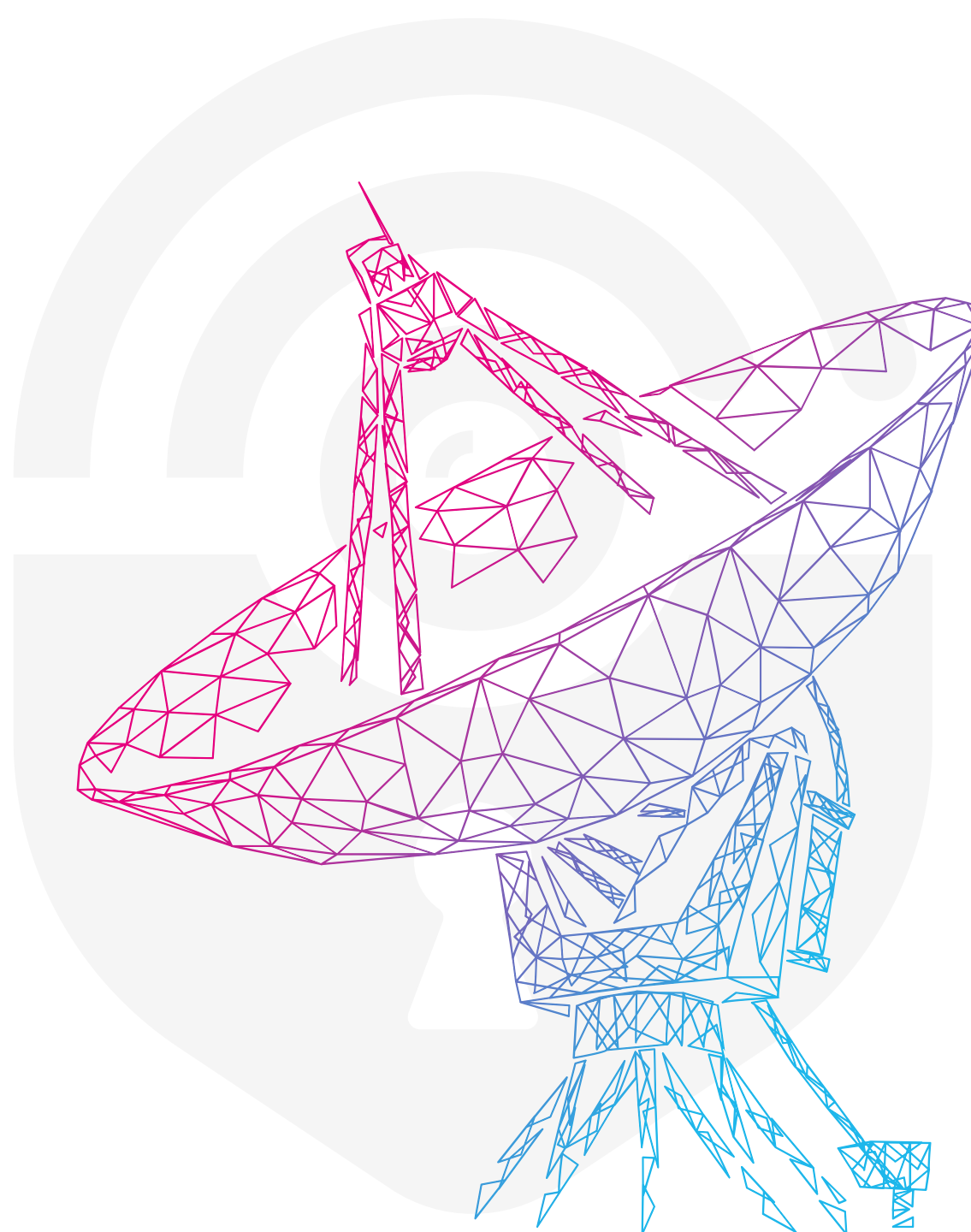


Bug hunting in 5G-NR devices and networks

No Hat 2021

By Sébastien Dudek



Founder of Penthertz

- Sébastien Dudek ([@FIUxluS](#))
- Could be CTO → Chief Taxes Officer...
- > 10 years of experience in Software / Hardware security
 - Former Sogeti ESEC R&D as a researcher
 - And Synacktiv as a pentester + vulnerability researcher
- Specialized in Wireless communications
- Also, security researcher [@Trend Micro](#)
 - Industrial IoT
 - Mobile, and other RF/SDR things...

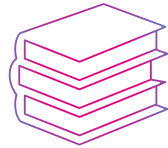


Main activities



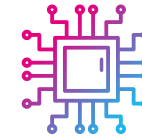
Security assessments

- Wireless communications (RFID, Wi-Fi, Mobile communication, 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...



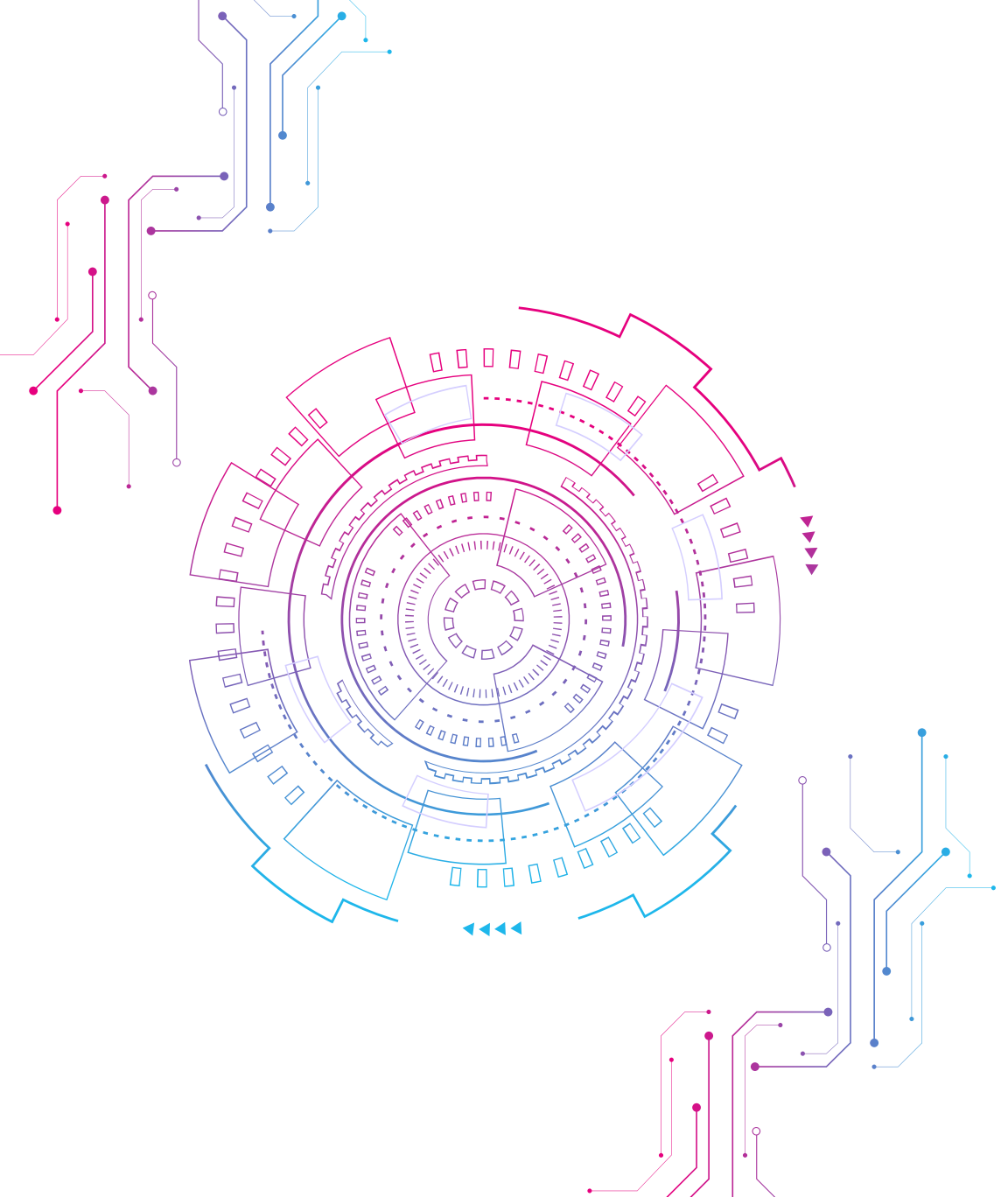
Hardware security

- Firmware extraction
- Chip-off
- Secrets extraction
- Libraries analysis
- Vulnerability hunting

Part of the SDR material

- Need to manage any type of transmission (2G-5G, Wi-Fi, Remotes, RFID, Bluetooth, ZigBee, RFID, etc).
- Able to get large bandwidth in some situation (sometimes > 100 MSps)
- Others supporting > 300 MSps incoming...





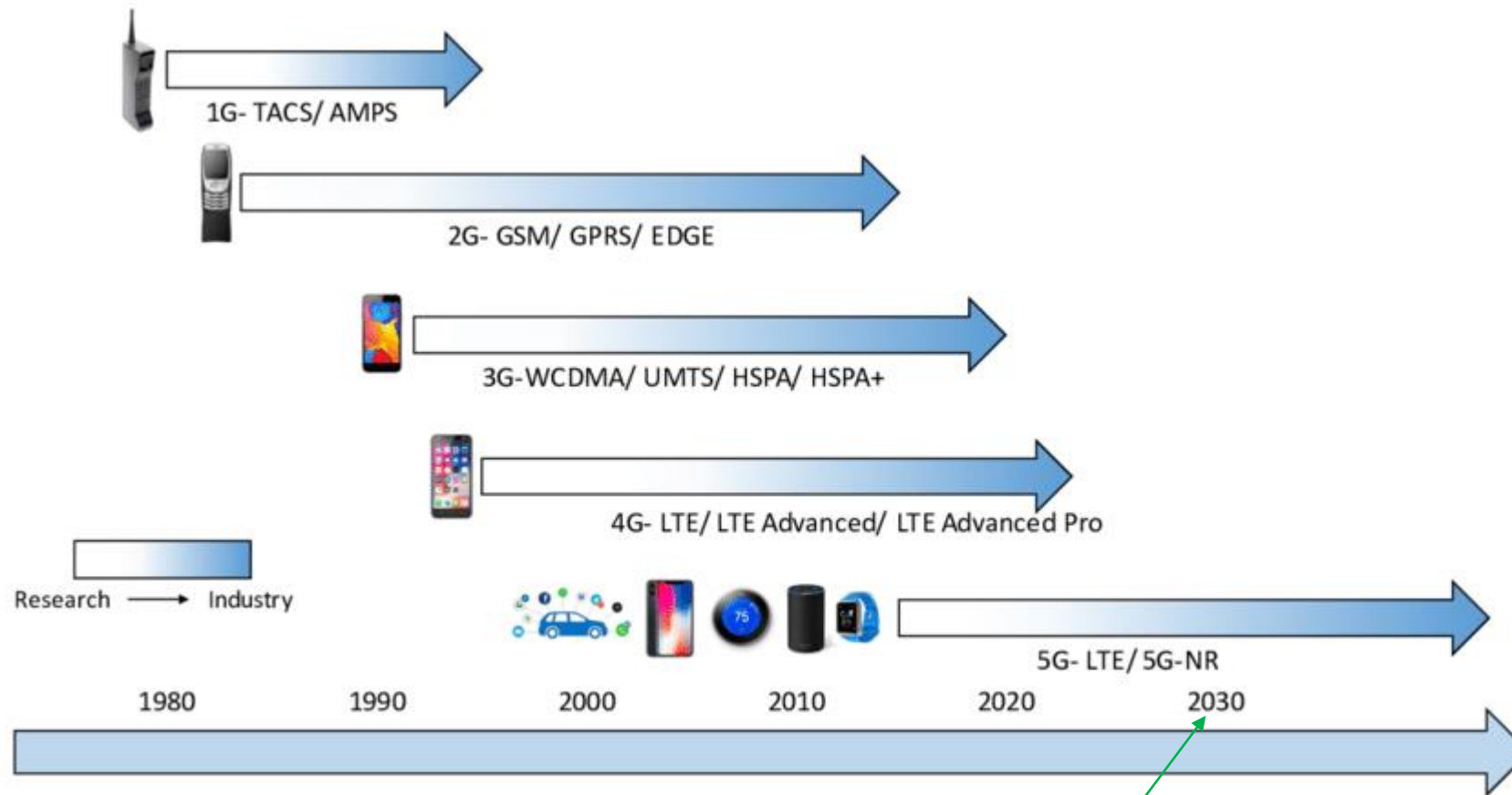
Summary

- 01** Evolution of mobile networks in brief
- 02** Mobile radio pentests in 5G
- 03** 5G NewCore network pentests
- 04** Looking further on protocol stack vulnerabilities
- 05** Conclusion



