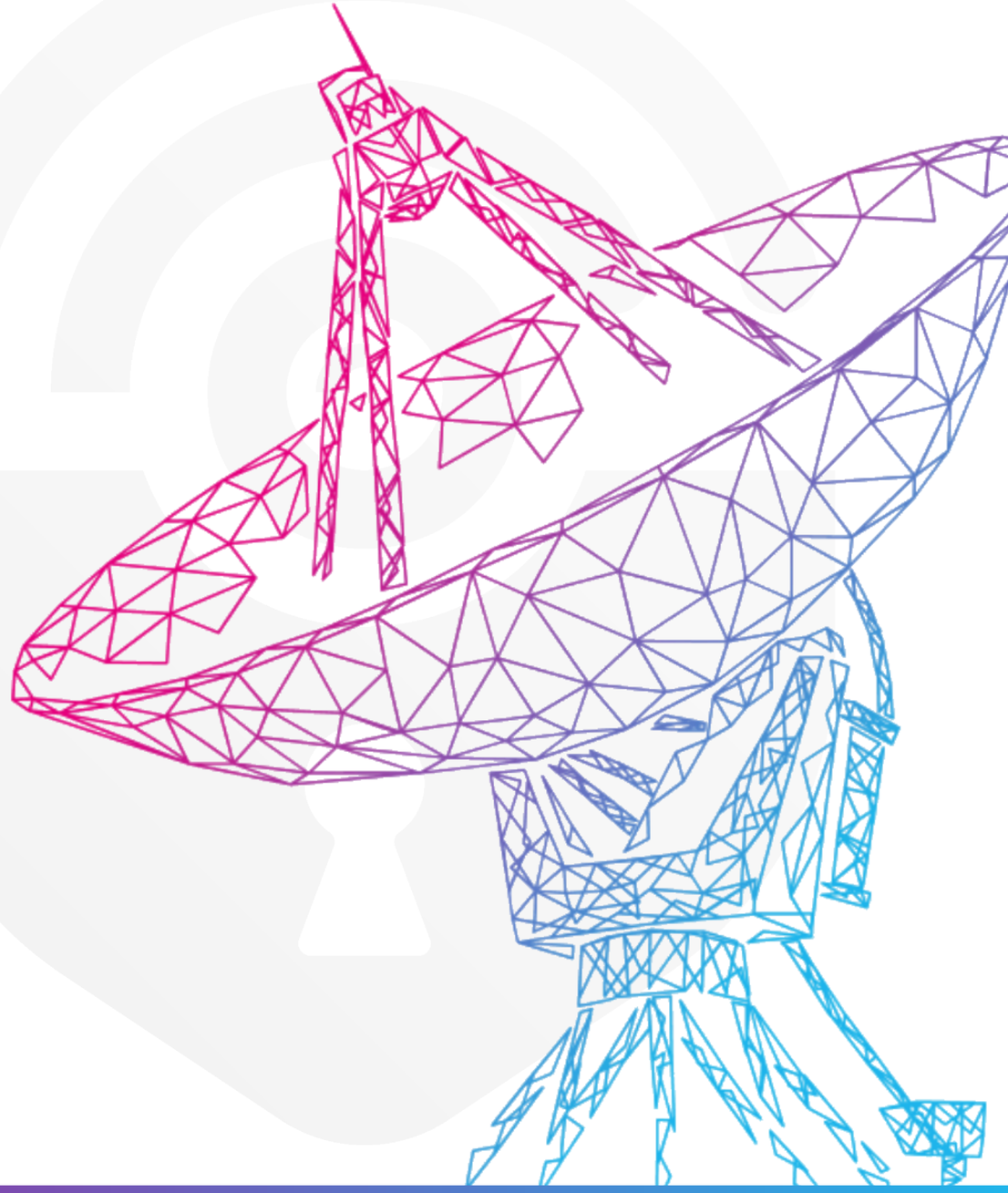
QUBES OS SUMMIT 2026, BERLIN

RF Swift on Qubes OS

Disposable, compartmentalized RF hacking labs

Sébastien Dudek | Founder, Penthertz



ABOUT ME

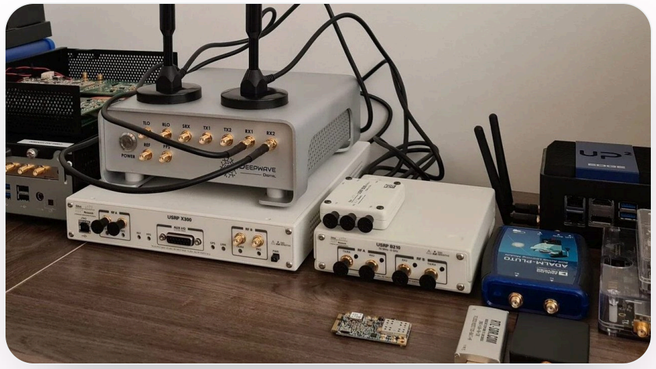
whoami



- **Sébastien Dudek**, `@FlUxIuS`
- Founder of **Penthertz**: wireless & hardware security
- Research: 5G, Open RAN, baseband, PLC, EV charging, and exotic radio...
- Author of **RF Swift**
- Qubes OS on a few laptops. Yes, even for RF.

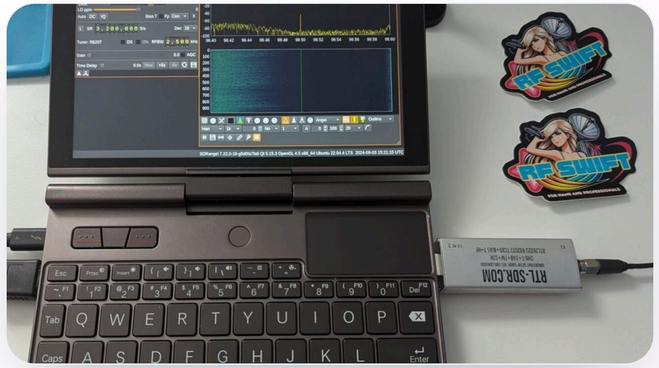
What we do

Pentest & Red Team



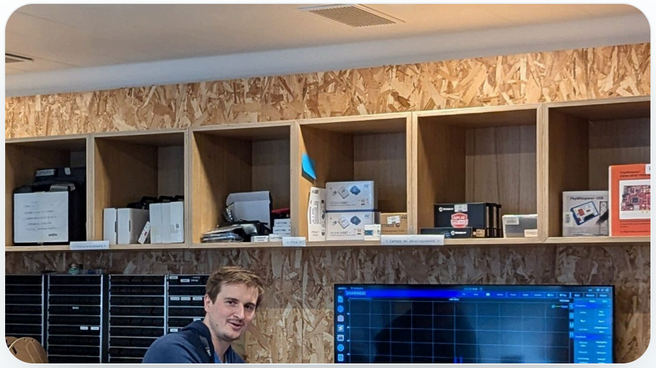
- Mobile 2G to 5G, Wi-Fi, BLE
- RFID, LoRa, cars, satellites
- CRA & RED readiness

Research & tooling



- Firmware, glitching, chip-off
- Vuln research up to CVE
- Open source: **RF Swift**

Trainings



- SDR, mobile, RFID, Wi-Fi
- Live and on-demand (Hackademy)
- Hardware kit included

Trained and spoke at Hardwear.io, Black Hat, Troopers, HITB, NorthSec, SSTIC, FOSDEM

WHY IT EXISTS

RF Swift was born at 6am

- Every mission: Wi-Fi, RFID, BLE, SDR, cellular, exotic
- Hundreds of tools: **compile, deps, conflicts**
- **1 to 5 days** to set up a laptop
- One broken dependency: reinstall, **until 6am**
- Kali, Pentoo, DragonOS: great for **hobbyists**
- Not for **pro use**, **corporate laptops**, **Qubes OS**

> **Pull a toolbox, plug the radio, hack.** Minutes, not days.



