

Researching Unisoc baseband like in the army

cp<> Slava Makkaveev





50%
of Chinese
market



10%
of Global
market

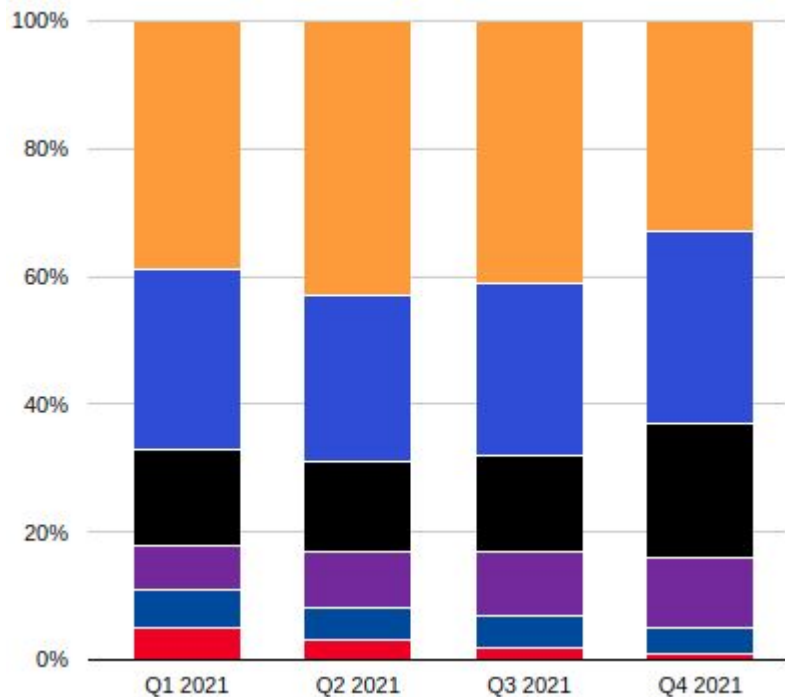
2001

2011

2018

2021

Mobile chip market share



	Q4 2021
Mediatek	33%
Qualcomm	30%
Apple	21%
Unisoc	11%
Samsung	4%
HiSilicon	1%

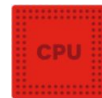
Unisoc SoC

Released in 2021/22

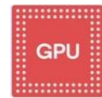
- T606
- T616
- T700
- T760

Tiger T700 SoC

CPU Cortex-A75 (x2)
CPU Cortex-A55 (x6)