Evolution of mobile networks

Mobile network* → more than 30 years

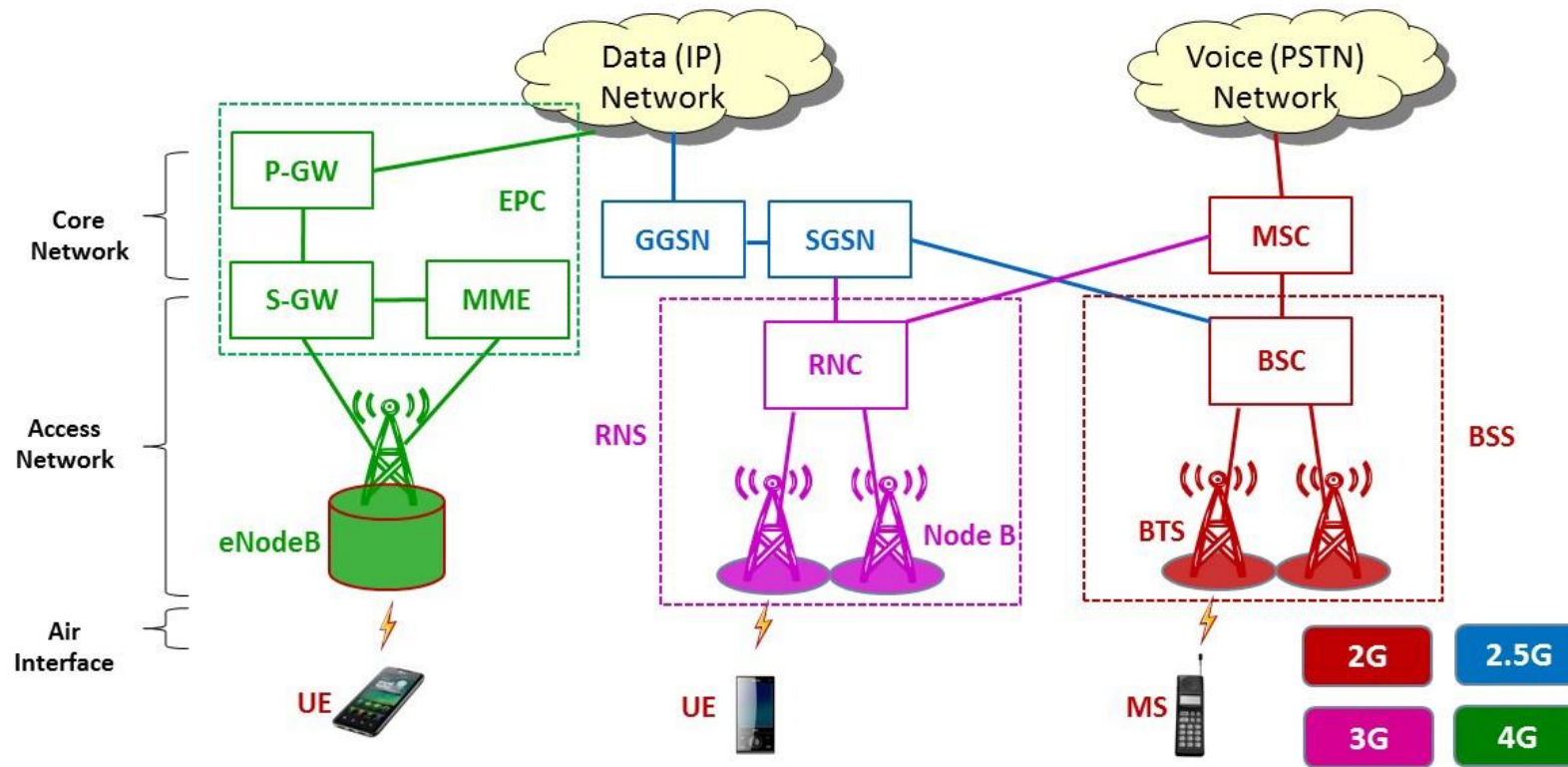


* Other technologies exist for US and Asia

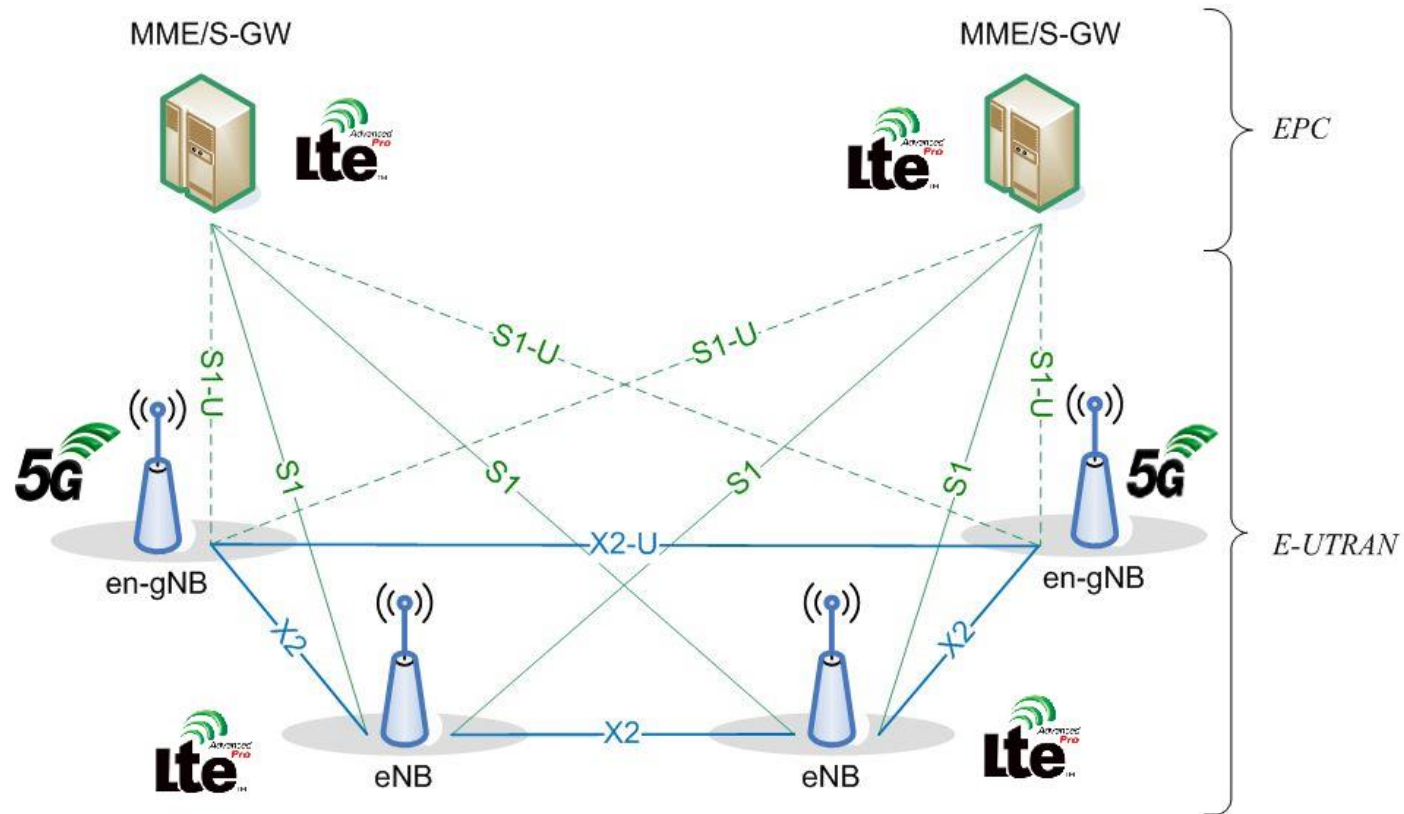
And we are already talking about 6G!

2G-4G Networks

2G, 3G & 4G Network Architecture

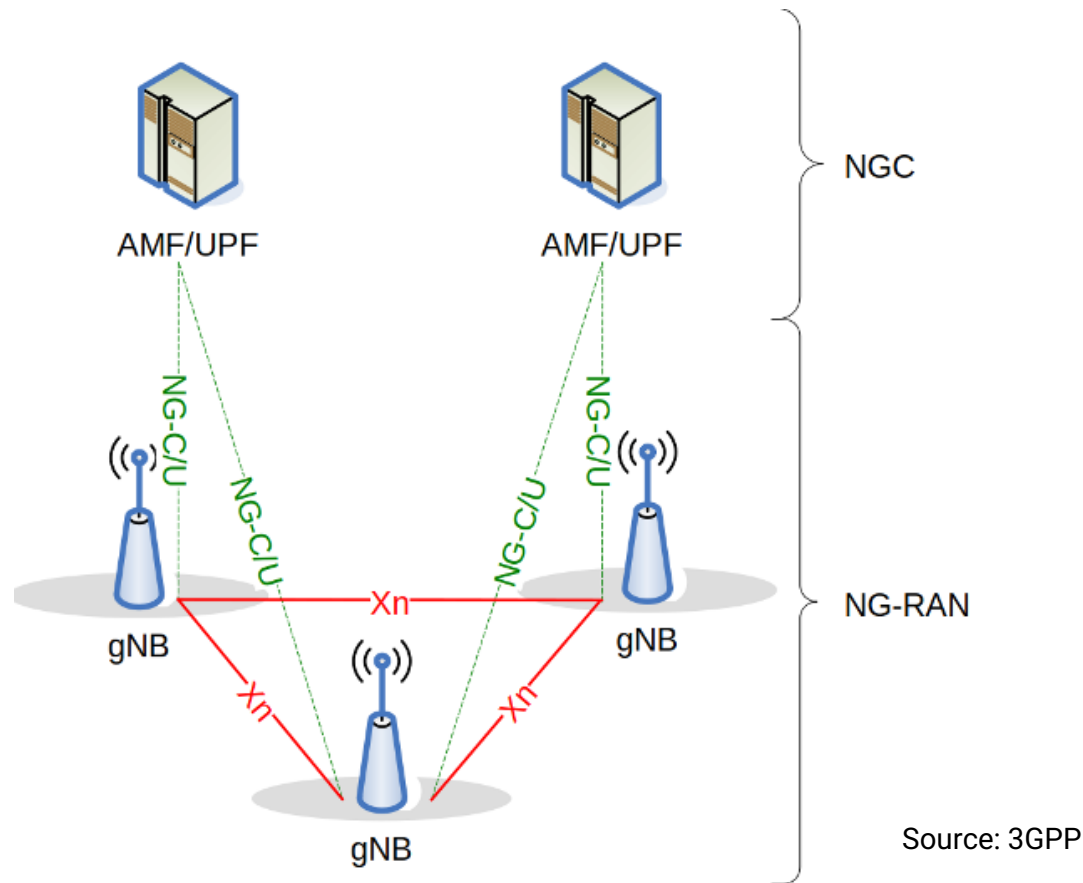


5G NSA: what we have currently



Source: 3GPP

5G SA: what should we expect in 2021/2022?