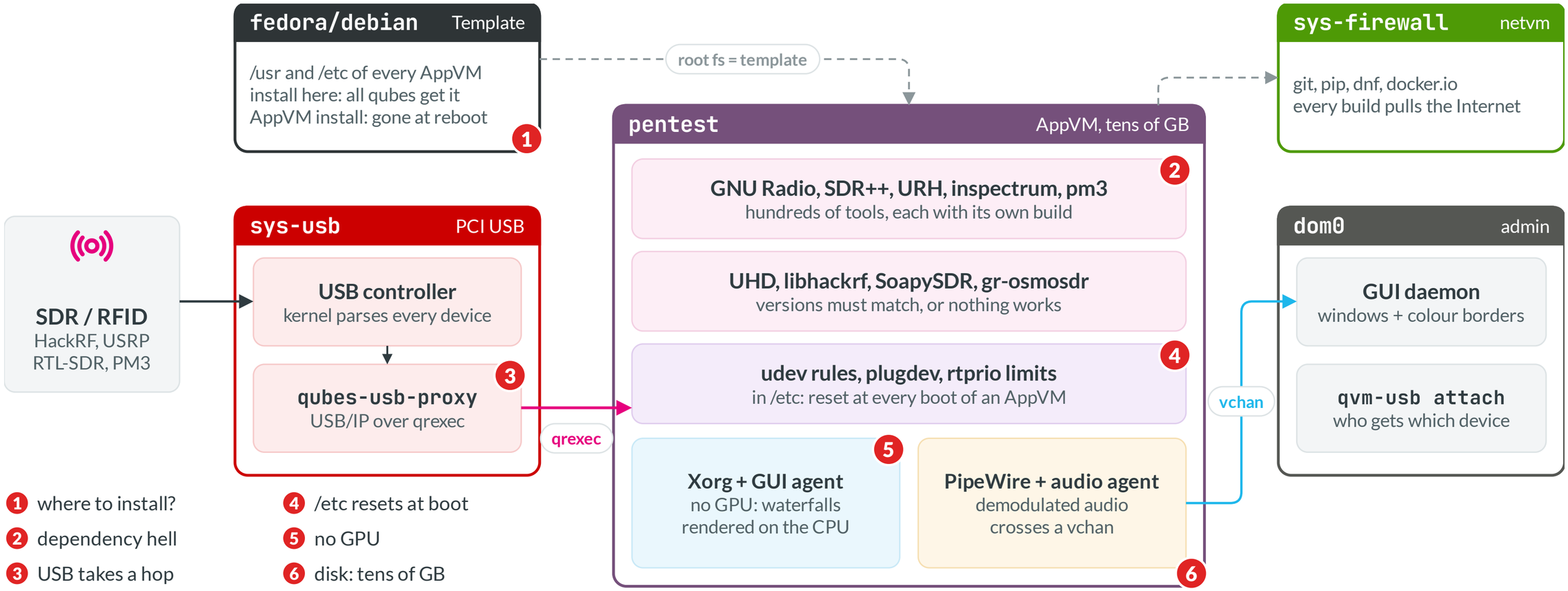
6am, setup still broken



9am, how I look meeting the client

WHY IT HURTS MORE HERE

On Qubes OS, every layer is a hop



1. where to install?
2. dependency hell
3. USB takes a hop
4. /etc resets at boot
5. no GPU
6. disk: tens of GB

Not impossible. Just **a lot of plumbing**, per qube, by hand.

SECTION 01

Enter RF Swift



01

RF SWIFT

What RF Swift offers

Your RF lab as a toolbox: **pull, plug, hack**.

Toolboxes

Hundreds of RF & HW tools, **prebuilt images**

Minutes, not days

Pull, create, hack. Broken? **Recreate.**

Hardware plumbing

USB, serial, X11, audio, GPU, **realtime**

One lab per mission

Client A **never meets** client B

Runs anywhere

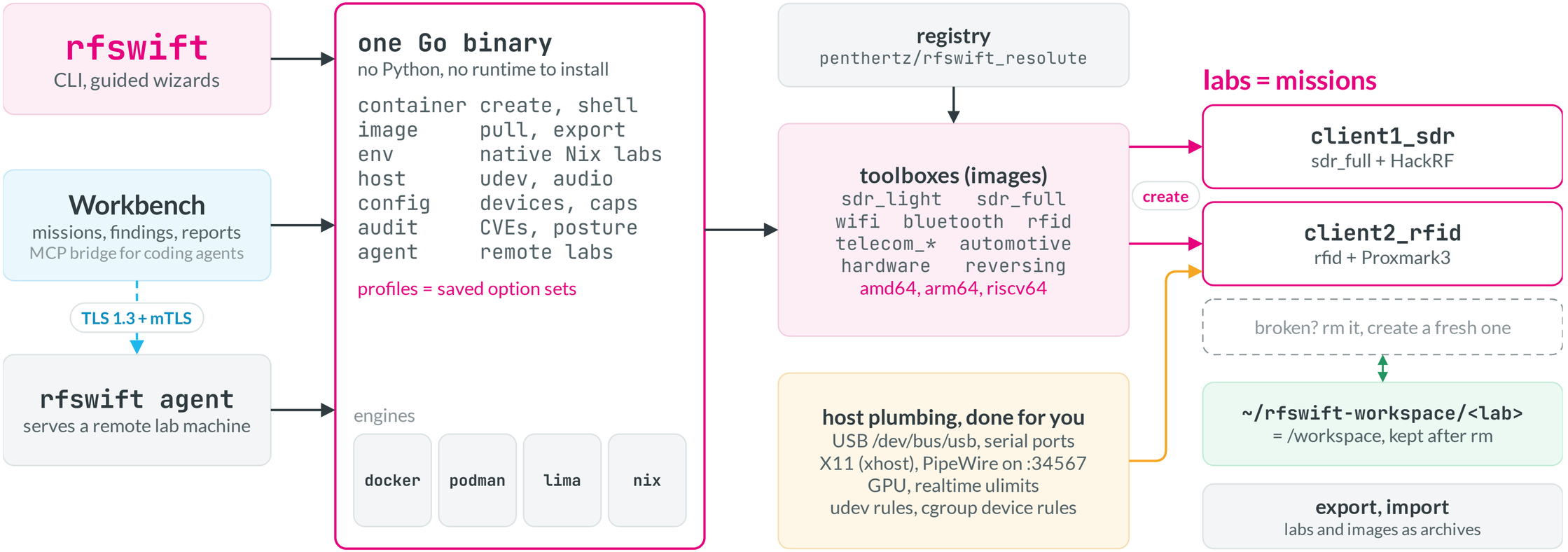
Linux, Windows, macOS. **amd64, arm64, riscv64**