GPU Mali-G52 MC2



LTE Cat. 7 modem



ISP



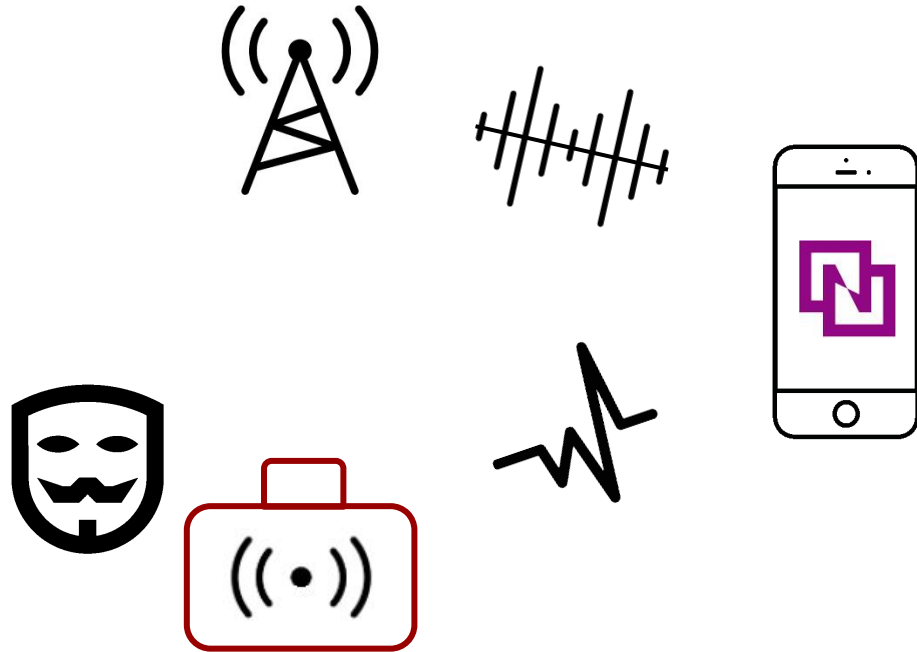
NPU



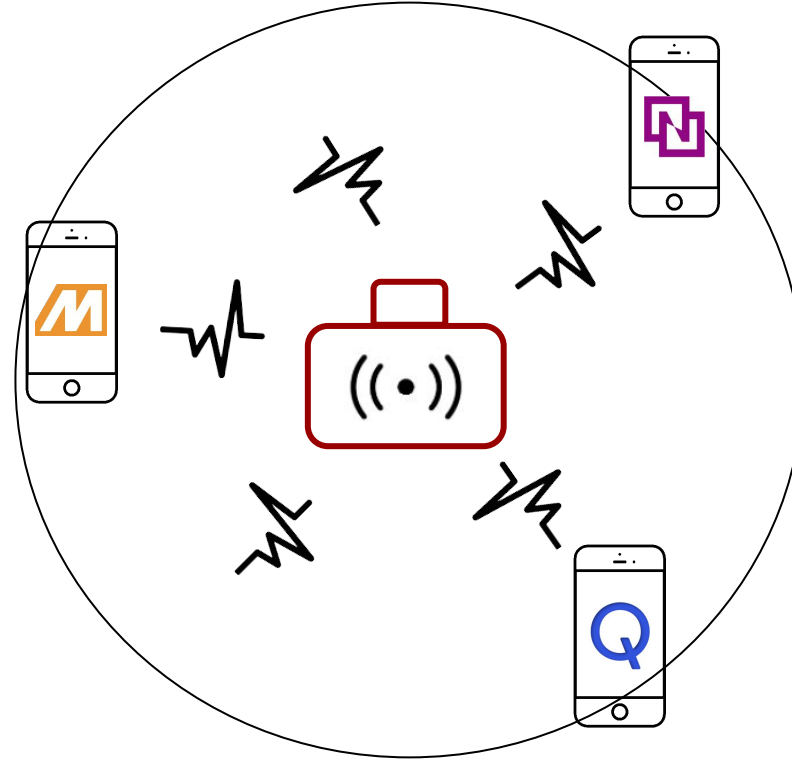
...