Use of SDN (Software-Defined Network) → CUPS, reduce operation costs, faster services, plug/unplug instances in the network, etc.



Mobile radio pentests in 5G

Some 5G Targets

- Goal:
 - Extract secrets exchanged between the device and the backend
 - Attack the device and its exposed services
 - Get knowledge on how to interact with the backend → attack!
- Mostly devices support different mobile stacks 2G/3G/4G, etc.
- We can still use older stacks to perform assessment



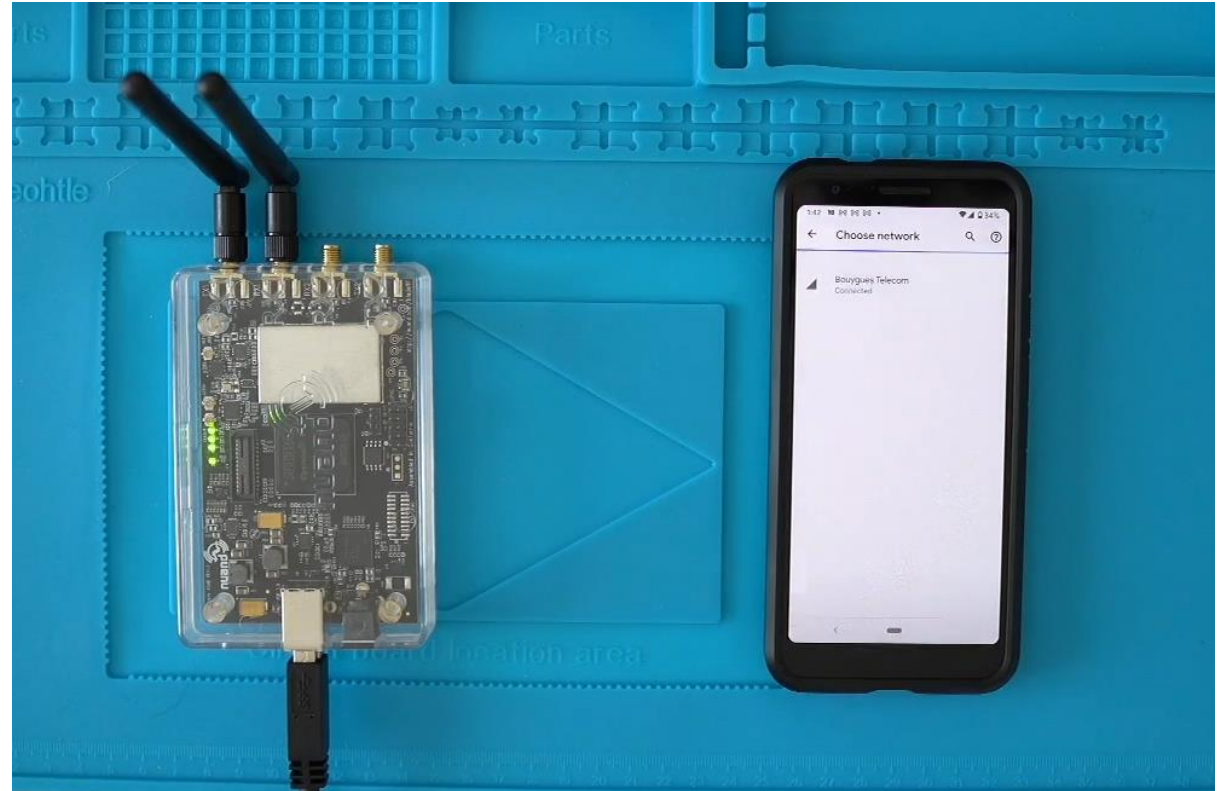
Isolation

- Use of Faraday cage/shield: ~5000€ (custom one)
- Goal: avoid conflicts with operators + isolate from other devices, noise, etc.
- Cheaper ways:
 - A home made can be made + using attenuation in software + hardware → but not certified ͇_(\ツ)_/͇
 - Possible to use good well made SMA extension cable → if external antennas



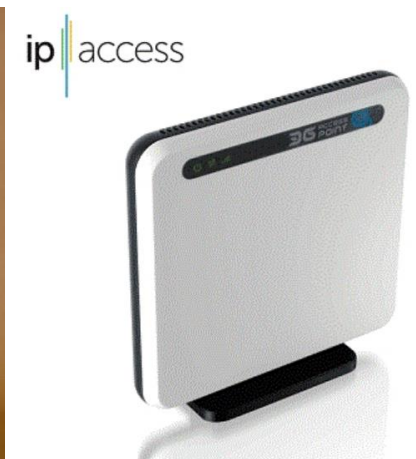
Setup with SDR in 2G

- SDR: Software-Defined Radio
- 2G:
 - USRP (~900€ minimum): OpenBTS, or Osmocom GSM softwares (with E/GPRS)
 - BladeRF 1 or 2 (~400€): YateBTS, OpenBTS with YateBTS transceiver
 - LimeSDR (~300€): OpenBTS without GPRS, or Osmocom GSM softwares (with E/GPRS)
 - Others... (UmTRX, etc.)



Setup with SDR in 3G

- 3G:
 - USRP (~900€ minimum): OpenBTS-UMTS
 - But can only operate with SIM (mode) cards != USIM (mode)
- Anyway, it is also possible to use femtocells to compensate:
 - Further read: https://www.isti.tu-berlin.de/fileadmin/fg214/Papers/conf_t2_2010.pdf
 - ip.access femto packaged by Sysmocom: <https://www.sysmocom.de/products/lab/3g5starterkit/index.html>



Setup with SDR in 4G

- Opensource:
 - Minimum budget: ~300€
 - OpenLTE
 - Open AIR Interface (supporting LTE-A)
 - srsRAN (previously srsLTE) based on OpenLTE
- Commercial:
 - Amarisoft (also supports NB-IoT, Cat-M1)
 - srsRAN with NB-IoT extension
 - Rohde&Schwarz



Setup with SDR in 5G-NR = more \$\$\$

- Opensource:
 - Working budget: >20 000€
 - Hardware (1* USRP X300 + 2* TX/RX daughter board)
 - srsRAN (only NSA but limited): needs 2 antennas for 4G and 5G
 - USRP B210 not working (USB 3 interface maybe too slow)
 - OpenAirInterface5G: still under-dev?
 - Etc.
- Commercial:
 - Amarisoft → preferable for SA for the moment
 - Minimum budget: >20 000€ (hardware + license)
 - Rohde&Schwarz



Mini version only supports 5G-NR SA

Security comparison in brief

	2G	3G	4G	5G
Client authentication	YES	YES	YES	YES
Network authentication	NO	Only in USIM mode	YES	YES
Signaling integrity	NO	YES	YES	YES
Encryption	A5/1 in use	KASUMI SNOW-3G	SNOW-3G AES-128 CTR ZUC	SNOW-3G AES-128 CTR ZUC

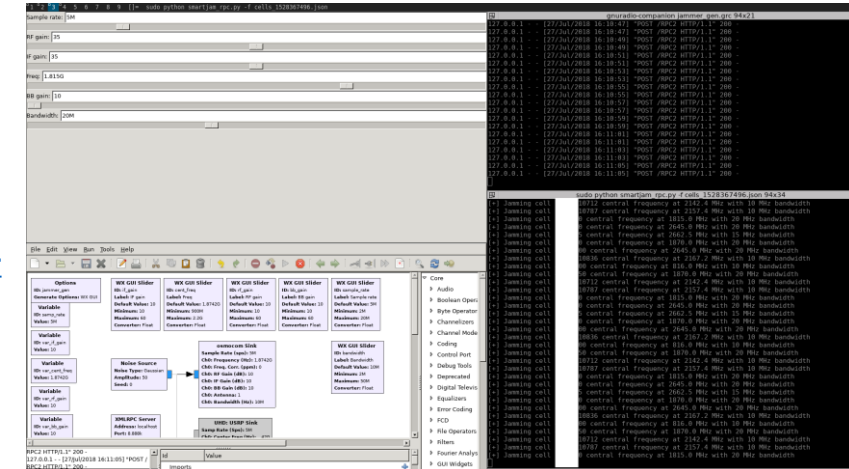
Downgrading security

- Goal → downgrade to 2G → bypass mutual authentication
- Could be done naturally (parking stations, uncovered areas...)
- Complex way: using signaling vulnerability/tricks (e.g: https://www.researchgate.net/figure/Mobile-device-soft-downgrading-to-GSM-by-rogue-LTE-base-station_fig6_305401180)
- Or dumb way with Jamming:
 - Dedicated device → available on AliExpress
 - Smart-jamming:
 - ModMobMap (soon available for 5G) & Modmobjam

→ <https://github.com/PentHertz/Modmobjam>

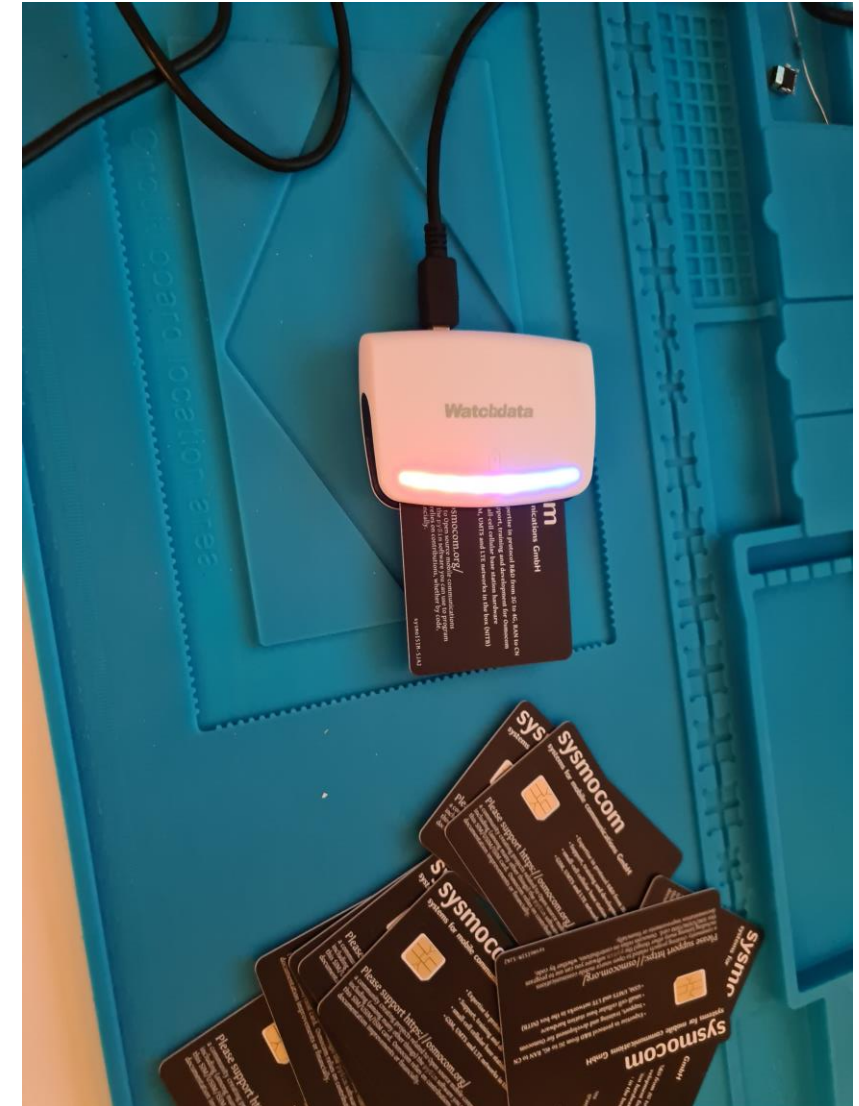
Caution: > stacks to downgrade → more resources!