Assessment kit

Workbench, recordings, findings, **reports, audits**

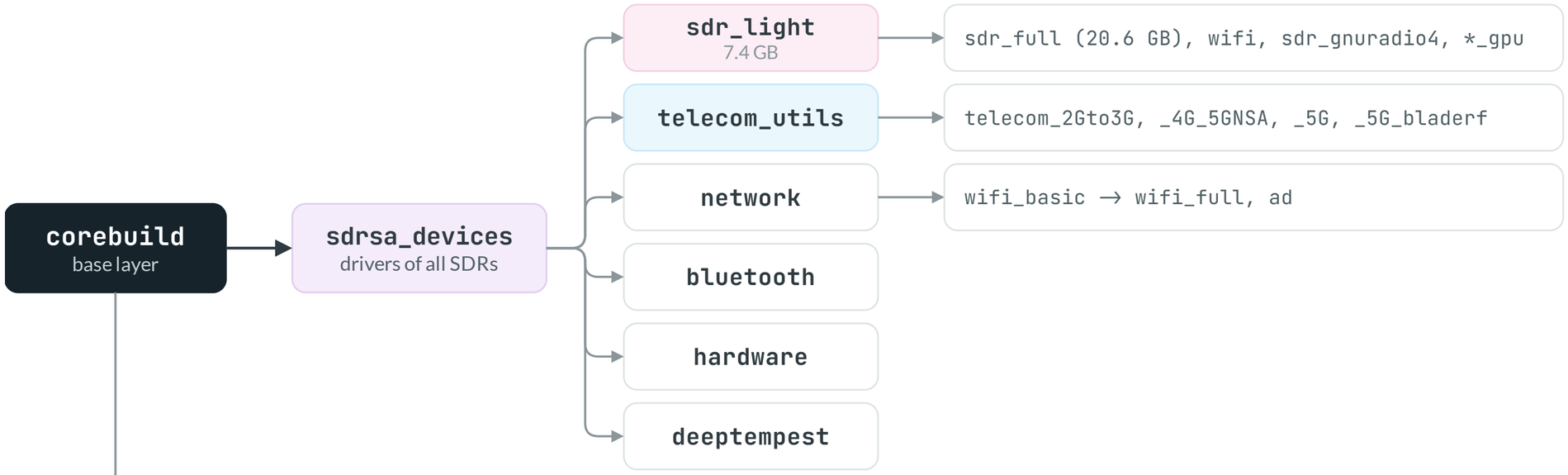
Open source, GPLv3. Docker, Podman, Lima or Nix underneath.

How it works: one binary, many labs



Same labs from the **CLI**, the **Workbench** or a **remote agent**.

Toolboxes are layered images



straight from corebuild
rfid, automotive, reversing, osint, android
sdrsa_devices_antsdr, sdrsa_devices_rtlsdrv4

layers are shared
pull sdr_full after sdr_light: only the difference comes down
one image store per qube: share it via the template,
or via a disposable template (see later)

Pick the toolbox for the job: **sdr_light** to start, **sdr_full** for everything.

From zero to waterfall

the lab lifecycle

```
$ rfswift image remote
$ rfswift image pull -i sdr_light
$ rfswift container create -i sdr_light \
    -n clientA_sdr --realtime
$ rfswift container shell -c clientA_sdr
$ rfswift container last
$ rfswift container rm -c clientA_sdr
# captures stay in ~/rfswift-workspace/
# no flags? rfswift container create = wizard
```

inside clientA_sdr

```
$ hackrf_info
$ SoapySDRUtil --find
$ hackrf_transfer -r /workspace/fob.cs8 \
    -f 433920000 -s 2000000
$ inspectrum /workspace/fob.cs8
$ urh
$ sdrpp
$ rtl_433 -f 868M
```

RF SWIFT

Built for assessments

Profiles	`--profile sdr-full`, `rfid`, `headless`
Realtime	`--realtime`: rtprio 95, memlock
Recordings	`--record`: asciinema per session
Change later	`rfswift config`: devices, caps, ports
Offline	`rfswift image export` / `import`
Audit	`rfswift audit`: CVEs, lab posture
Health	`rfswift doctor`: what to fix
No containers	`--engine nix --lazy`