**Why is the cyber-army researching
Unisoc?**

~~Targeted attack~~

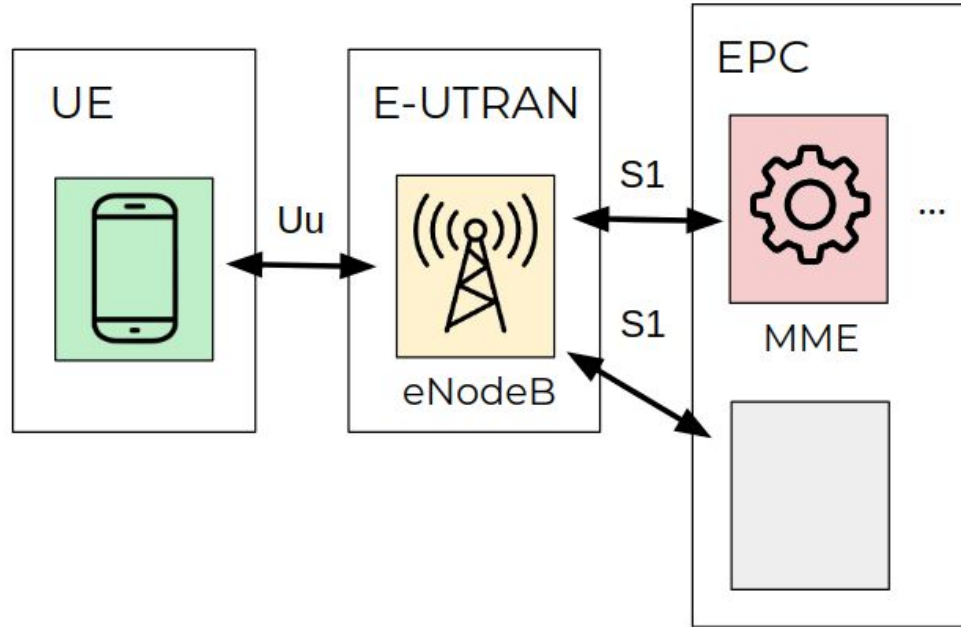


Broadcast to neutralize communication

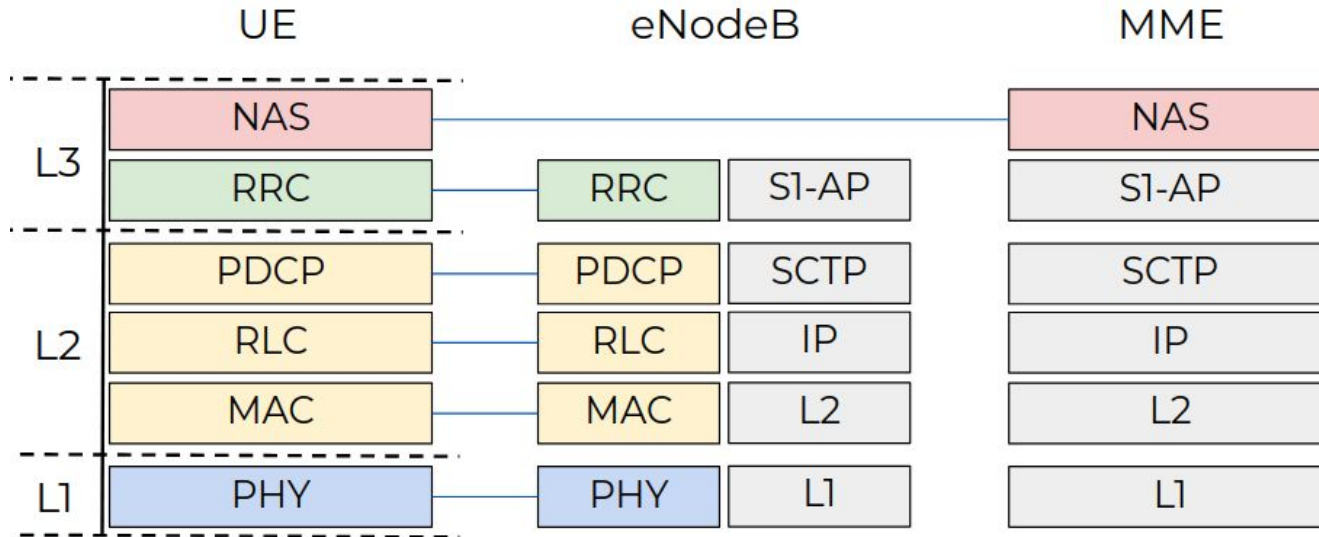


**Where is the smartphone modem
in the LTE network?**

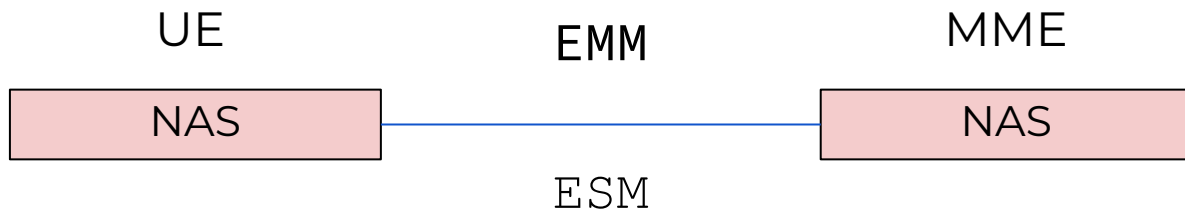
Long-Term Evolution network



LTE protocol stacks



Non-access stratum



- security control
- tracking area management
- mobility between access networks
- bearer management
- ...