Caution 2: Jamming is not legal



Use of custom ISIM cards

- Goal: Complete mutual authentication
- At least a custom USIM, preferably ISIM:
 - USIM: Universal Subscriber Identity Module
 - ISIM: IP Multimedia Services Identity Module → SIP/IMS procedure
 - Generate Ki, OPC → provide on the gNB side
- Reference: <http://shop.sysmocom.de/products/sysmoISIM-SJA2> (USIM version are interesting to disable USIM mode)
- Only a PC/SC reader is needed
- Cheaper version → AliExpress (with a *nice* software)



Use of custom ISIM cards (2)

```
fluxius@trendmate ~/Projects/USIM/pysim <master>
$ sudo python3 pySim-prog.py -p0 -t sysmoISIM-SJA2 -i 901700000048419 -c 33 -x 001 -y 01 -s 8988211000000484199 -a 63682414
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
Using PC/SC reader interface
Ready for Programming: Insert card now (or CTRL-C to cancel)
Generated card parameters :
> Name      : Magic
> SMSP      : e1ffffffffffffffffffffffff058100335555ffffffffffffffff000000
> ICCID     : 8988211000000484199
> MCC/MNC   : 001/01
> IMSI      : 901700000048419
> Ki        : c6bd3a9172b188b8daf69643b65d686a
> OPC       : 787f3270a48ef92b91146f5ef21681ea
> ACC       : None
```

Note: Need of mutual authentication → need to know secrets → but an attacker can have access to SS7, DIAMETER, 5G infra (we will few aspects later...)

5G SA support on phones

- Be careful of supported bands!
- A good reference:
<https://cacombos.com/>
- But despite these notes
→ surprises!

LG Velvet 5G (T-Mobile USA) (LM-G900TM) 4G/5G Bands and Combos

Modem Specification	
Modem Model	Dimensity 1000C
Release Year	2020
LTE DL/UL Modulation	256QAM / 64QAM
LTE Bands	1, 2, 3, 4, 5, 8, 12, 13, 17, 20, 25, 26, 28, 39, 41, 66, 71
LTE 4x4 Bands	2, 4, 25, 41, 66
LTE Category (DL/UL) ?	18 / 13
LTE Max Speed (DL/UL) ?	1200 / 150 Mbps
NR NSA Bands	25, 41, 66, 71
NR SA Bands	71

5G NR Bands

- To be respected in the configuration:

NR band n71 basic information

RAT	NR	NR band	n71										
Name	600	Duplex mode	FDD										
Frequency (UL)	663.00 MHz - 698.00 MHz	Frequency (DL)	617.00 MHz - 652.00 MHz										
NR-ARFCN (UL)	132600 - 139600	NR-ARFCN (DL)	123400 - 130400										
N _{ref} Step	20	ΔF _{raster} (kHz)	100										
Band bandwidth (UL/DL)	35 MHz	Duplex spacing	46 MHz										
Geography area	NAR												
Channel bandwidth	Channel Bandwidth(MHz)												
SCS(KHz)	5	10	15	20	25	30	40	50	60	70	80	90	100
15	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
30	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
60	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

NR band n71 spectrum/overlapped bands



<https://itectec.com/band/nr-band-n71/>

But support is “fictive” sometimes...

- Not all phones supports 5G SA
- Even if the commercial tells you so (e.g.: Exynos basebands, MKT Dimensity 1000C, HiSilicon, etc.)
- Some Huawei phones used to support it (e.g.: Mate X):
 - But now only in firmware for CN → need to downgrade them, maybe patch them
 - New upgrade in EU → restrict SA
 - Sometimes need to adjust PLMN to Chinese one + in ISIM (e.g.: MCC/MNC = “460 11”)

MCC	MNC	Brand	Operator
460	00	China Mobile	China Mobile
460	01	China Unicom	China Unicom
460	02	China Mobile	China Mobile
460	03	China Telecom	China Telecom
460	04	China Mobile	Global Star Satellite
460	05	China Telecom	China Telecom
460	06	China Unicom	China Unicom
460	07	China Mobile	China Mobile
460	08	China Mobile	China Mobile
460	09	China Unicom	China Unicom
460	11	China Telecom	China Telecom
460	20	China Tietong	China Tietong

Ref:
[https://en.wikipedia.org/wiki/Mobile_network_codes_in_ITU_region_4xx_\(Asia\)](https://en.wikipedia.org/wiki/Mobile_network_codes_in_ITU_region_4xx_(Asia))

Case of failure (e.g: LG Velvet 5G)

- Fail even respecting:
 - Duplex mode
 - NR-ARFCN
 - NR band
 - SCS
 - Etc.
- Possible to debug →
LTE registration try

```
22:56:00.267 [NAS] UL 0003 EMM: Attach request
Protocol discriminator = 0x7 (EPS Mobility Management)
Security header = 0x1 (Integrity protected)
Auth code = 0x9b6dbbb3
Sequence number = 0x03
Protocol discriminator = 0x7 (EPS Mobility Management)
Security header = 0x0 (Plain NAS message, not security protected)
Message type = 0x41 (Attach request)
EPS attach type = 2 (combined EPS/IMSI attach)
NAS key set identifier:
  TSC = 0
  NAS key set identifier = 0
Old GUTI or IMSI:
  MCC = 001
  MNC = 01
  MME Group ID = 32769
  MME Code = 1
  M-TMSI = 0x84749e1a
UE network capability:
  0xf0 (EEA0=1, 128-EEA1=1, 128-EEA2=1, 128-EEA3=1, EEA4=0,
  0xf0 (EIA0=1, 128-EIA1=1, 128-EIA2=1, 128-EIA3=1, EIA4=0,
  0xc0 (UEA0=1, UEA1=1, UEA2=0, UEA3=0, UEA4=0, UEA5=0, UEA
  0xc0 (UCS2=1, UIA1=1, UIA2=0, UIA3=0, UIA4=0, UIA5=0, UIA
  0x1d (ProSe-dd=0, ProSe=0, H.245-ASH=0, ACC-CSFB=1, LPP=1
  0x00 (ePCO=0, HC-CP ClOT=0, ERw/oPDN=0, S1-U data=0, UP C
  0x10 (15 bearers=0, SGC=0, N1mode=0, DCNR=1, CP backoff=0
```

Missing N1 mode



Case of success

- Expected even in LTE registration:

```
00:01:03.640 [NAS] UL 003d EMM: Attach request
0000: 17 91 af 06 b7 11 07 41 12 0b f6 00 f1 10 80 01 .....A.....
0010: 01 e2 f8 e3 5f 07 f0 70 c0 40 18 80 b0 00 38 02 ....p.@....8.
0020: 06 d0 11 d1 27 31 80 80 21 10 01 00 00 10 81 06 ....'1!.....
0030: 00 00 00 00 83 06 00 00 00 00 00 0d 00 00 0a 00 .....
0040: 00 05 00 00 10 00 00 11 00 00 17 01 02 00 1a 01 .....
0050: 01 00 23 00 00 24 00 5c 0a 00 31 04 65 e0 34 01 ..#..$.\\..1.e.4.
0060: 90 11 03 57 58 a6 5d 01 00 e0 c1 6f 04 f0 00 70 ...WX.]....o...p
0070: 00 6d 01 00 .....m..
Protocol discriminator = 0x7 (EPS Mobility Management)
Security header = 0x1 (Integrity protected)
Auth code = 0x91af06b7
Sequence number = 0x11
Protocol discriminator = 0x7 (EPS Mobility Management)
Security header = 0x0 (Plain NAS message, not security protected)
Message type = 0x41 (Attach request)
EPS attach type = 2 (combined EPS/IMSI attach)
NAS key set identifier:
  TSC = 0
  NAS key set identifier = 1
Old GUTI or IMSI:
  MCC = 001
  MNC = 01
  MME Group ID = 32769
  MME Code = 1
  M-TMSI = 0xe2f8e35f
UE network capability:
  0xf0 (EEA0=1, 128-EEA1=1, 128-EEA2=1, 128-EEA3=1, EEA4=0, EEA5=0, EEA6=0, EEA7=0)
  0x70 (EIA0=0, 128-EIA1=1, 128-EIA2=1, 128-EIA3=1, EIA4=0, EIA5=0, EIA6=0, EIA7=0)
  0xc0 (UEA0=1, UEA1=1, UEA2=0, UEA3=0, UEA4=0, UEA5=0, UEA6=0, UEA7=0)
  0x40 (UCS2=0, UIA1=1, UIA2=0, UIA3=0, UIA4=0, UIA5=0, UIA6=0, UIA7=0)
  0x18 (ProSe-dd=0, ProSe=0, H.245-ASH=0, ACC-CSFB=1, LPP=1, LCS=0, 1xSRVCC=0, NF=0)
  0x80 (ePCO=1, HC-CP CIOt=0, ERw/oPDN=0, S1-U data=0, UP CIOt=0, CP CIOt=0, ProSe-relay=0, ProSe-dc=0)
  0xb0 (15 bearers=1, SGC=0, N1mode=1, DCNR=1, CP backoff=0, RestrictEC=0, V2X PC5=0, multipleDRB=0)
```

Observations: only Snapdragon X5x are up to support SA in EU

5G-NR SA registration: Amarisoft's Web interface

Logs: 6217

MME

Stats

IMS

ENB

UL/DL

Layer

UE ID

IMSI

Cell ID

Info

Level

Time origin: 00:00:00.000

Group UE ID:

Clear

Search

Analytics

RB

UE Caps

Time	Diff	RAN	CN	IMS	UE ID	IMSI	Cell	SFN	RNTI	Info	Message
-		PHY			3		1	4 399.19	0x4603	PUCCH	format=1 prb=50 prb2=0 symb=0:14 cs=0 occ=0 ack=1 snr=21.
-		RLC			3					DRB1	D/C=0 CPT=0 ACK_SN=39
-		MAC			3		1				LCID:4 len=3 PAD: len=219
-		PHY			3		1	4.400.6	0x4603	PDCCH	ss_id=2 cce_index=4 al=2 dci=1_1
-		PHY			3		1	4.400.6	0x4603	PDSCH	harq=0 prb=50 symb=1:13 k1=12 nl=2 CW0: tb_len=225 mod=
23:53:01.691	+0.003	GTPU									127.0.1.100:2152 G-PDU TEID=0x32149638 QFI=1 SDU_le
23:53:01.692	+0.001	PDCP			3					DRB1	D/C=1 SN=41
-		RLC			3					DRB1	D/C=1 P=1 SI=00 SN=41
-		MAC			3		1				LCID:4 len=59 PAD: len=163
-		PHY			3		1	4 400.14	0x4603	PDCCH	ss_id=2 cce_index=4 al=2 dci=1_1
-		PHY			3					PDSCH	harq=1 prb=50 symb=1:13 k1=5 nl=2 CW0: tb_len=225 mod=8
-		GTPU									127.0.1.100:2152 G-PDU TEID=0x32149638 QFI=1 SDU_le
-		PHY			3		1	4.400.8	0x4603	PUCCH	format=1 prb=50 prb2=0 symb=0:14 cs=8 occ=2 sr=1 snr=21.5
-		PDCP			3					DRB1	D/C=1 SN=42
-		RLC			3					DRB1	D/C=1 P=1 SI=00 SN=42
-		MAC			3		1				LCID:4 len=59 PAD: len=163
-		PHY			3		1	4 400.15	0x4603	PDCCH	ss_id=2 cce_index=2 al=2 dci=0_1 k2=4
-		PHY			3		1	4.400.15	0x4603	PDCCH	ss_id=2 cce_index=6 al=2 dci=1_1
-		PHY			3		1	4 400.15	0x4603	PDSCH	harq=2 prb=50 symb=1:13 k1=4 nl=2 CW0: tb_len=225 mod=8
23:53:01.693	+0.001	PHY			3		1	4.400.9	0x4603	PUCCH	format=2 prb=49 prb2=1 symb=2:2 csi=1011111 epre=-104.4
23:53:01.697	+0.004	PHY			3		1	4.400.18	0x4603	PUCCH	format=1 prb=50 prb2=0 symb=0:14 cs=0 occ=0 ack=1 snr=21.

