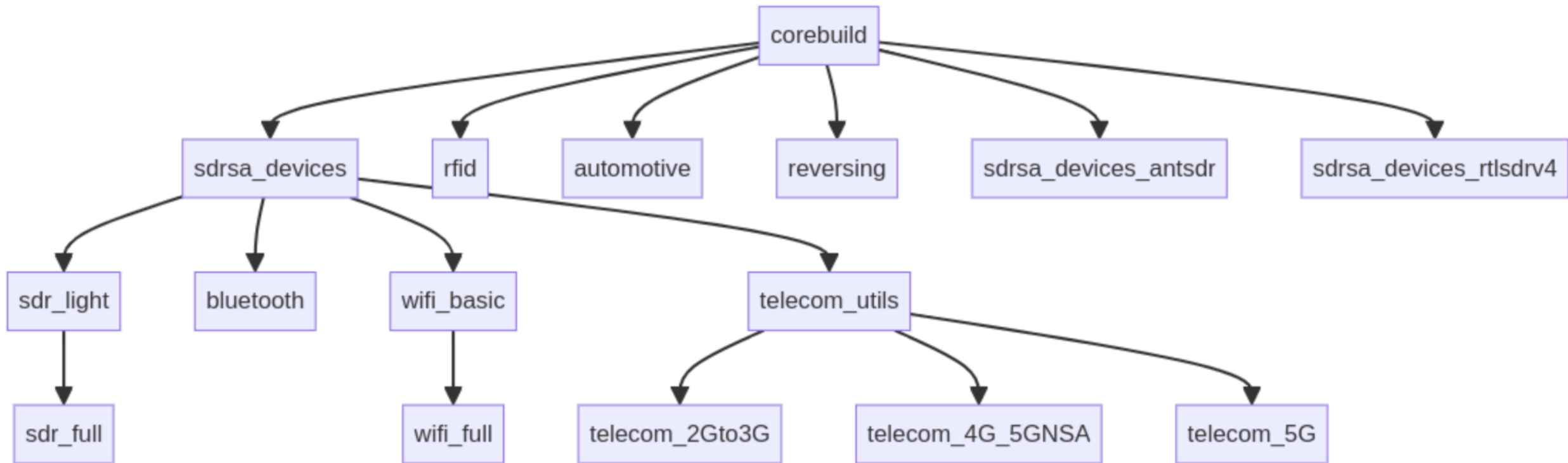


# Windows GPRS stations (in few minutes)



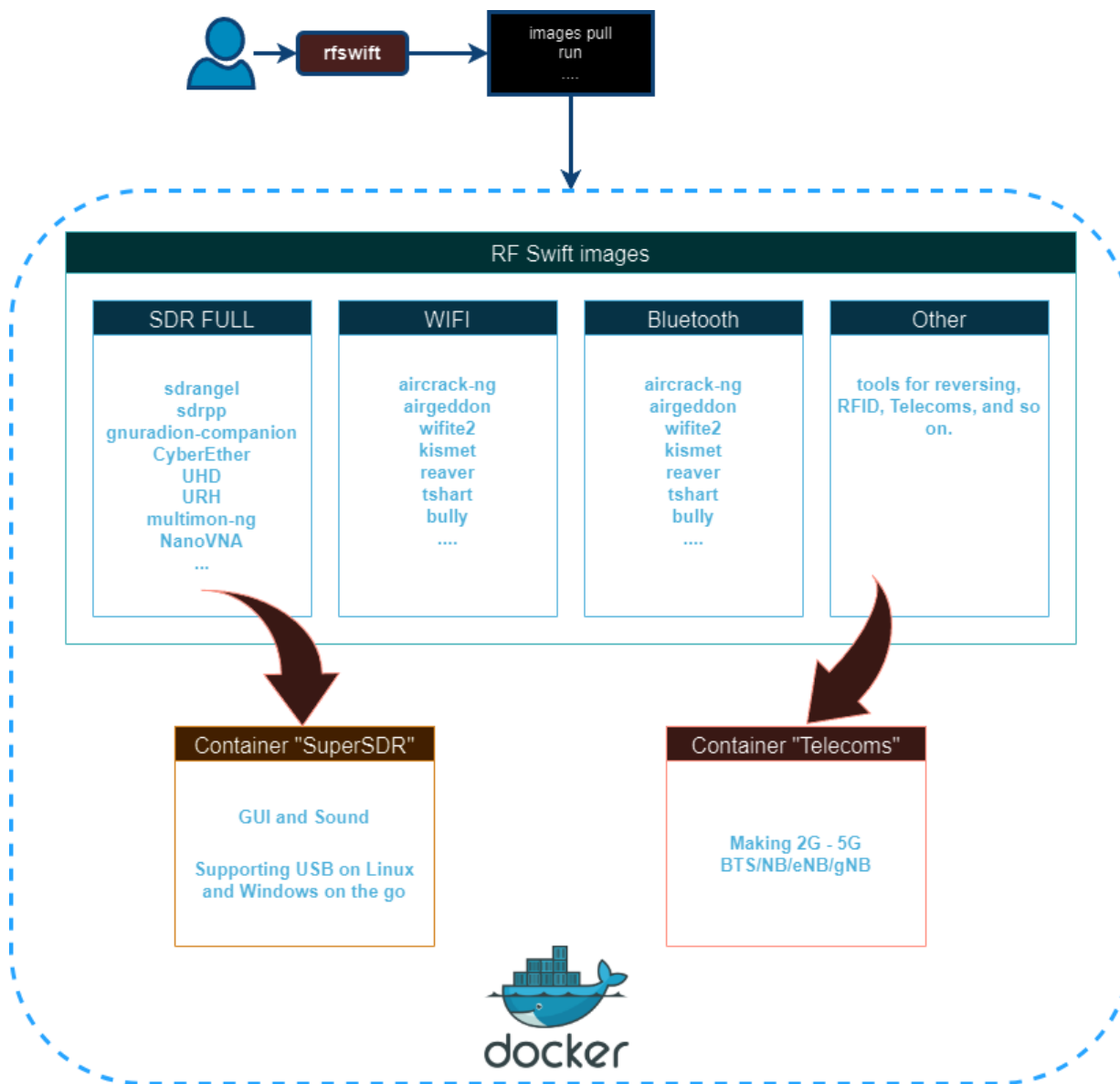
# Images' hierarchy

- Following Docker images layers concept: reuse of layers -> speed and space saving



# Architecture

- Each created container has tools included in dedicated images
- Each container represent a "mission"
  - Perfect for assessments separation: client1 and client2 are not in the same space
  - Messing with one container -> throw it and run a new container!





# Key commands

- Impatient to make your images? Pull one from our registry:

```
$ sudo ./rfswift images pull -i <reference> -t <tag name to apply>
```

- Run a container:

```
$ sudo ./rfswift run -i <image name> -n <container name> [-e <command>]  
[-b <extra volumes>]
```

- Shoot! I kill my terminal!! shhN00 problem:

```
$ sudo ./rfswift exec [-c <container name>] -e <command>
```

- Let's discover what's inside!

The background features a color gradient from magenta on the left to blue on the right. Overlaid on this are two sets of concentric circles. The circles on the left are magenta and larger, while the circles on the right are blue and smaller.

**Demo time!**



# Conclusion

# To conclude

- You can travel and assess devices safely with RF Swift
  - e.g. -> my computer that was just erased and used for traveling
- Keep you setup light based on your own "recipes"
- RF Swift is 1y old -> still a lot to do!
- Need also contributors:
  - Documentation: <https://rfswift.io/>
  - Go binary for instrumentation and user experience
- Our discord: <https://discord.com/invite/NS3HayKrpA>





# Thank You

Please contact us:

✉ [contact@penthertz.com](mailto:contact@penthertz.com)

☎ +33 1 73 13 82 77

🌐 [penthertz.com](http://penthertz.com)

Watch us on