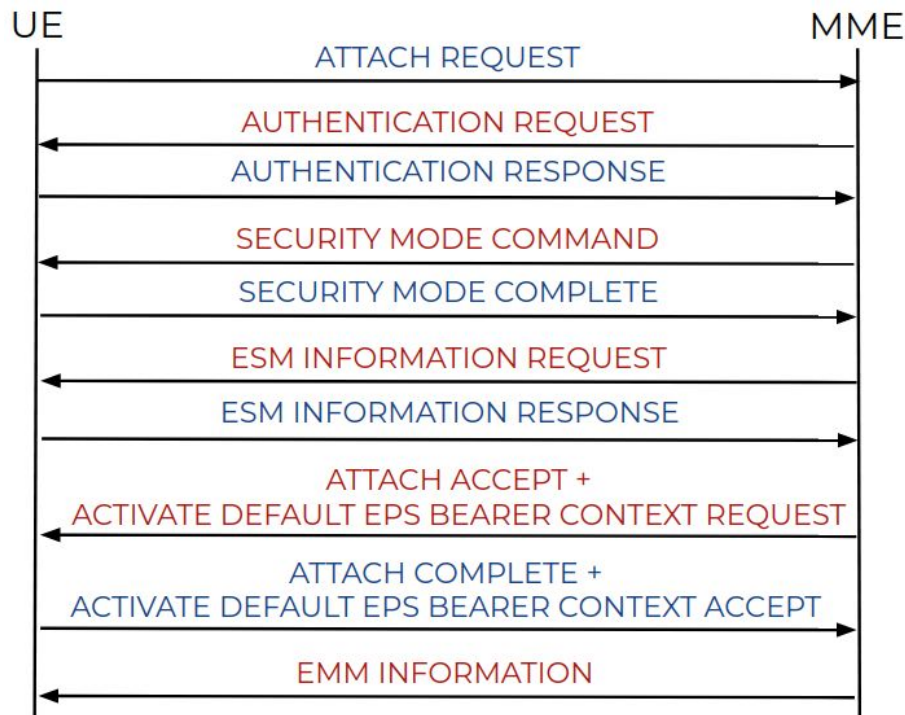
Network attach flow

Modem: 10.90.10.1

MME: 10.201.150.41

Source	Destination	Protocol	Length	Info
10.90.10.1	10.201.150.41	S1AP/NAS-EPS	176	InitialUEMessage, Attach request, PDN connectivity request
10.201.150.41	10.90.10.1	S1AP/NAS-EPS	140	DownlinkNASTransport, Authentication request
10.90.10.1	10.201.150.41	S1AP/NAS-EPS	140	UplinkNASTransport, Authentication response
10.201.150.41	10.90.10.1	S1AP/NAS-EPS	124	DownlinkNASTransport, Security mode command
10.90.10.1	10.201.150.41	S1AP/NAS-EPS	148	UplinkNASTransport, Security mode complete
10.201.150.41	10.90.10.1	S1AP/NAS-EPS	116	DownlinkNASTransport, ESM information request
10.90.10.1	10.201.150.41	S1AP/NAS-EPS	176	UplinkNASTransport, ESM information response
10.201.150.41	10.90.10.1	S1AP/NAS-EPS	288	InitialContextSetupRequest, Attach accept, Activate default EPS bearer context request
10.90.10.1	10.201.150.41	S1AP/NAS-EPS	124	UplinkNASTransport, Attach complete, Activate default EPS bearer context accept
10.201.150.41	10.90.10.1	S1AP/NAS-EPS	140	DownlinkNASTransport, EMM information
10.90.10.1	10.201.150.41	S1AP/NAS-EPS	132	UplinkNASTransport, Detach request (Combined EPS/IMSI detach / switch-off)

Network attach flow



Open-source UE stack

srsRAN



```
switch (msg_type) {
case LIBLTE_MME_MSG_TYPE_ATTACH_ACCEPT:
    parse_attach_accept(lcid, std::move(pdu));
    break;
case LIBLTE_MME_MSG_TYPE_AUTHENTICATION_REQUEST:
    parse_authentication_request(lcid, std::move(pdu), sec_hdr_type);
    break;
case LIBLTE_MME_MSG_TYPE_SECURITY_MODE_COMMAND:
    parse_security_mode_command(lcid, std::move(pdu));
    break;
case LIBLTE_MME_MSG_TYPE_ESM_INFORMATION_REQUEST:
    parse_esm_information_request(lcid, std::move(pdu));
    break;
...
}
```

“Attach accept” message

▼ Non-Access-Stratum (NAS)PDU

0010 = Security header type: Integrity protected and ciphered (2)
.... 0111 = Protocol discriminator: EPS mobility management messages (0x7)
Message authentication code: 0xa9c0b9f0
Sequence number: 2
0000 = Security header type: Plain NAS message, not security protected (0)
.... 0111 = Protocol discriminator: EPS mobility management messages (0x7)
NAS EPS Mobility Management Message Type: Attach accept (0x42)
0000 = Spare half octet: 0
.... 0... = Spare bit(s): 0x00
.... .010 = Attach result: Combined EPS/IMSI attach (2)

- GPRS Timer - T3412 value
- Tracking area identity list - TAI list
- ESM message container
- EPS mobile identity - GUTI
- GPRS Timer - T3423 value
- EPS network feature support

- list of tracking area identifiers
- GPRS timers
- access point name
- mobile identity
- emergency numbers
- protocol configuration options


...