From: ENB

Info: 192.168.1.116:9001, v2021-09-18

Time: 23:52:49.213

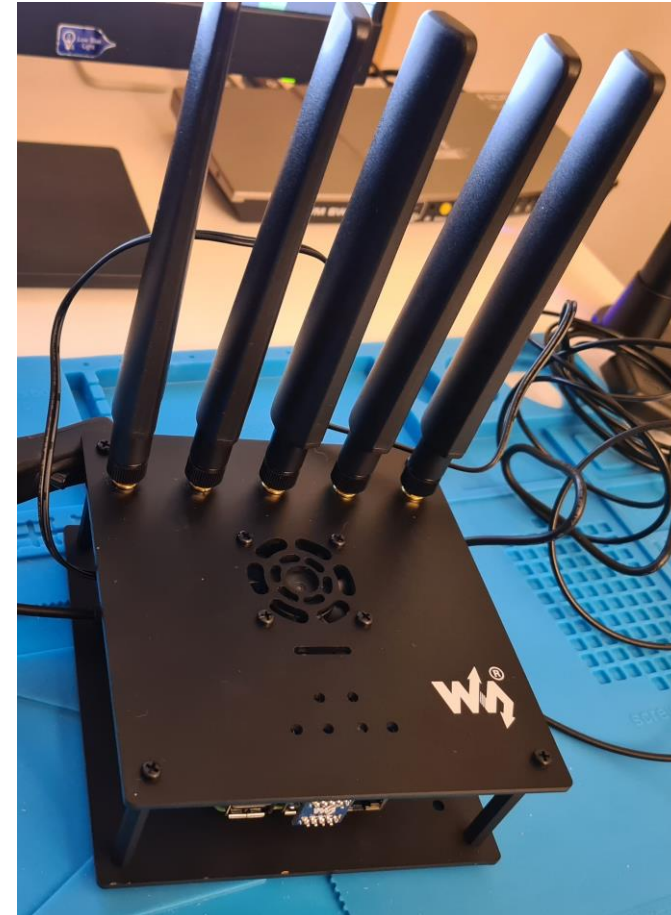
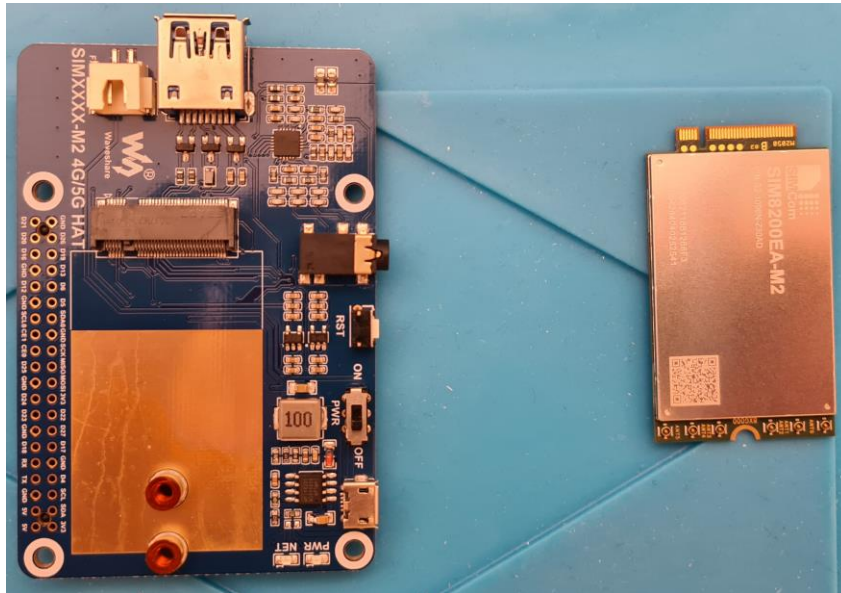
Index: 5031

Message: format=2 prb=49 prb2=1 symb= epre=-106.5

ENB constellations

Extracting secret: target simulation

- Let's simulate a 5G-NR SA target
- Base on SIM8200EA-M2 + adapter → Raspberry Pi → perform the communication



Extracting secret: monitoring net interfaces

- Looking at right interface, we can then smoothly monitor:

7	96.959980	192.168.2.2	194.0.5.123	NTP
8	96.971592	194.0.5.123	192.168.2.2	NTP
9	129.219952	192.168.2.2	194.0.5.123	NTP
10	129.228208	194.0.5.123	192.168.2.2	NTP
11	147.179925	192.168.2.2	8.8.8.8	DNS
12	147.189364	8.8.8.8	192.168.2.2	DNS
13	152.099879	192.168.2.2	8.8.8.8	DNS
14	152.109304	8.8.8.8	192.168.2.2	DNS
15	161.279902	192.168.2.2	194.0.5.123	NTP
16	161.287666	194.0.5.123	192.168.2.2	NTP
17	180.659918	192.168.2.2	37.187.1.112	UDP

▶ Frame 17: 52 bytes on wire (416 bits), 52 bytes captured (416 bits)
Raw packet data
▶ Internet Protocol Version 4, Src: 192.168.2.2, Dst: 37.187.1.112
▶ User Datagram Protocol, Src Port: 37894, Dst Port: 6999
▼ Data (24 bytes)
Data: 706173737068726173653d4023665230404023446662642e
[Length: 24]

```
0010 25 bb 01 70 94 06 1b 57 00 20 4b 8b 70 61 73 73 %..p...W.K.pass
0020 70 68 72 61 73 65 3d 40 23 66 52 30 40 40 23 44 phrase=@ #fR0@#@D
0030 66 62 64 2e fbd.
```

```
140: tun1: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdis
p default qlen 500
    link/none
    inet 192.168.3.1/24 scope global tun1
        valid_lft forever preferred_lft forever
    inet6 fe80::b9a2:a514:b97d:f8c3/64 scope link stable-privacy
        valid_lft forever preferred_lft forever
141: tun2: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdis
p default qlen 500
    link/none
    inet 192.168.4.1/24 scope global tun2
        valid_lft forever preferred_lft forever
    inet6 2001:468:3000:1::/48 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::f737:cbad:e1dc:ab03/64 scope link stable-privacy
        valid_lft forever preferred_lft forever
142: tun3: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdis
p default qlen 500
    link/none
    inet 192.168.5.1/24 scope global tun3
        valid_lft forever preferred_lft forever
    inet6 2001:468:4000:1::/48 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::5856:1764:46db:347b/64 scope link stable-privacy
        valid_lft forever preferred_lft forever
[root@CBM-2021030900 ~]# tcpdump -i tun0 -w /tmp/capture5G.pcap
```

eSIM case

- Embedded SIM could be complex to hook
- But 2nd slot exists in most cases + need to force the use with AT commands

Pin name	Pin no.	Electrical description	Description	Comment
(U)SIM1_PWR	36	PO	Power supply for (U)SIM1 card	1.8/3.0V voltage domain, all (U)SIM interfaces should be protected against ESD. If unused, please keep open
(U)SIM 1_DATA	34	DIO	(U)SIM1 card data, which has been pulled up to (U)SIM1_VDD via a 20KR resistor internally	
(U)SIM 1_CLK	32	DO	(U)SIM1 clock signal	
(U)SIM1_RESET	30	DO	(U)SIM1 Reset control	
(U)SIM 1_DET	66	DI	(U)SIM1 card detect, which has been pulled up to VDD_P3 via a 470KR resistor internally	
(U)SIM2_PWR	48	PO	Power supply for (U)SIM2 card	
(U)SIM2_DATA	42	DIO	(U)SIM2 card data, which has been pulled up to (U)SIM2_VDD via a 20KR resistor internally	
(U)SIM2_CLK	44	DO	(U)SIM2 clock signal	
(U)SIM2_RESET	46	DO	(U)SIM2 Reset control	
(U)SIM2_DET	40	DI	(U)SIM2 card detect, which has been pulled up to VDD_P3 via a 470KR resistor internally	

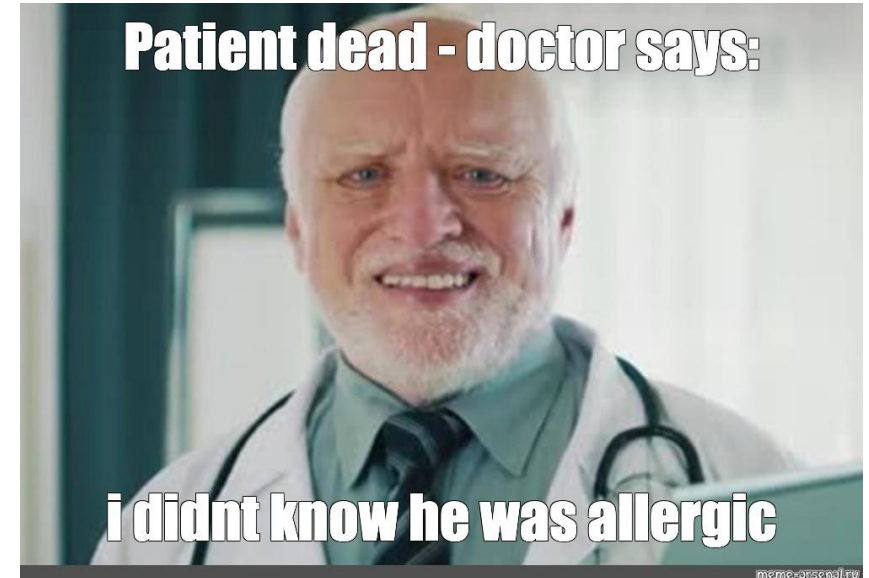
5GNC network pentests

Access to the core network

- Not an easy task
- 2 ways:
 - Legit → paying an access (SS7/DIAMETER/5G)
 - Need lot of paperwork and \$\$\$
 - Non-legit → attacking exposed gateways
- Core network can be badly configured
- Good opportunities to play with 5G core networks:
 - 5G NSA (2019): <https://5gcyberhack.fi/2019/> → 1st on the hospital assignment with DeepCuts team
 - 5G SA (2021): <https://5gcyberhack.fi/> → 3rd on PwC & Aalto assignment with DeeperCuts team
- These challenges reflects misconfigurations, or vulnerabilities real operators may have