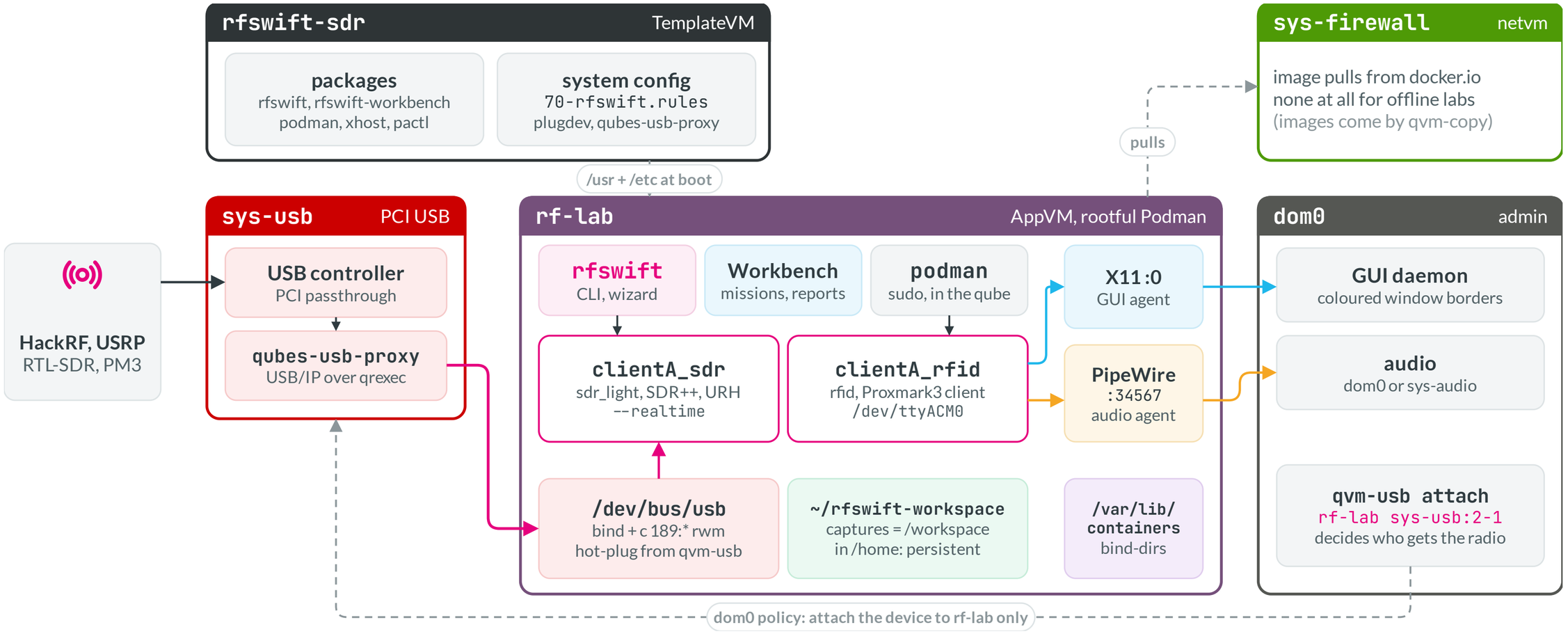
SECTION 02

RF Swift on Qubes OS

02

ARCHITECTURE

One template, one lab qube, many labs



Setup: three qubes, a few commands

dom0

```
$ qvm-clone fedora-42-xfce rfswift-sdr
$ qvm-create --class AppVM --template rfswift-sdr \
    --label purple rf-lab
$ qvm-volume resize rf-lab:private 90GB
$ qvm-prefs rf-lab vcpus 4
$ qvm-usb attach rf-lab sys-usb:2-1
```

rfswift-sdr (TemplateVM)

```
# netvm on for the install, off after
$ U=https://raw.githubusercontent.com/PentHertz/RF-Swift
$ curl -fsSL $U/refs/heads/main/get_rfswift.sh | sh
$ rfswift host udev        # /etc lives here
```

rf-lab (AppVM)

```
$ sudo rfswift image pull -i sdr_light
$ sudo rfswift container create -i sdr_light \
    -n clientA_sdr --realtime
$ sdrpp
```

> **rpm in the template**: every AppVM gets the `rfswift` binary, the rules and Podman.

REAL OUTPUT

Live from a Qubes 4.3 AppVM

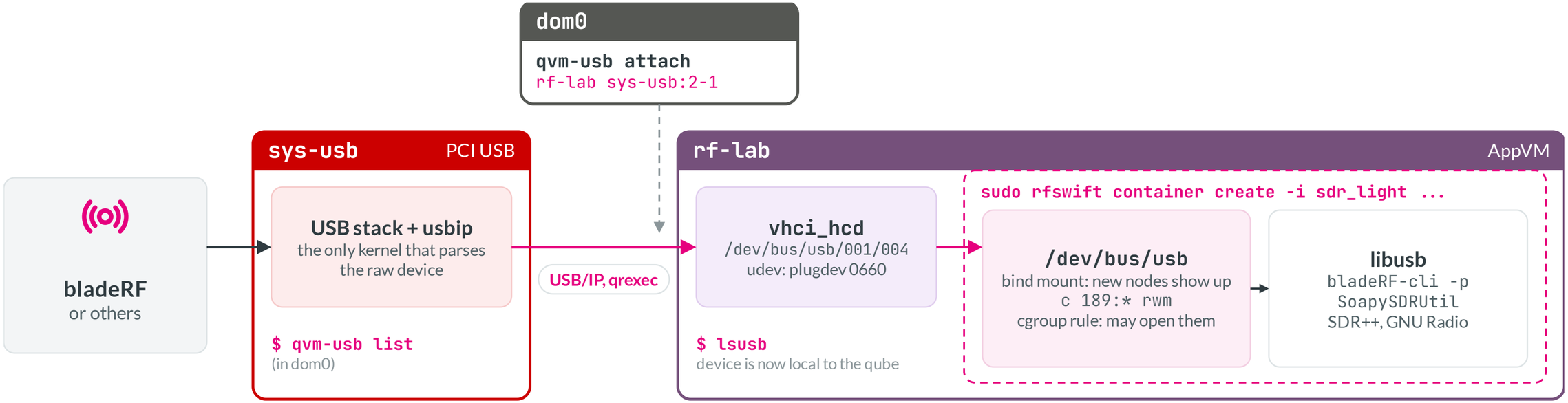
sudo rfswift doctor

```
$ sudo rfswift doctor
  ✓  Container engine     Podman (rootful)
  ✓  RF Swift images      4 image(s) available
  ✓  X11 display          DISPLAY=:0, socket present
  ✓  Audio system         PipeWire
  ✓  Audio TCP server     localhost:34567
  ✓  USB devices          /dev/bus/usb (2 buses)
  ✓  udev rules           70-rfswift.rules, plugdev
  ✓  Nix jail (--isolate) bubblewrap works
  13 passed  1 warning
```

sudo rfswift container create

```
$ sudo rfswift container create -i sdr_light \
    -n summit_lab --realtime
[i] Container engine: Podman (auto-detected)
non-network local connections being added to ACL
[i] PipeWire is already running.
[i] Realtime mode enabled: rtprio=95, memlock=unlimited
  Cgroup rules  c 189:* rwm,c 166:* rwm,c 188:* rwm
  ✓ Container 'summit_lab' started successfully
$ hackrf_info
libhackrf version: 2026.01.3 (0.9.2)
No HackRF boards found.   # not attached yet
$ ulimit -r
95
```

USB: from the dongle to libusb



rootless Podman
no cgroup rules = no hot-plug,
RF Swift 4.0.4 stops there
in a qube: sudo rfswift ...