A vulnerability in the NAS parser

liblte_mme_unpack_emergency_number_list_ie()

```
uint32 sent_length, length, idx;  
sent_length = (*ie_ptr)[0] + 1;  
length = 1;  
emerg_num_list->N_emerg_nums = 0;  
while (length < sent_length) {  
    idx = emerg_num_list->N_emerg_nums;  
    emerg_num_list->emerg_num[idx].N_emerg_num_digits = ((*ie_ptr)[length++] - 1) * 2;  
    ...  
    emerg_num_list->N_emerg_nums++;  
}
```

We control the length and
contents of the message




































The max length of the
emergency list is 12



**Looking for the NAS parsing functions
in the Unisoc modem**

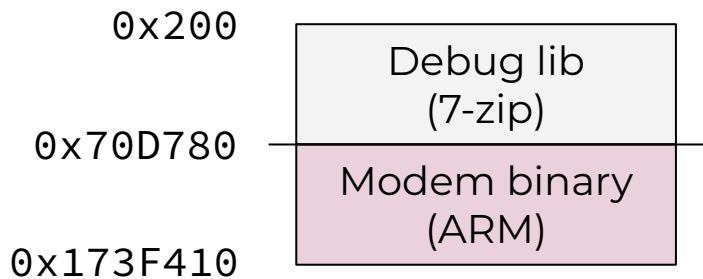
Moto G20 [Unisoc T700]

 boot.img	 prodnv.img	 super.img
 cache.img	 SC9600_sharkl5pro_pubcp_modem.dat	 teecfg-sign.bin
 dtbo.img	 sharkl5pro_cm4.bin	 tos-sign.bin
 EXEC_KERNEL_IMAGE-sign.bin	 sharkl5pro_pubcp_AGCP_DSP.bin	 u-boot-sign.bin
 fdl1-sign.bin	 sharkl5pro_pubcp_CDMA_DSP.bin	 u-boot-spl-16k-sign.bin
 fdl2-sign.bin	 sharkl5pro_pubcp_customer_deltanv.bin	 userdata.img
 gnssmodem-sign.bin	 sharkl5pro_pubcp_customer_nvitem.bin	 vbmeta_product.img
 logo.bin	 sharkl5pro_pubcp_DM_DSP.bin	 vbmeta-sign.img
 odmko.img	 sharkl5pro_pubcp_LTEA_DSP.bin	 vbmeta_system.img
 p352.xml	 sml-sign.bin	 vbmeta_system_ext.img
 persist.img	 socko.img	 vbmeta_vendor.img

Modem image

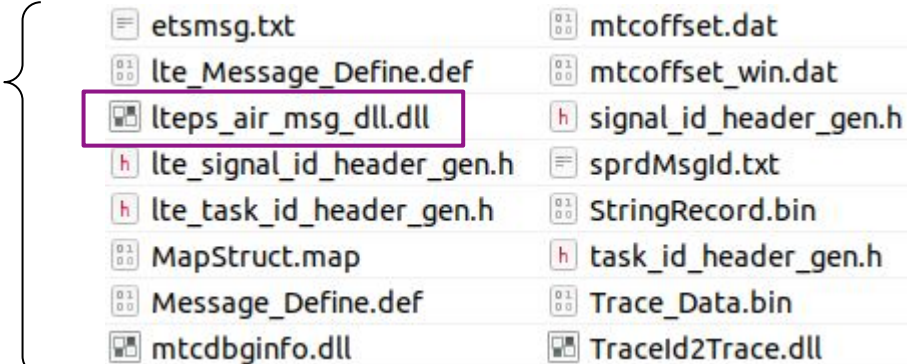
```
struct data_block_header_t {  
    uint32_t type;  
    uint32_t offset;  
    uint32_t length;  
};
```

















	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	53	43	49	31	00	00	00	00	00	00	00	00	02	04	00	00	SCI1.....
0010h:	00	02	00	00	77	D5	70	00	01	03	00	00	80	D7	70	00	...w0p...€xp.
0020h:	90	1C	03	01	00	00	00	00	00	00	00	00	00	00	00	00



Debug lib?