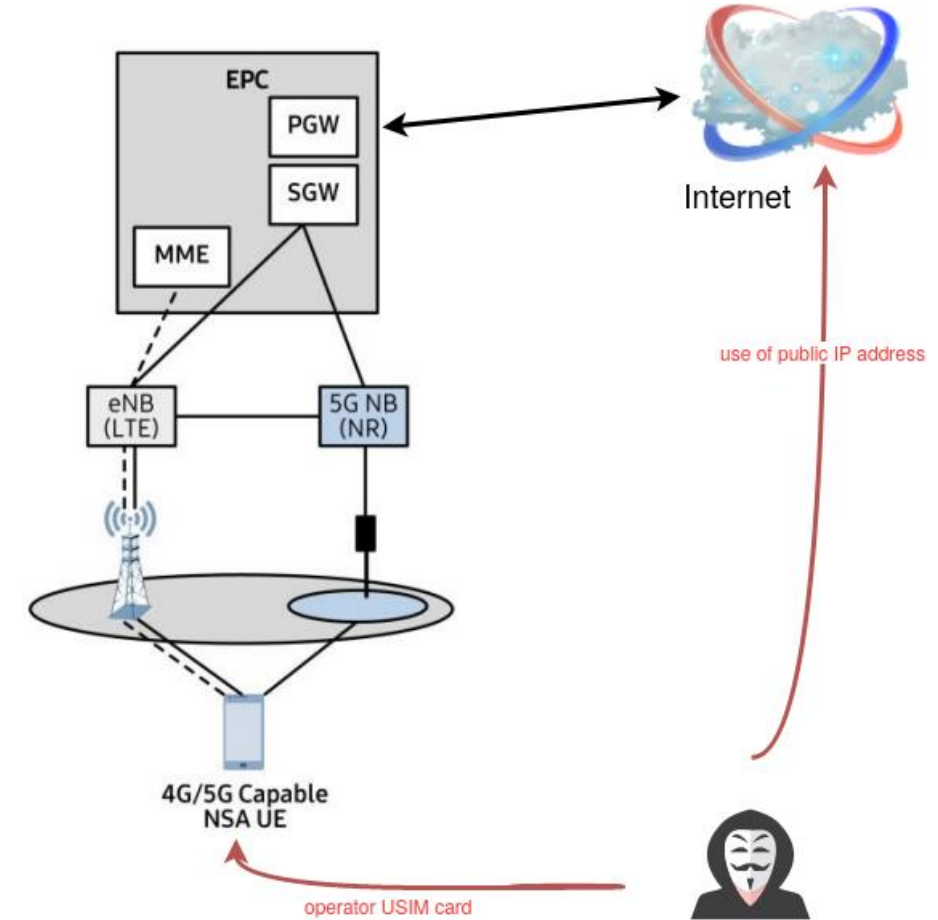
2019 5G Cybersecurity hackathon

- Target to hack:



Vectors of attacks: e.g. on 5G-NR NSA

- From public IP address → some parts could be exposed → a good scan or Shodan/Censys.io/etc. can reveal interesting things...
- But use of legit SIM card → interesting in case of bad isolation



Bad isolation

- Illustration with 2019 5G Cybersecurity Hackathon + provided SIM card:



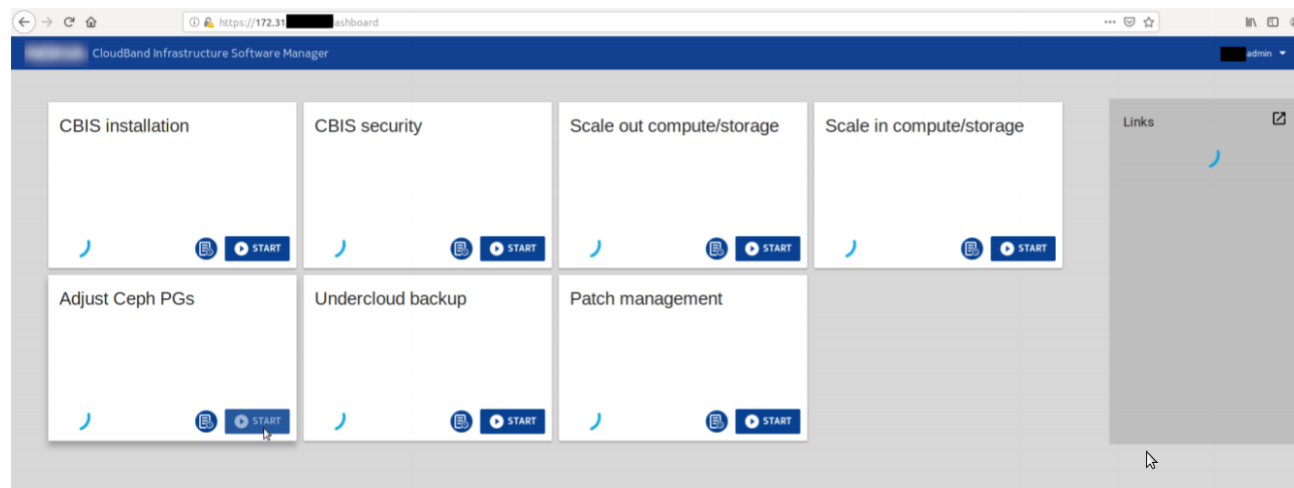
Scanning using the SIM card:

- After retrieving some subnets thanks to **traceroute**
- With right (default) operators' APN → we can try to do a Nmap scan
- And bingo! We get some interesting endpoints to test

```
Nmap scan report for 193.***.***.69
Host is up (0.025s latency).
Not shown: 957 filtered ports, 39 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.4 (protocol 2.0)
[...]
80/tcp    open  http     Apache httpd 2.4.6 ((Red Hat
Enterprise Linux))
| http-methods:
|   Supported Methods: OPTIONS GET HEAD POST TRACE
|_  Potentially risky methods: TRACE
|_ http-server-header: Apache/2.4.6 (Red Hat
Enterprise Linux)
|_ http-title: Index of /
8088/tcp  open  http     Apache httpd 2.4.6 ((Red Hat
Enterprise Linux))
| 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/
|
```

Access to endpoints → PWN of MME

- Several vulnerabilities have been found:
 - RCE to some interface → gain persistent root → pivot to other subnetworks
 - Default weak passwords
 - Traversal vulnerabilities
 - Etc.
- And of the journey:
 - Access to MME and its secrets
 - Could impersonate and compromise communications



```
Red Hat Enterprise Linux Server 7.5 (Maipo)
Kernel 3.10.0-862.9.1.el7.x86_64 on an x86_64

cmm-cmm-necc1 login: root
Password:
Last login: Sat Nov 30 18:28:06 EET 2019 from cmm-cmm-necc0.local on ssh
Last login: Sat Nov 30 18:28:44 on tty1

=====
Cloud Mobility Manager release: CMM19.0
Build: [REDACTED] mme-necc
You're logged to: necc1
Hostname: cmm-cmm-necc1.local
IP Address: 169.254.64.31/23
You are logged in as root
Time: Sat Nov 30 18:28:44 EET 2019
=====

[root@cmm-cmm-necc1 ~]# _
```

Read further

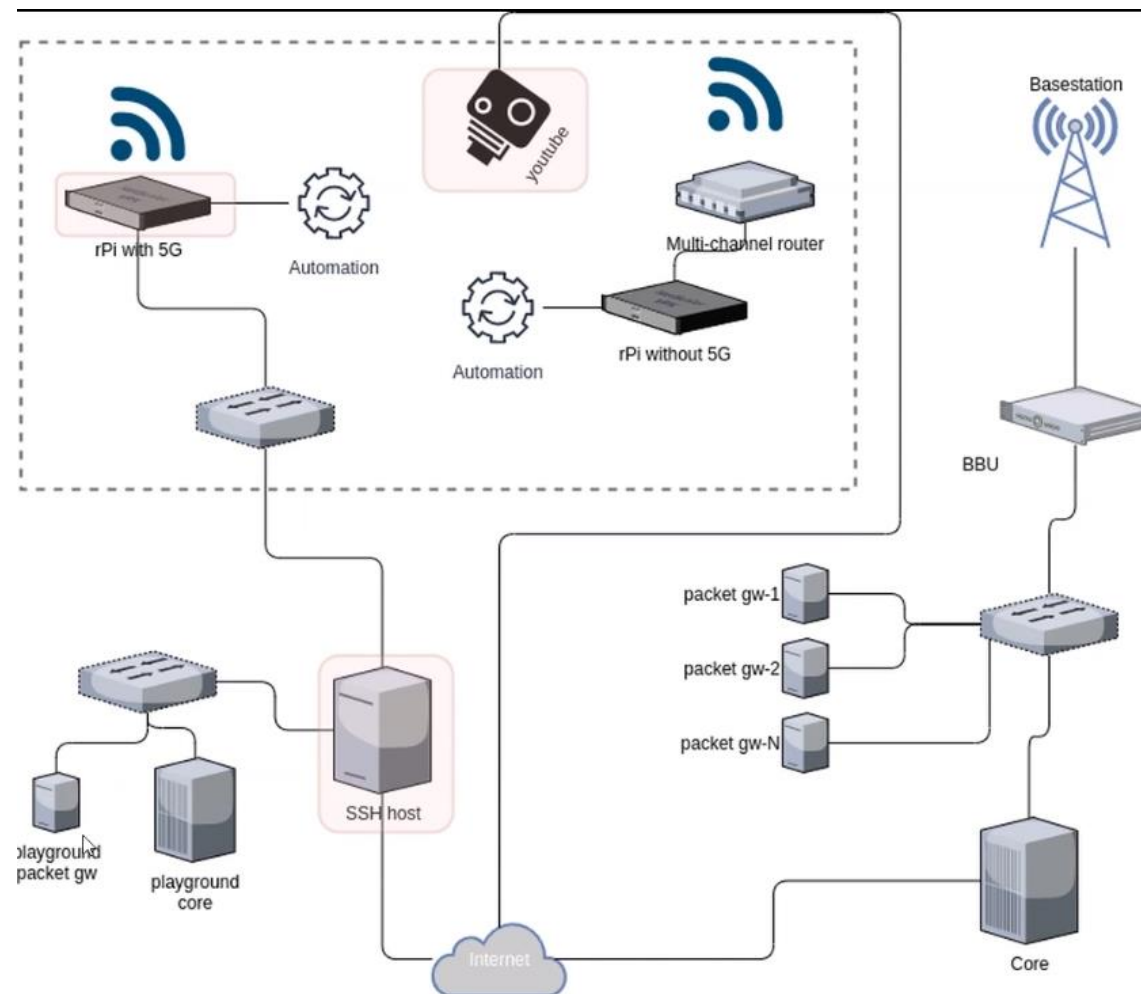
- Article: Introduction to mobile network intrusions from a cell phone (in Medium)



Similar stories but with cars on Sprint M2M network: <https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/>

5G SA: case of 2021 5G Cybersecurity event

- Got SSH access to attack the core network, with some testing playground space
- But we took the exercise to another angle...