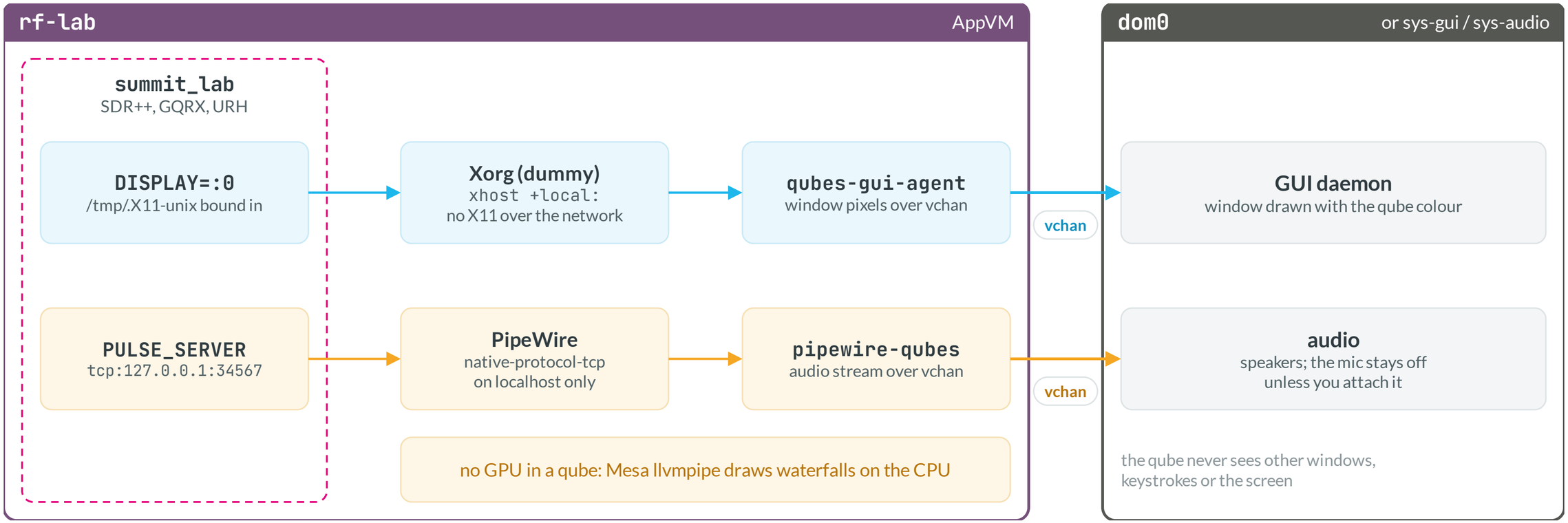
re-enumeration
firmware load (USRP B2xx, FX3)
shows up as a new device:
attach it again with qvm-usb

bandwidth
USB/IP + vchan cost CPU, high
sample rates may drop samples:
vCPUs, lower rate, or a PCI ctrl

Rootful Podman **inside** the qube: root there is not a privilege, **Xen is the wall**.

DISPLAY & SOUND

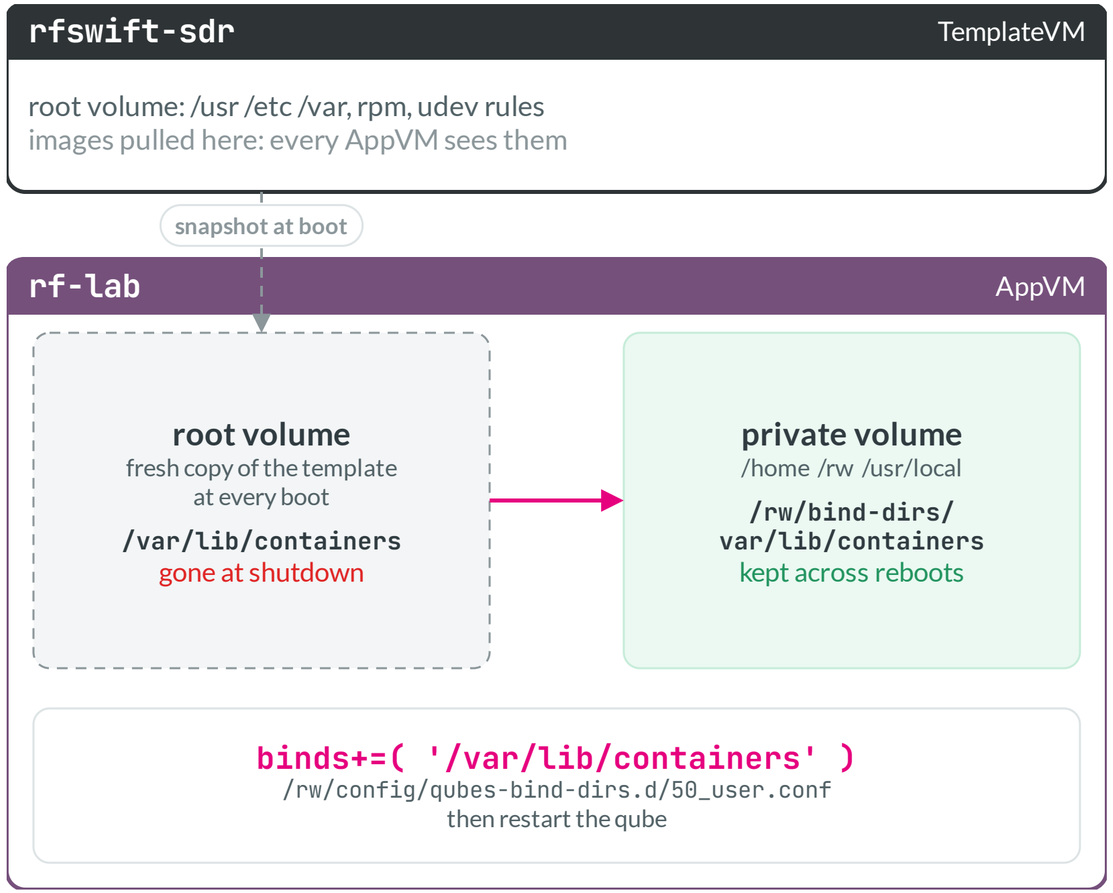
GUI and audio, still isolated



Waterfalls run on CPU: give rf-lab vCPUs. **VirtIO-GPU** talk at 15:50, same room!

STORAGE TRICK

Keep your images: /var/lib/containers



- **Template**: pull there, every AppVM gets the images
- **bind-dirs**: per qube, kept across reboots
- **Disposable template**: pull once, every disposable ready