Debug lib
(7-zip)



 etsmg.txt	 mtcoffset.dat
 lte_Message_Define.def	 mtcoffset_win.dat
 lteps_air_msg_dll.dll	 signal_id_header_gen.h
 lte_signal_id_header_gen.h	 sprdMsgId.txt
 lte_task_id_header_gen.h	 StringRecord.bin
 MapStruct.map	 task_id_header_gen.h
 Message_Define.def	 Trace_Data.bin
 mtcdbginf.dll	 Traceld2Trace.dll

Do you also see these x86 DLLs on the ARM chip?!

lteps_air_msg_dll.dll

 LNAS_airMsgEmm_attachAcc_decode	1001DE33
 LNAS_airMsgEmm_attachAcc_encode	1001D7C0
 LNAS_airMsgEmm_attachCmpl_decode	1001F1B6
 LNAS_airMsgEmm_attachCmpl_encode	1001F0F0
 LNAS_airMsgEmm_attachRej_decode	1001F477
 LNAS_airMsgEmm_attachRej_encode	1001F260
 LNAS_airMsgEmm_attachReq_decode	10020493
 LNAS_airMsgEmm_attachReq_encode	1001FBC0

 LNAS

Line 186 of 186

```
typedef int (__stdcall *LNAS_airMsgIE_mobileId_decode)(void* out, void* in, void* offset);
```

Modem binary

Modem
binary

Base address

Header length

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	4D	45	43	50	56	31	2E	30	00	06	00	00	00	00	00	00	MECPV1.0.....
0010h:	73	69	70	63	2D	6D	65	6D	00	00	00	00	00	00	00	00	sipc-mem.....
0020h:	00	00	00	00	00	00	00	00	00	00	80	87	00	00	00	00€†
0030h:	00	00	80	00	00	00	00	00	00	00	00	00	00	00	00	00	..€.....
0040h:	63	70	2D	6D	6F	64	65	6D	00	00	00	00	00	00	00	00	cp-modem.....
0190h:	04	F0	1F	E5	00	06	00	8B	00	00	00	00	00	00	00	00	.ä.ä...<.....
01A0h:	6D	6F	64	65	6D	00	00	00	00	00	00	00	00	00	00	00	modem.....
01B0h:	00	00	00	00	00	00	00	00	00	00	00	8B	00	00	00	00<....
01C0h:	00	00	38	01	00	00	00	00	01	00	00	00	00	00	00	00	..8.....
01D0h:	63	70	63	6D	64	6C	69	6E	65	00	00	00	00	00	00	00	cpcmdline.....
01E0h:	00	00	00	00	00	00	00	00	00	00	F2	8A	00	00	00	00öŠ.....
01F0h:	00	08	00	00	00	00	00	00	02	00	00	00	00	00	00	00
0600h:	00	F0	9F	E5	00	F0	9F	E5	58	27	00	8B	00	00	00	00	.äYä.äYäX'.<....
0610h:	0E	40	2D	E9	FF	5F	20	E9	00	10	0F	E1	0E	20	A0	E1	._@-éÿ_ é...ä. ä
0620h:	38	30	8D	E2	0E	00	20	E9	0E	80	BD	E8	1F	B5	9D	E8	80.ä... é.€%è.µ.è

The code










Modem code

- For convenience, we:
 - wrapped the code blob with an ELF32 header
 - added the data segment
- Not compressed/encrypted
- No symbols
- There are many strings

Name	Start	End
 .text	8B000600	8C031C90
 .data	8D0C0000	8DF00000



Where are the NAS handlers?

	.text:8B1A1670	0000003E	C	PS/stack/nas/emm/msg_codec/msg_emm/lnas_air_msg_emm_att_acc.c
	.text:8B1A1984	0000003F	C	PS/stack/nas/emm/msg_codec/msg_emm/lnas_air_msg_emm_att_cmpl.c
	.text:8B1A1E38	0000003E	C	PS/stack/nas/emm/msg_codec/msg_emm/lnas_air_msg_emm_att_req.c
	.text:8B1A26A8	0000003E	C	PS/stack/nas/emm/msg_codec/msg_emm/lnas_air_msg_emm_det_acc.c
	.text:8B1A6944	0000003E	C	PS/stack/nas/emm/msg_codec/msg_emm/lnas_air_msg_emm_tau_acc.c
	.text:8B1A6CAC	0000003F	C	PS/stack/nas/emm/msg_codec/msg_emm/lnas_air_msg_emm_tau_cmpl.c
	.text:8B1A714C	0000003E	C	PS/stack/nas/emm/msg_codec/msg_emm/lnas_air_msg_emm_tau_rej.c
	.text:8B1A7644	0000003E	C	PS/stack/nas/emm/msg_codec/msg_emm/lnas_air_msg_emm_tau_req.c
	lnas_air_msg			

