A SIM Hacking Odyssey

Can a SIM Hack YOU?

TROOPERS 2026

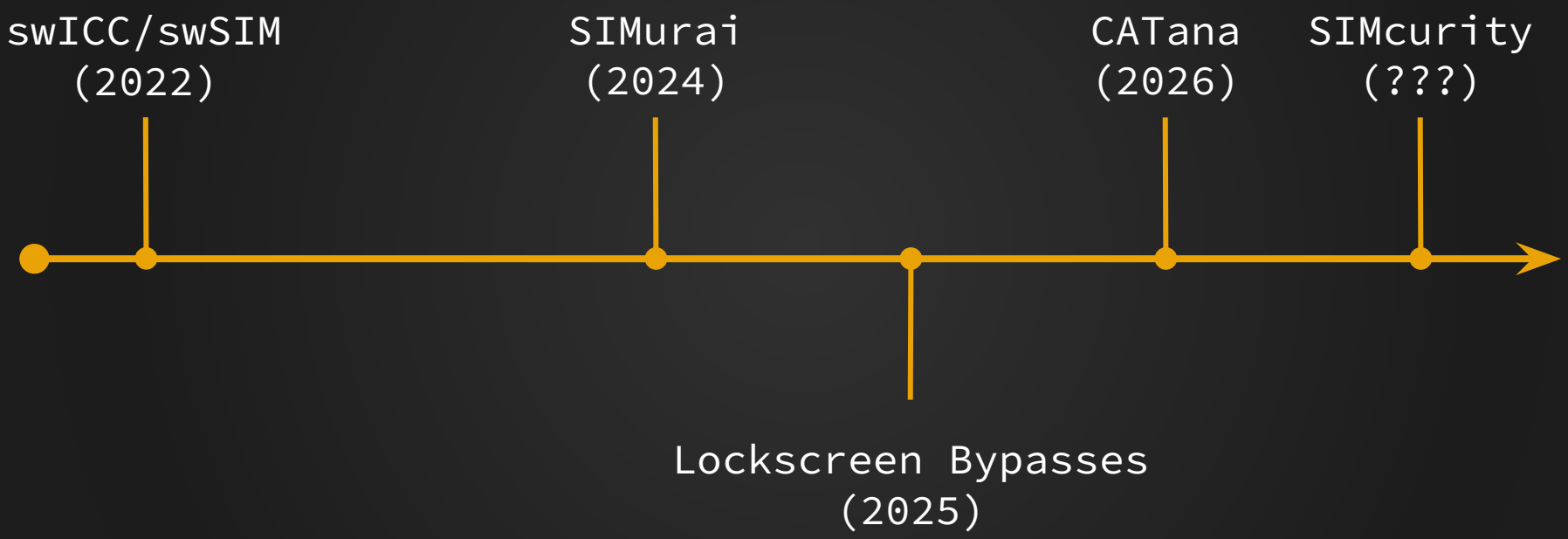
Who are We?

Tomasz Lisowski

- PhD + Demoscene
- Low level hard/software
- Cellular Security

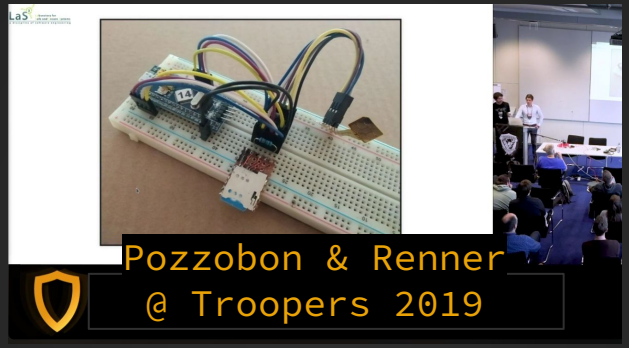
Marius Muench

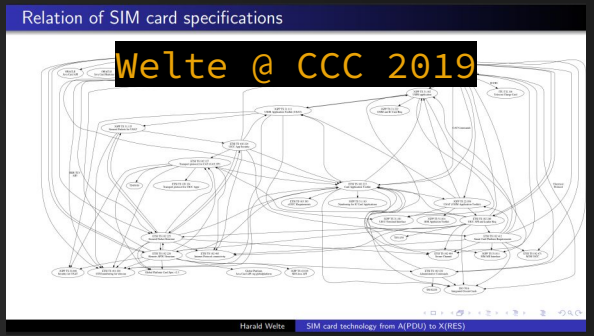
- Assistant Prof @ University of Birmingham, UK
- Rehosting, Fuzzing, Binary Analysis, Cellular Security


swICC/swSIM
(2022)
SIMurai
(2024)
CATana
(2026)
SIMcurity
(???)
Lockscreen Bypasses
(2025)