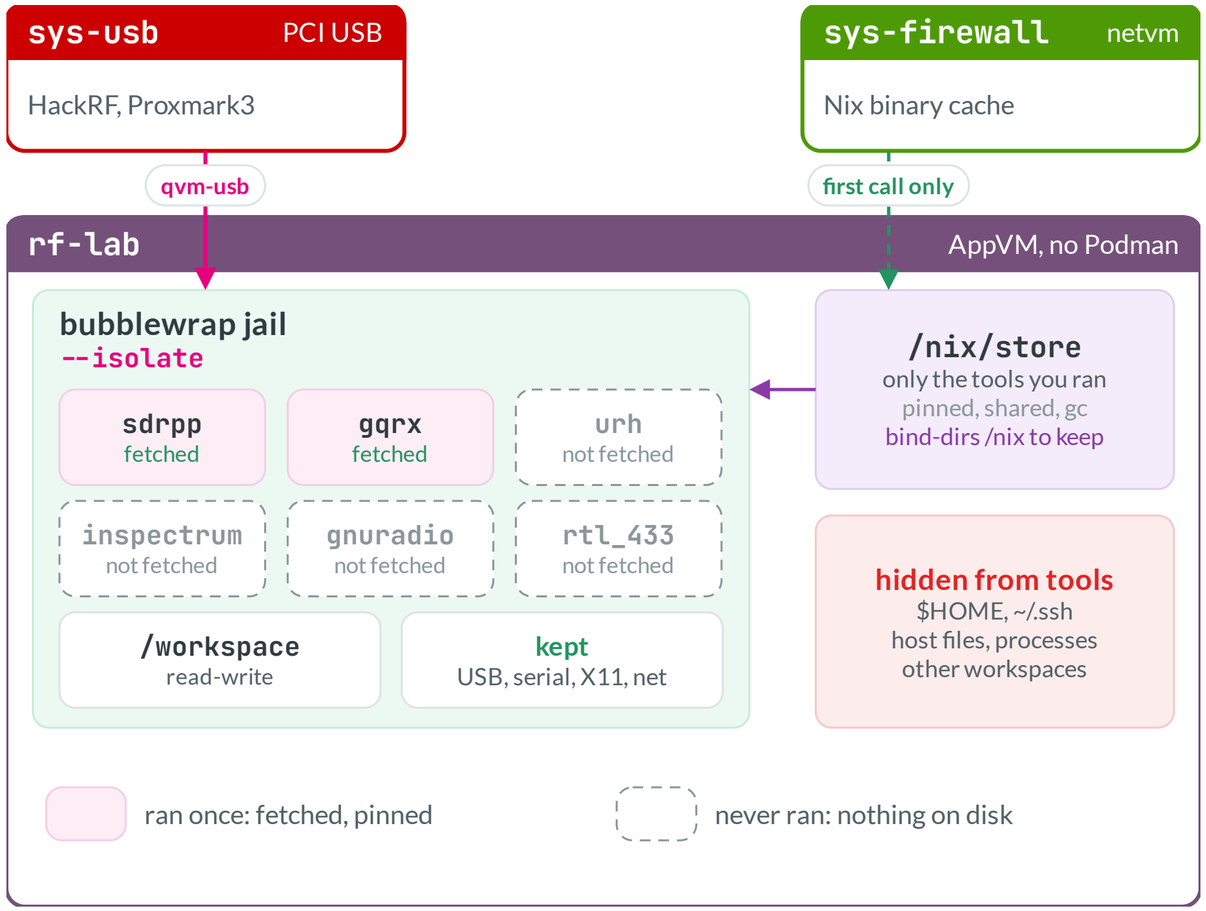
rf-lab (AppVM)

```
$ sudo mkdir -p /rw/config/qubes-bind-dirs.d \
    /rw/bind-dirs/var/lib/containers
$ echo "binds+=( '/var/lib/containers' )" | \
    sudo tee /rw/config/qubes-bind-dirs.d/50_user.conf
# restart the qube, pull once, keep forever
```

sudo: workspace in /root. Use `--workspace` in /home + SELinux `container_file_t`

NIX ENGINE

No containers: Nix, lazy and jailed



- **Nix engine**: native tools as your user, no Podman
- `--lazy`: each tool fetched **on first call**
- Run SDR++ only? **only SDR++ on disk**
- `--isolate`: **bubblewrap** jail, USB still works

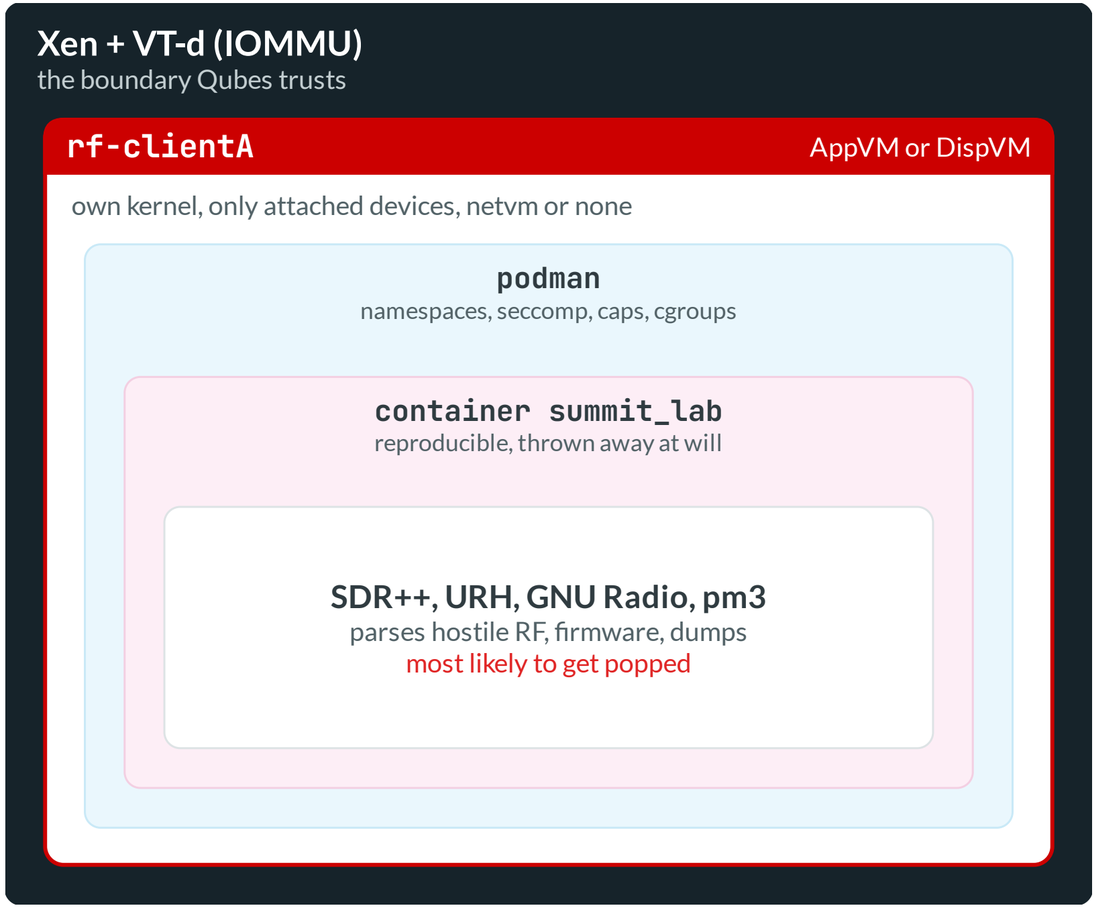
```
rf-lab (AppVM)

$ rfswift container create --engine nix \
    -i sdr_light -n clientA --lazy --isolate
$ sdrpp      # fetched now, pinned after
```

Tested on **Linux** and **macOS** only. On Qubes: not yet, help welcome!