Line 1 of 62

```
v7 = _malloc(0, 724, 1, 140, 648);
if ( !v7 )
    _loge(
        0,
        "PS/stack/nas/emm/src/control/adm/adm.c",
        649,
        (int)"LOGGER NULL POINTER",
        (int)"attachAccept_Ptr"
    );
v9 = _liblte_mme_unpack_attach_accept_msg((int)v7, *(char **)(a1 + 0x18), (unsigned int *) (a1 + 0x1C));
```

srsRAN vs Unisoc

srsRAN

```
#define LIBLTE_MME_MS_IDENTITY_IEI 0x23

LIBLTE_ERROR_ENUM liblte_mme_unpack_attach_accept_msg (LIBLTE_BYTE_MSG_STRUCT* msg,
LIBLTE_MME_ATTACH_ACCEPT_MSG_STRUCT* attach_accept) {
    uint8* msg_ptr = msg->msg;

    if (LIBLTE_MME_MS_IDENTITY_IEI == *msg_ptr) {
        msg_ptr++;
        liblte_mme_unpack_mobile_id_ie(&msg_ptr, &attach_accept->ms_id);
    }
}
```

Modem

```
int __fastcall _liblte_mme_unpack_attach_accept_msg(int a1, char *msg_buf, unsigned int *a3)
{
    v18 = read_from_msg((int)msg_buf, &offset, 8);
    if ( v18 == 0x23 )
    {
        _liblte_mme_unpack_mobile_id_ie(v62, (int)msg_buf, &offset);
    }
}
```

srsRAN vs Unisoc

srsRAN

```
LIBLTE_ERROR_ENUM liblte_mme_unpack_mobile_id_ie (uint8** ie_ptr, LIBLTE_MME_MOBILE_ID_STRUCT* mobile_id) {  
    uint32 length;  
  
    length = **ie_ptr;  
    *ie_ptr += 1;  
  
    mobile_id->type_of_id = **ie_ptr & 0x07;  
}
```

Modem

```
void __fastcall _liblte_mme_unpack_mobile_id_ie(int *a1, int msg_buf, int *a3)  
{  
    offset[0] = *a3;  
    id_len = read_from_msg(msg_buf, offset, 8);  
    if ( (unsigned int)(id_len - 1) < 9 )  
    {  
        offset[0] += 5;  
        id_type = read_from_msg(msg_buf, offset, 3);  
    }  
}
```

Looking for vulnerabilities

The unified parsing function format

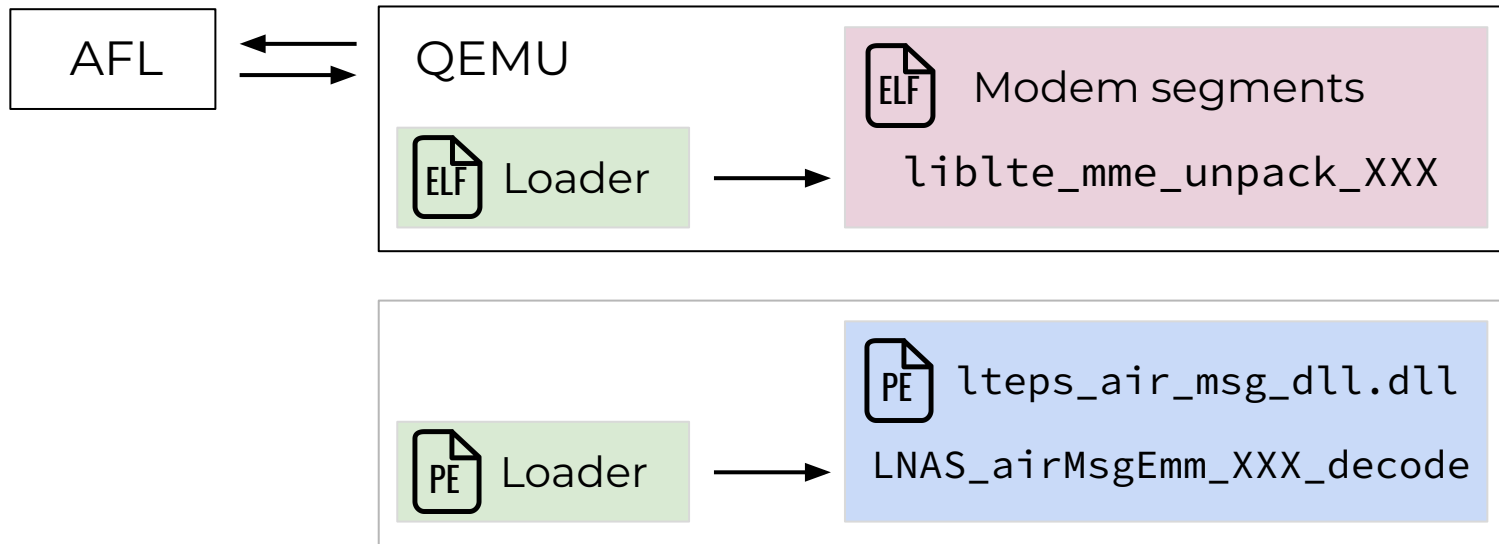
```
int __fastcall _liblte_mme_unpack_tracking_area_identity_list_ie(  
    unsigned __int8 *out,  
    unsigned __int8 *in,  
    int *offset)
```