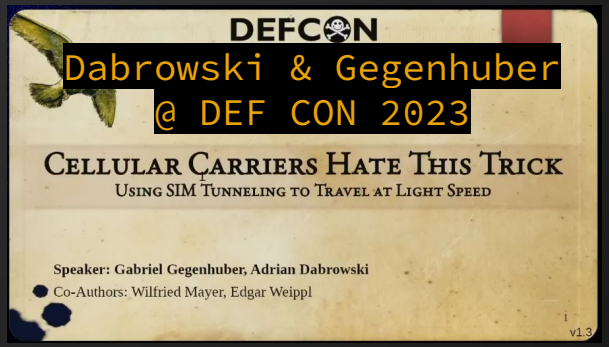
Prior Art


Koscher & Butler
@ DEF CON 2013
Call Waiting
Show My Caller ID
TTY
OFF
SIM PIN
SIM Applications
AT&T Services


Pozzobon & Renner
@ Troopers 2019


Relation of SIM card specifications
Welte @ CCC 2019
Harald Welte
SIM card technology from A(PDU) to X(RES)

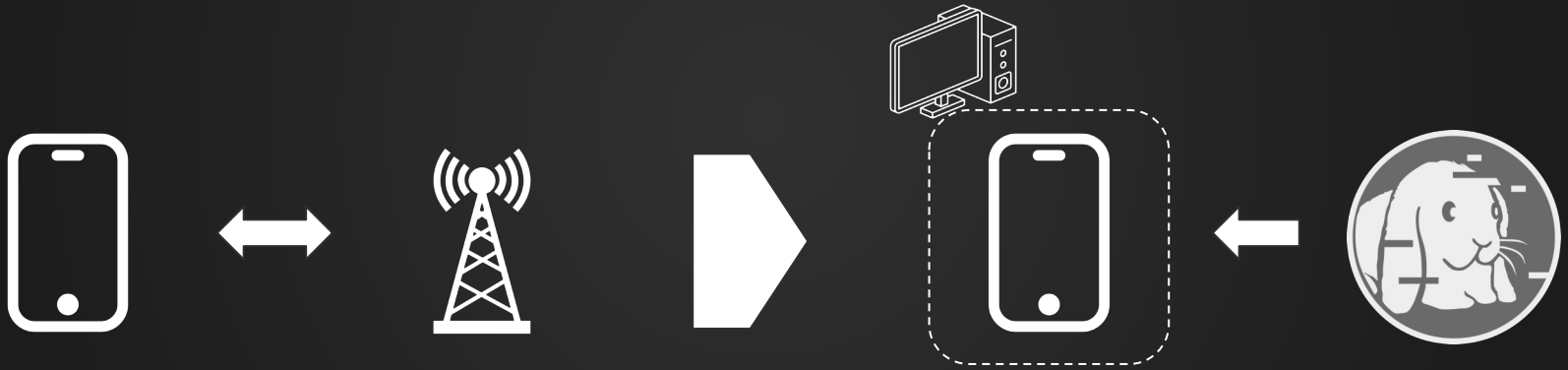

Burgess @
DEEPSEC 2021
Proactive SIMs
Nov 2021
David Allen Burgess
@dburgess00
david.allen.burgess@gmail.com
DEEPSEC


DEFCON
Dabrowski & Gegenhuber
@ DEF CON 2023
Cellular Carriers Hate This Trick
Using SIM Tunneling to Travel at Light Speed
Speaker: Gabriel Gegenhuber, Adrian Dabrowski
Co-Authors: Wilfried Mayer, Edgar Weippl
v1.3