SECURITY

Where the boundaries actually sit



A popped tool reaches

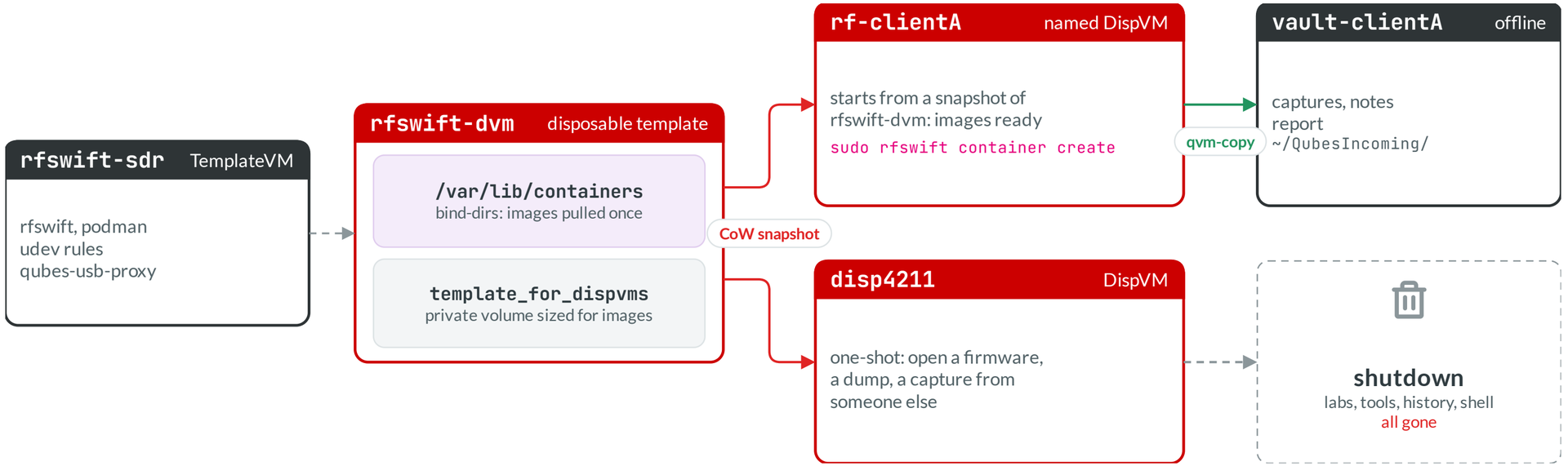
- its lab and **/workspace**
- **this** client's radio and files
- this qube's network, or **none**

It does not reach

- other clients, other labs
- dom0, your mail, your keys
- devices you did not attach

WORKFLOWS

Disposable RF labs

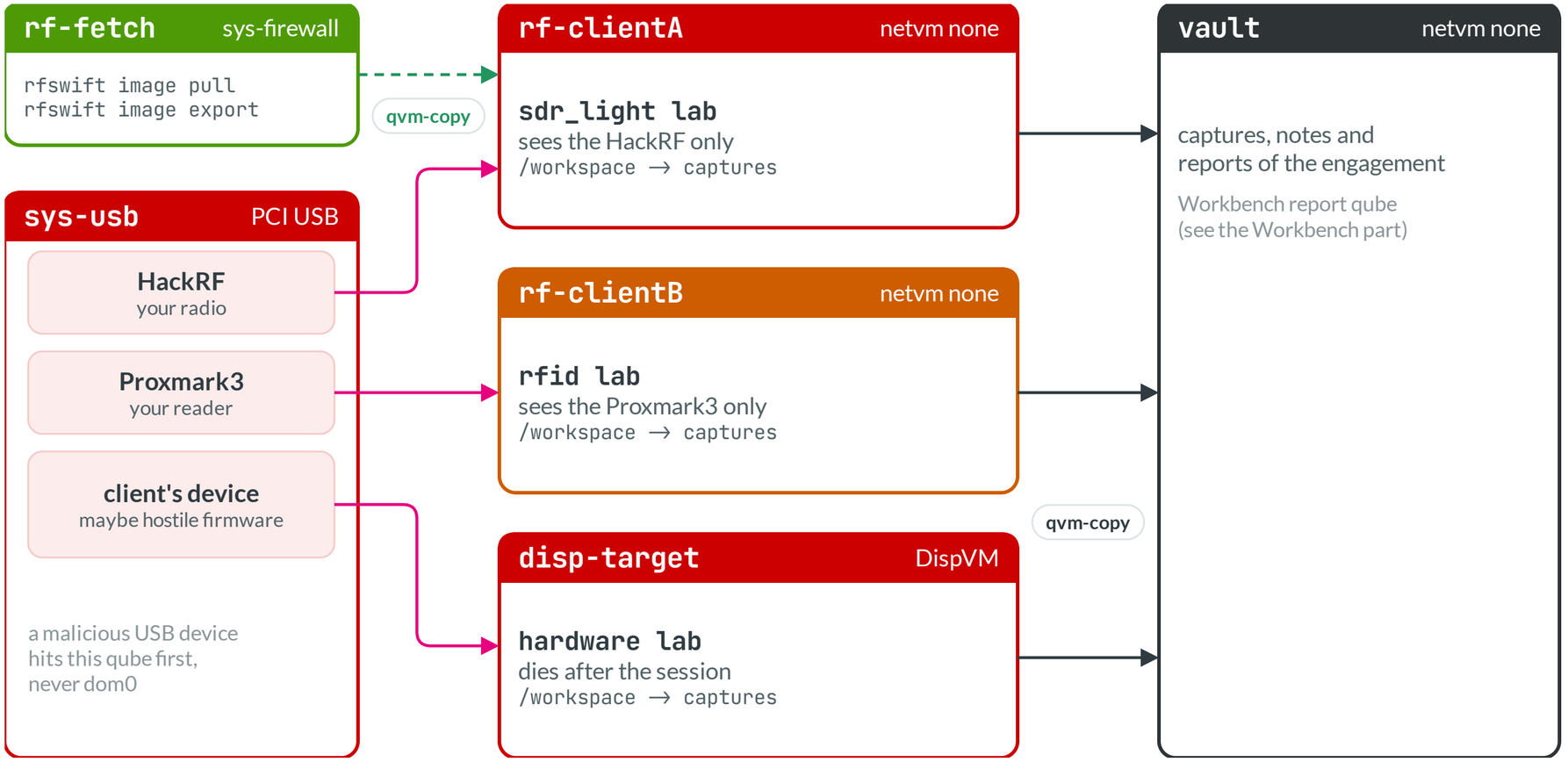


dom0

```
$ qvm-create --template rfswift-sdr --label red rfswift-dvm
$ qvm-prefs rfswift-dvm template_for_dispvms True
$ qvm-create -C DispVM --template rfswift-dvm --label red rf-clientA
```