Exposed gateways

- Scanning the public range of provided SSH IP address, we have found interesting gateway exposed
- First, we got to make a reverse lookup:

```
whois 195.148.**.*
[...]

inetnum:      195.148.***.0 - 195.148.***.255
netname:      FI-TKK****-NET
descr:        TKK Comnet
country:      FI
admin-c:      MP14***-RIPE
tech-c:       MP14***-RIPE
status:       ASSIGNED PA
mnt-by:       AS17**-MNT
created:      2009-02-13T10:44:24Z
last-modified: 2009-02-13T10:44:24Z
source:       RIPE
```

Similar story with exposed GGSN nodes: <https://positive-tech.com/expert-lab/research/vulnerabilities-of-mobile-internet-gprs-2014/>

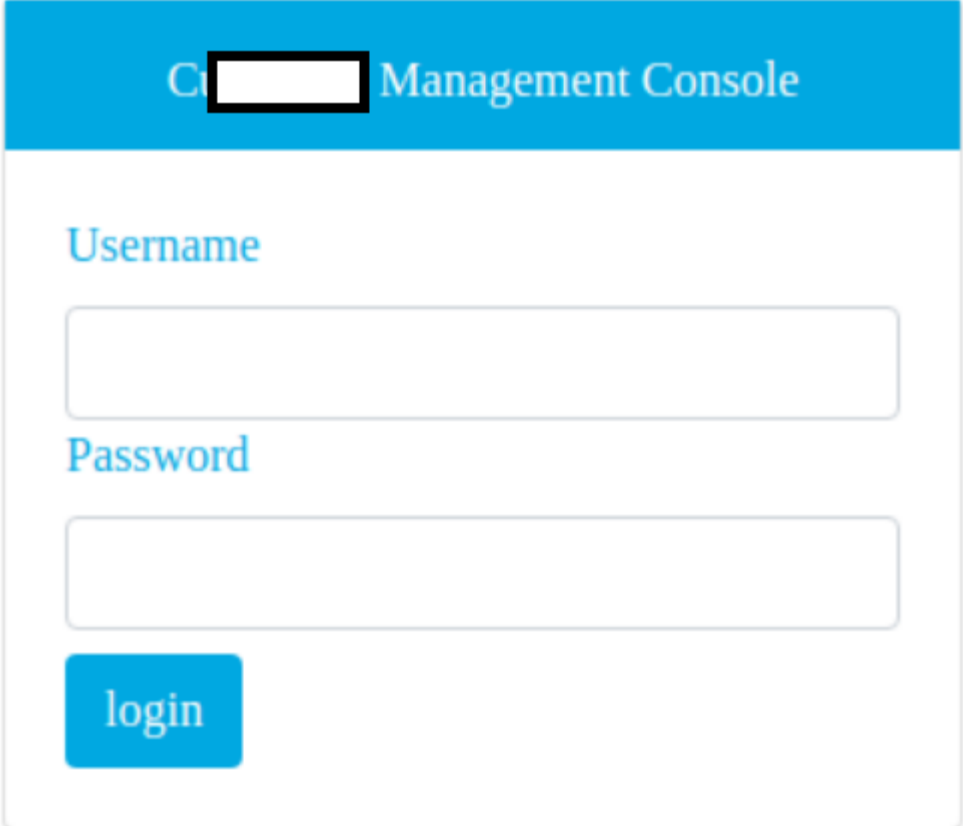
Exposed gateways (2)

- Then we identified two 5GCs with same exposed services in the range:
 - TCP 3000
 - TCP 22
 - TCP 5050

```
Nmap scan report for 5GC1.research.****.aalto.fi (195.148.***.***)  
Host is up (0.044s latency).  
Not shown: 997 closed ports  
PORT      STATE      SERVICE  VERSION  
[...]  
3000/tcp  open      ssl/http Node.js (Express middleware)  
|_ http-methods:  
|_ Supported Methods: GET HEAD POST OPTIONS  
|_ http-title: EPC USER INTERFACE  
|_ ssl-cert: Subject: commonName=www.localhost.com/organizationName=cu*****/stateOrProvinceNa  
|_ Issuer: commonName=www.localhost.com/organizationName=cumucore/stateOrProvinceName=ESP00/co
```

Exposed web interfaces

- One of the interface was vulnerable to traversal:
 - Bypass login
 - Export IMSI + Ki secrets
 - Found an SQL injection vulnerability → retrieve more information



C[redacted] Management Console

Username

Password

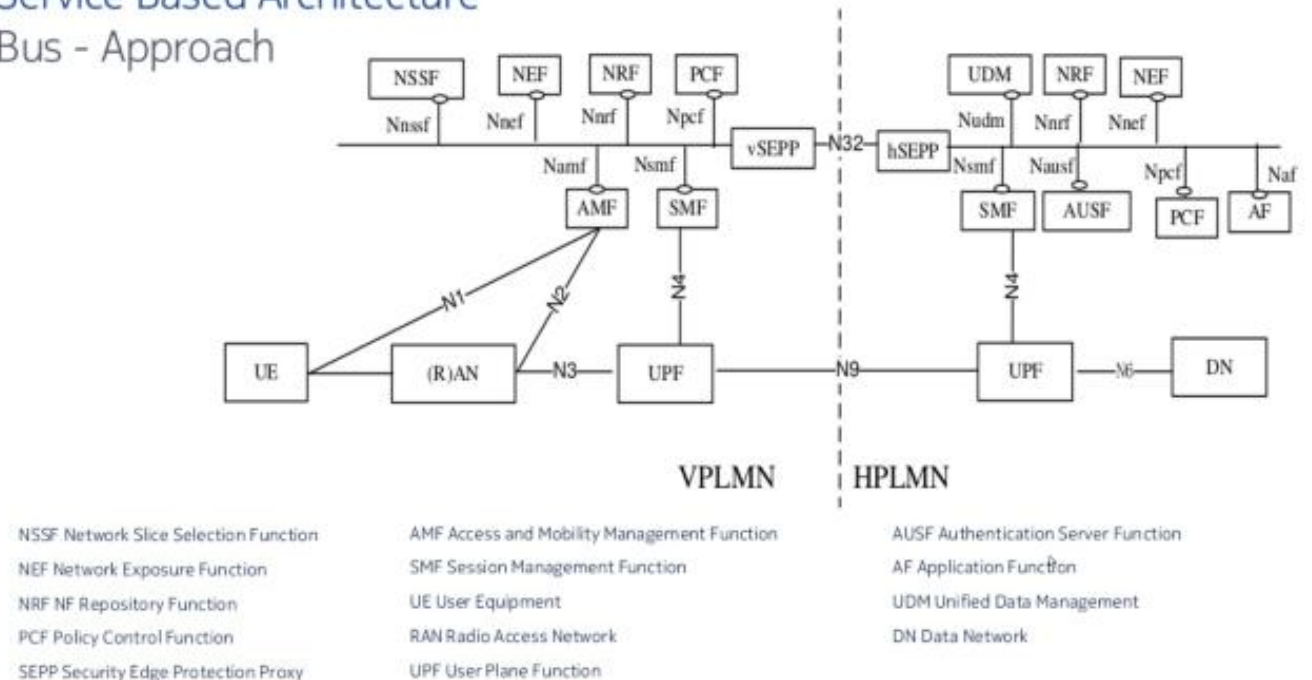
login

© Copyright 2020 [redacted] oy

NGCN

- SA mode provides Next Generation Core Network (NGCN) → replace EPC for LTE and 5G-NR NSA
- Functions are virtualized → VNF (Virtual Network Functions) → very fast connectivity + lot of applications
- 3GPP opted for a Service-Based Architecture (SBA) for control plane → services use HTTP
- Precisely RESTful API based on ETSI framework

Service Based Architecture Bus - Approach



Source: 5G VNF functions (Source: Secure Interworking Between Networks in 5G Service Based Architect)

Interacting with NRF

- NRF: Network Repository Function
- TCP port 8000 on lot of implementation, including free5gc
- Need an understanding of the API → use of OpenAPI specifications
- A nice swagger also exists
- But before → not tools to assess the NRF... ☹️

The image shows a side-by-side comparison of an OpenAPI specification and its visual representation in Swagger UI.

OpenAPI Specification (Left):

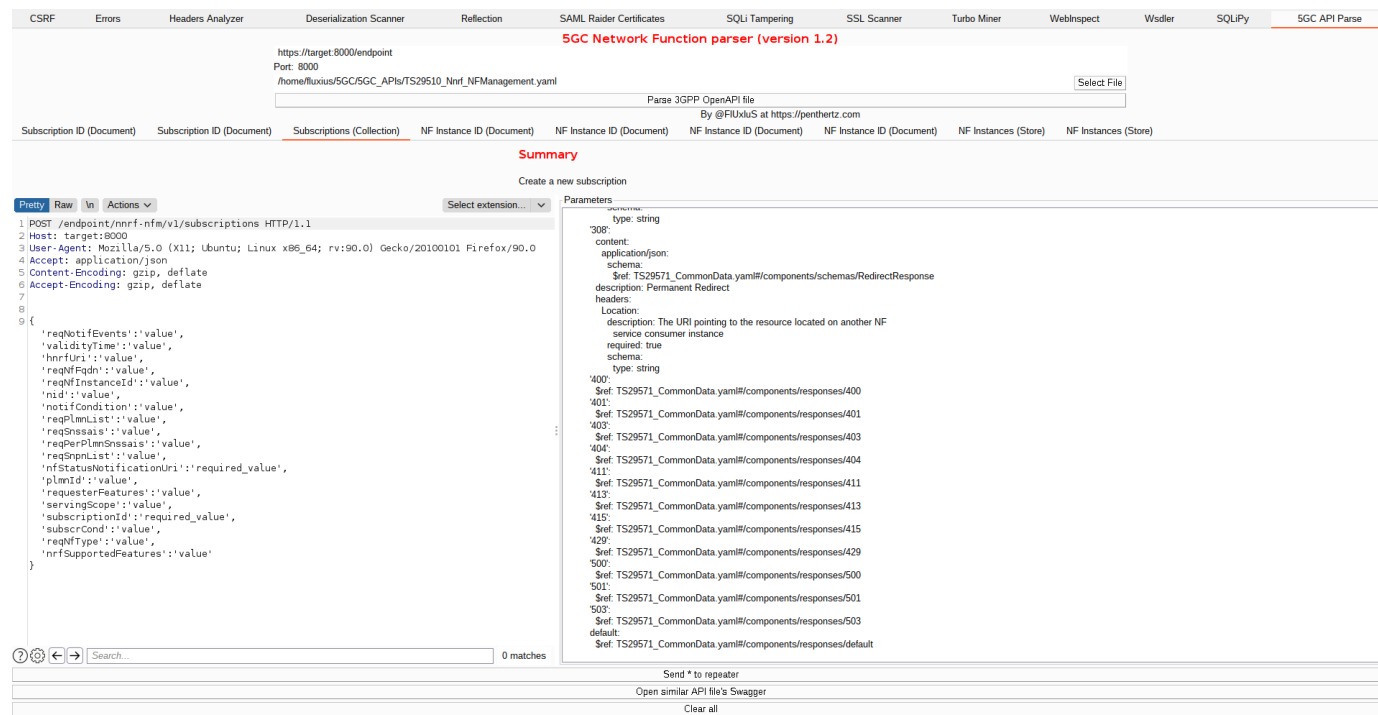
```
1 openapi: 3.0.0
2 info:
3   version: 1.0.0.alpha-3
4   title: VAE_V2X_Message_Delivery
5   description: VAE V2X Message Delivery Service
6 security:
7   - {}
8 externalDocs:
9   description: 3GPP TS 29.486 V0.3.0 V2X Application Enabler (VAE) Services
10  url: 'http://www.3gpp.org/ftp/Specs/archive/29_series/29.486/'
11 servers:
12   - url: '{apiRoot}/vae-v2x-message-delivery/v1'
13     variables:
14       apiRoot:
15         default: https://example.com
16         description: apiRoot as defined in clause 4.4 of 3GPP TS 29.501
17 paths:
18   /message-deliveries:
19     post:
20       summary: VAE V2X Message delivery resource create service Operation
21       tags:
22         - message-deliveries collection (Document)
23       operationId: CreateMessageDelivery
24       requestBody:
25         content:
26           application/json:
27             schema:
28               $ref: '#/components/schemas/V2xMessageDeliveryData'
29             required: true
30       responses:
31         '201':
32           description: Message Delivery Resource Created
33         default:
34           description: Message Delivery Resource Created
```

Swagger UI (Right):

- message deliveries collection (Document)**
 - POST** `/message-deliveries`: VAE V2X Message delivery resource create service Operation
- default**
 - GET** `/message-deliveries/{deliveryId}`
- Individual message delivery (Document)**
 - DELETE** `/message-deliveries/{deliveryId}`: VAE V2X Message delivery resource delete service Operation

We developed a tool: 5GC API parse

- Available:
https://github.com/PentHertz/5GC_API_parse
- Officially in Burp Suite as a Python Extension →
<https://portswigger.net/bappstore/bdedb48d25594922adb1a5bcc99f89c9>
- Parses OpenAPI files → initial OpenAPI parser failed
- Create queries into *Reapeter* → ready to attack the services!



Fun fact: authentication is not enforced by default 😊 → unauthenticated user can create/modify/remove things

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. The text is centered in the middle of the image.