qemu-arm

out	an output buffer	malloc(XXX)	Redirect + modem's malloc to libc::malloc
in	the message blob	Test data	
offset	the current offset in the in	0	

Fuzzing

The classic combo: AFL + QEMU



Mobile identity

liblte_mme_unpack_attach_accept_msg, ...



liblte_mme_unpack_mobile_id_ie

IMSI	313 460 000 000 001
IMEI	35 161508 297879 5
IMEISV, TMSI, TMGI, ...	

Malformed mobile ID

“Attach accept” message

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000h:	07	0F	81	00	0D	03	0D	2F	00	23	00	F5	07	00	07	00
0010h:	00	00	00	21	00	00	00	00	00	00	10	FF	E1	00	00	19
0020h:	00	B9	00	00	FC	E0	02	EF	FF	05	08	FC	FE	0F	81	00
0030h:	0D	03	0D	2F	00	23	01	01	00	00	00	00	00	00	00	00

Mobile ID
field

ID length

ID type

```
void __fastcall _liblte_mme_unpack_mobile_id_ie(int *a1, int msg_buf, int *a3)
{
    offset[0] = *a3;
    id_len = read_from_msg(msg_buf, offset, 8); // The ID length is 1
    if ( (id_len - 1) < 9 )
    {
        offset[0] += 5;
        id_type = read_from_msg(msg_buf, offset, 3); // The ID type is 1 (IMSI)
        *a1 = id_type;
        switch ( *a1 )
        {
            case 1:
                offset[0] -= 8;
                parse_imsi(a1 + 4, msg_buf, offset, id_len); // This is IMSI parser
            default:
                break;
        }
    }
}
```


Integer underflow

The length-2 bytes of the message data are copied to the output buffer
What if the length is 0 or 1?

```
void __fastcall parse_imsi(unsigned __int8 *out_buf, int msg_buf,
{
    int *offset_, int imsi_data_len)
    offset[0] = *offset_;
    out_buf[1] = read_from_msg(msg_buf, offset, 4); // 0
    v1 = read_from_msg(msg_buf, offset, 1); // 0
    offset[0] += 3;
    v2 = 2 * imsi_data_len - 2; // 2 * 1 - 2 = 0
    v3 = v2 + (v1 == 1); // 0
    *out_buf = v3;
    if ( v1 == 1 ) // 0 - 1 = 0xFFFFFFFF
        v4 = v3 - 1;
    else
        v4 = v3 - 2; // 0 - 2 = 0xFFFFFFFFE
    copy((out_buf + 2), msg_buf, offset, v4); // the overflow
```

What have we achieved?

- For the first time, we looked at the Unisoc baseband as an attack target
 - We discovered DoS vulnerabilities in the NAS protocol
 - 5G Unisoc devices are also affected
-
- Unisoc fixed the CVE-2022-20210 (9.4) in May 2022
 - The open source srsRAN is still vulnerable

Thank you!



slavam@checkpoint.com

@_cpresearch_
research.checkpoint.com