WORKFLOWS

One engagement, one compartment



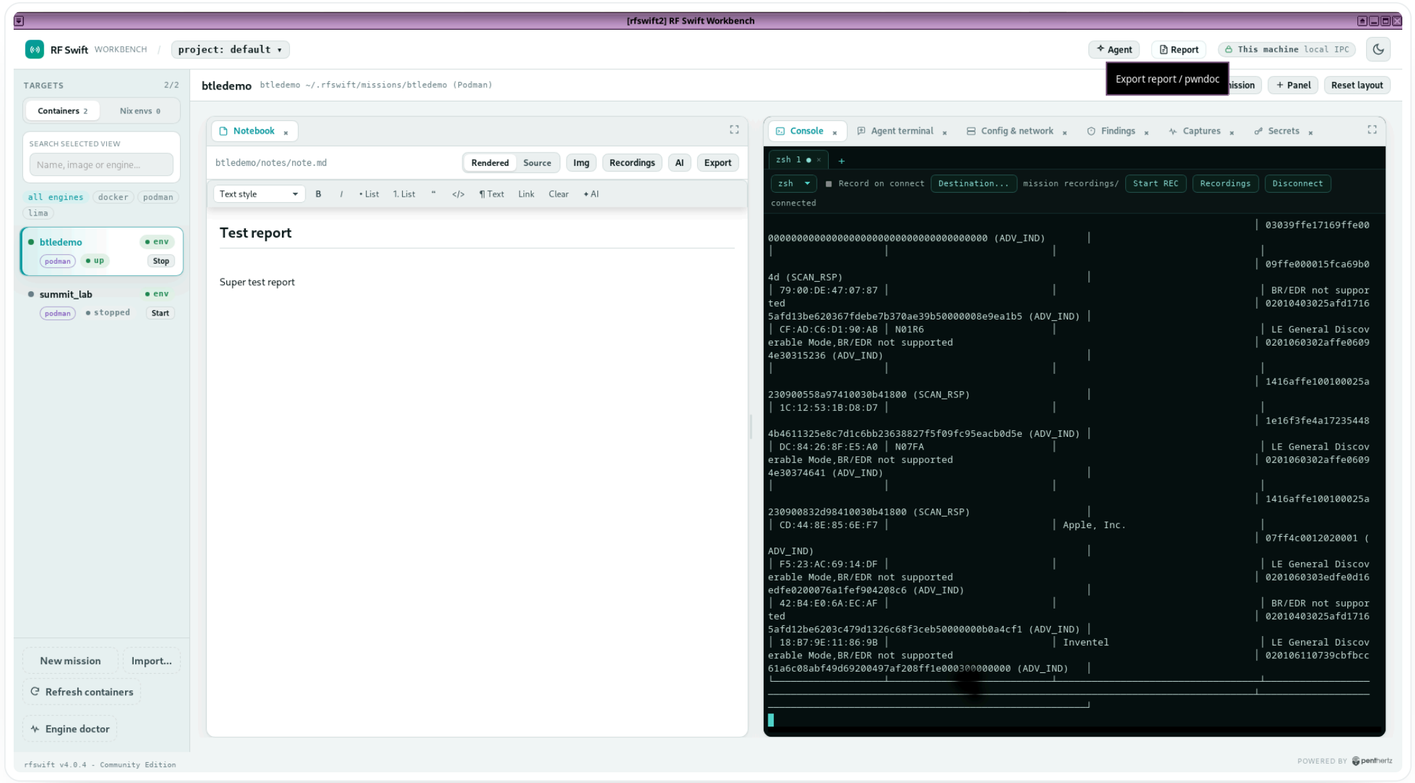
no cross-contamination
client A never shares a kernel, a disk or a radio with client B

no exfiltration path
offline labs: images in, evidence out, by qvm-copy only

hostile hardware contained
the target device lands in sys-usb or a disposable

WORKBENCH

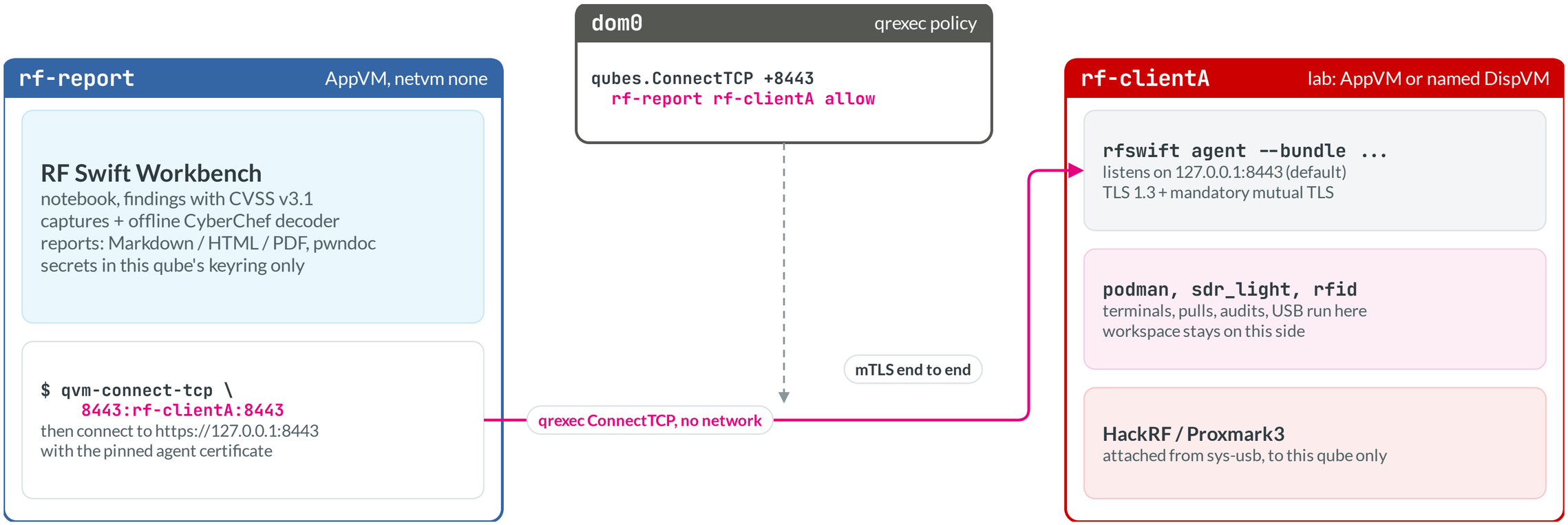
RF Swift Workbench: the mission notebook



Live in **rfswift2**: the btledemo lab scanning BLE

- One mission = one lab
- **Console**, recorded
- **Notebook**: Markdown, screenshots
- **Findings**: CVSS v3.1, pwndoc
- **Captures**: IQ, Flipper, pm3
- **Secrets**: OS vault only
- **Report**: MD, HTML, PDF

Workbench on Qubes: report qube + lab qube



Simple: Workbench **in rf-lab**. Split: notes and secrets **never leave rf-report**.

LIVE

Demo

attach
HackRF to a new **disposable**

→

lab
`sudo rfswift container create`

→

capture
SDR++, `hackrf_transfer`

→

analyze
URH, inspectrum

→

report
Workbench finding, `qvm-copy`

→

shutdown
nothing left behind

TAKEAWAYS

What to remember

1. Qubes is not scary for RF: **one template, one lab qube**, RF Swift does the plumbing
2. Toolboxes in **minutes**, one lab per mission, **disposables** per engagement
3. Security for free: hostile devices hit **sys-usb**, clients **never meet**
4. Honest limit: huge IQ streams over USB? **native Linux wins.** The rest: go.
5. Try it, break it, contribute: **rfswift.io**, GPLv3
6. Stay tuned for next RF Swift v4.1.x including Docker and Nix for QubeOS + some more improvements

penthertz

Thank you

PLEASE CONTACT US

contact@penthertz.com

rfswift.io

penthertz.com