**Go further on
protocol stacks**

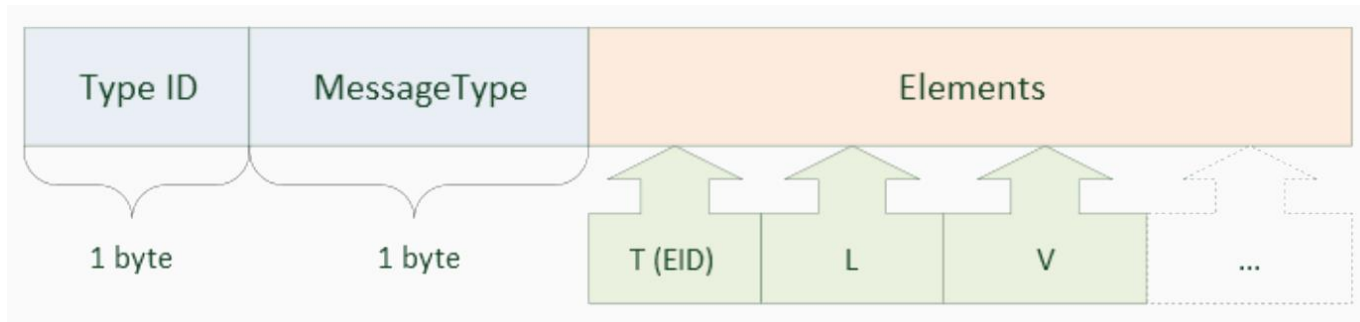
Why?

- On UE's (User Equipment) baseband:
 - Allows controlling at least the baseband → trigger silent call, send SMS, etc.
 - Controlling (part of) the application side if MMU/MCU bypass
- On the base station or core networks:
 - Having a permanent access on the operator side
 - Spying on conversations
- But several implementations exists → need to map
- Fuzzing:
 - Via Radio
 - Emulation
 - Reversing

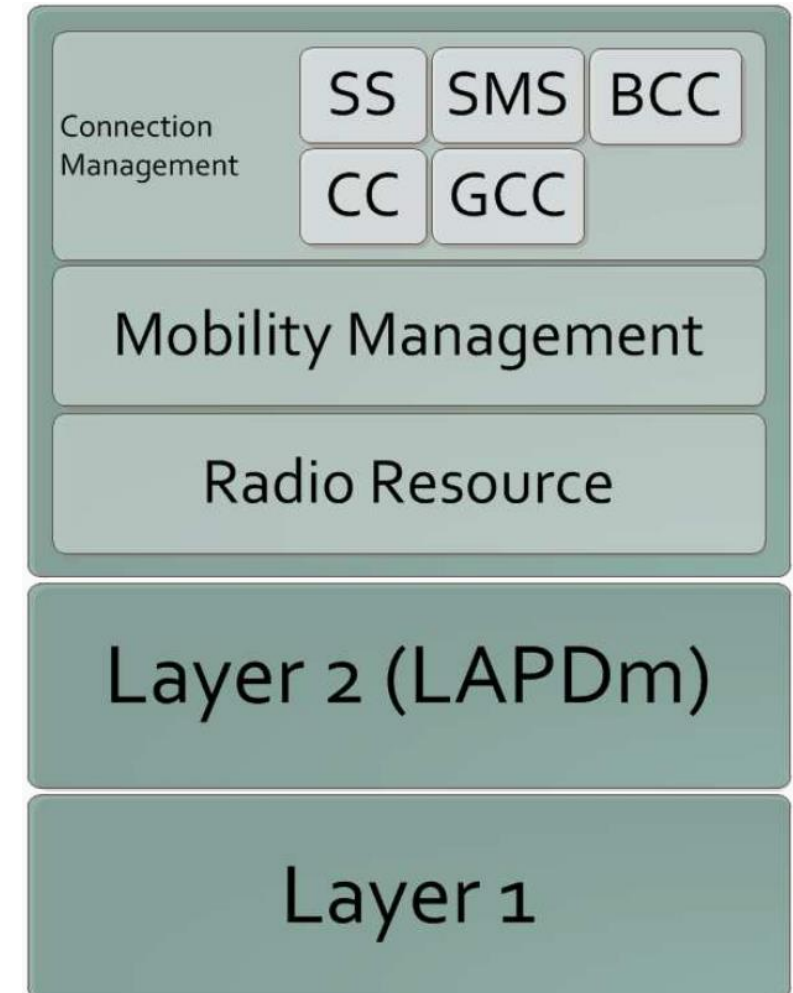
Hybrid approach is better

Interesting vectors: GSM/GPRS

- GSM/GPRS → a lot of non-encrypted/signed message can be sent
- Still lot of bugs to find → especially in state machines
- Targeting L3 messages structures:



- Tools for generation and mutation:
 - Libmitch → good to enable all elements but obsolete
 - Pycrate
- Use of OpenBTS with *testcall*, or modification of YateBTS or OsmoBTS → send payloads



From 3G

- Most of the packet are protected → most of attack as trusted network, roaming network
- Interesting new vector since LTE with VoLTE, and soon VoNR:
 - Gaining RCE using a VoLTE vulnerability on a 5G capable baseband:
<https://i.blackhat.com/USA21/Wednesday-Handouts/us-21-Over-The-Air-Baseband-Exploit-Gaining-Remote-Code-Execution-On-5G-Smartphones.pdf>
 - Attacking the XML parser of SIP

Sending raw payload in 5G

- 5G NSA:
 - Use and patch of srsRAN
- 5G SA:
 - Amarisoft → closed sourced
 - Hooking or modification on **lteenb** binary
 - Bin is now stripped → follow string references help

Mapping devices

- Map UEs:
 - For UE → use of TAC (Type Allocation Code) classification (e.g.: <http://tacdb.osmocom.org/>)
 - GSMA as also a commercial DB: <https://imei.db.gsma.com/imei/index>
- Map base station:
 - Much more complex
 - Need to correlate operators
 - Packet transmission behavior, implementation differences, etc.

UE Architectures

- UE use several different architectures:
 - ARM based: Exynos on most Samsung (starting Samsung Galaxy S6), HiSilicon on Huawei, Some MTK devices, old Qualcomm devices
 - X86: iPhone 11, 11 Pro/11 Pro Max, iPhone SE 2, Some X and 8 + some HP devices
 - MIPS (custom): some other MTK
 - Hexagon by Qualcomm: A larger number of devices, including Google Pixel Phones
- Hexagon is a complex target → no public tools, only some reversing engine for IDA, and no stable emulator yet for new modems
- But ARM based baseband are different

ARM based baseband

- Can be emulated with Qiling (<https://github.com/qilingframework/qiling>), Ghidra, or preferably Unicorn
- For fuzzing: AFL Unicorn preferably in C, or even Rust
- A lot of resource of Shannon basebands → very good to begin in baseband:
 - First resource on the Samsung Galaxy S6 with IDA scripts: <https://github.com/ni0n/shannonRE>
 - A huge progress to reverse pretty everything (Ghidra scripts, Loaders, Structures, etc.): <https://github.com/grant-h/ShannonBaseband>
- Context can be dumped in old Samsung (S6-S8 at least) → use for newer version

Qualcomm basebands

- ~10y ago:
 - REX was simpler at the time
 - **qcombbdb** could debug MSM6280 → <https://github.com/akibsayyed/qcombbdbg>
- But today:
 - Hexagon architecture
 - Tools are pretty inexistant → IDA engines (**idp_hexagon**)
 - Emulation is hard and buggy with existing tools like **Trace32** :/ → need to invest some time with **qemu_hexagon** for QDSP6
 - But may come to Unicorn? (Keystone supports Hexagon... and the Unicorn team seemed motivated to support it?)
 - Possible to *debug* a little via QXDM logs

Qualcomm based modules

- Can have an easier access to DIAG interface
- Possible also to get root on application side with special AT commands...
- Examples → SIM8200A-M2 and hidden ADB access 😊
 - *AT+CUSBCFG=usbadb,1*



- Result:
- ```
pi@raspberrypi:~ $ sudo adb shell
* daemon not running; starting now at tcp:5037
* daemon started successfully
/ # ls
WEBSERVER data lib run systemrw var
bin dev media sbin target vendor
boot etc mnt sdcard tmp www
build.prop firmware persist sys userdata
cache home proc system usr
/ # cat build.prop
ro.build.version.release=SIM7100C_4534_140830_V1.00-372-g73d227a
ro.product.name=sdxprairie-mdm
/ # ls
WEBSERVER data lib run systemrw var
bin dev media sbin target vendor
boot etc mnt sdcard tmp www
build.prop firmware persist sys userdata
cache home proc system usr
/ # id
uid=0(root) gid=0(root) groups=1004,1007,1011,1015(sdcard),1028,3001,3002,3003(inet),3006
/ # cat target
sdxprairie
```

Enjoy the trick!

# Other Qualcomm RCE on APP CPU

- Similar case with Quectel but different hidden AT commands:
  - EC20 and similar with : <https://osmocom.org/projects/quectel-modems/wiki/EC20>
  - EG25-G: <https://nns.ee/blog/2021/04/03/modem-rce.html>

# References

- Further read on baseband exploitation:
  - Baseband Attacks: Remote Exploitation of Memory Corruptions in Cellular Protocol Stacks, by Ralf-Philipp Weinmann: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.456.3213&rep=rep1&type=pdf>
  - MobiDeke: Fuzzing the GSM Protocol Stack, by Sébastien Dudek Guillaume Delugré: [http://archive.hack.lu/2012/Fuzzing\\_The\\_GSM\\_Protocol\\_Stack\\_-\\_Sebastien\\_Dudek\\_Guillaume\\_Delugre.pdf](http://archive.hack.lu/2012/Fuzzing_The_GSM_Protocol_Stack_-_Sebastien_Dudek_Guillaume_Delugre.pdf)
  - Reverse engineering and exploiting the shannon baseband, Nico Golde, and Daniel Komaromy: [https://comsecuris.com/slides/recon2016-breaking\\_band.pdf](https://comsecuris.com/slides/recon2016-breaking_band.pdf)
  - Exploitation Of A Modern Smartphone Baseband: <https://i.blackhat.com/us-18/Thu-August-9/us-18-Grassi-Exploitation-of-a-Modern-Smartphone-Baseband-wp.pdf> ← Nice example for attacking CDMA with proprietary tools
  - A walk with Shannon: A walkthrough of a pwn2own baseband exploit by Amat Cama: <https://www.youtube.com/watch?v=6bpxrfB9ioo>
  - Emulating Samsung's Baseband for Security Testing by Grant Hernandez and Marius Muench: <http://i.blackhat.com/USA-20/Wednesday/us-20-Hernandez-Emulating-Samsungs-Baseband-For-Security-Testing.pdf>
  - BaseSAFE: Baseband SANitized Fuzzing through Emulation by Dominik Maier, Lukas Seidel, and Shinjo Park: <https://arxiv.org/pdf/2005.07797.pdf>
  - And many other that don't fit this page...

# Conclusion

# To conclude

- 5G-NR SA is different → time to relearn pretty everything
- Budget for radio is expensive, but the equipment powerful SDR but also a decent interface (PCI Express is preferred for transferring samples and fast)
- However != 2G-4G it will continue to be expensive → hardware need
- A lot of confusion between 5G marketing and reality
- Much harder to downgrade for 5G → 4G → 3G → finally 2G
- But attacking core network as an outsider could be a real scenario
- NGCN is completely different from 2G-4G → new vulnerabilities to find
- Still a lot of tools and techniques are missing, but incoming in this area 😊
- Qualcomm → very complex → but interesting target still





# Thank You

Please contact us:

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

☎ +33 1 88 33 94 15

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

Watch us on