Chapter 0:
We Want to Emulate a SIM

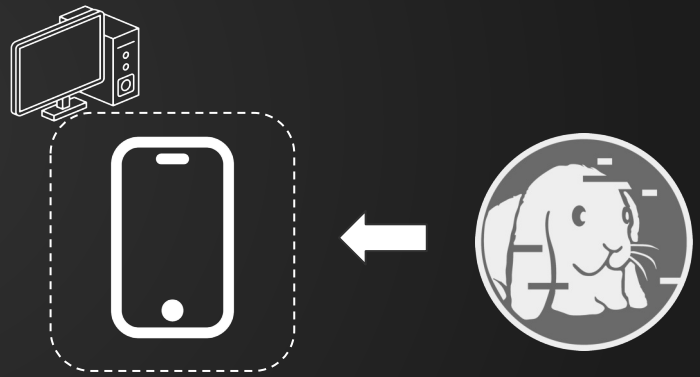
Where all of this began.

2022: Upcoming Release of FirmWire



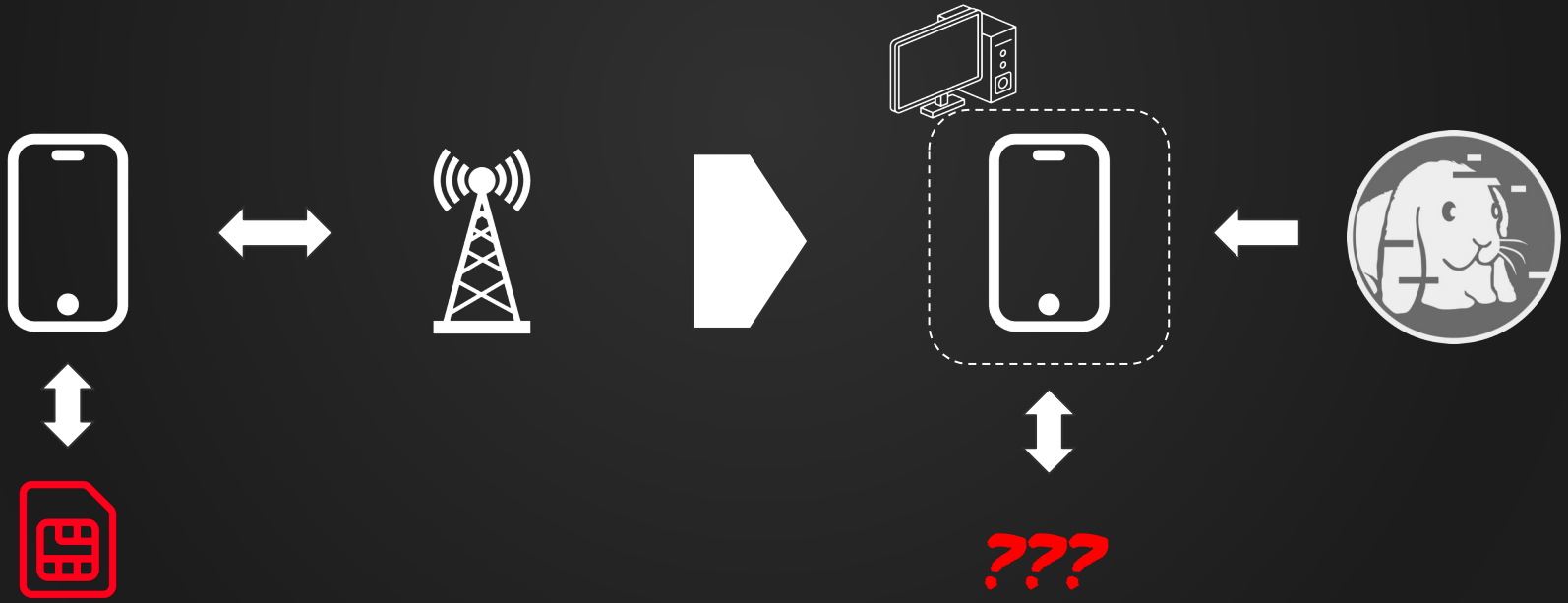
2022: Upcoming Release of FirmWire

```
[digital ~/code/FirmWire >>
[digital ~/code/FirmWire >>
[digital ~/code/FirmWire >> ./d ./firmwire.py --fuzz-triage gsm_cc --fuzz-input afl/crashes/GSM
_CC_Bearer_Crash_MEM_GUARD_G973F_ASG8.bin --restore crash CP_G973FXXU3ASG8_CP13372649_CL1648796
3_QB24948473_REV01_user_low_ship.tar.md5.lz4
```

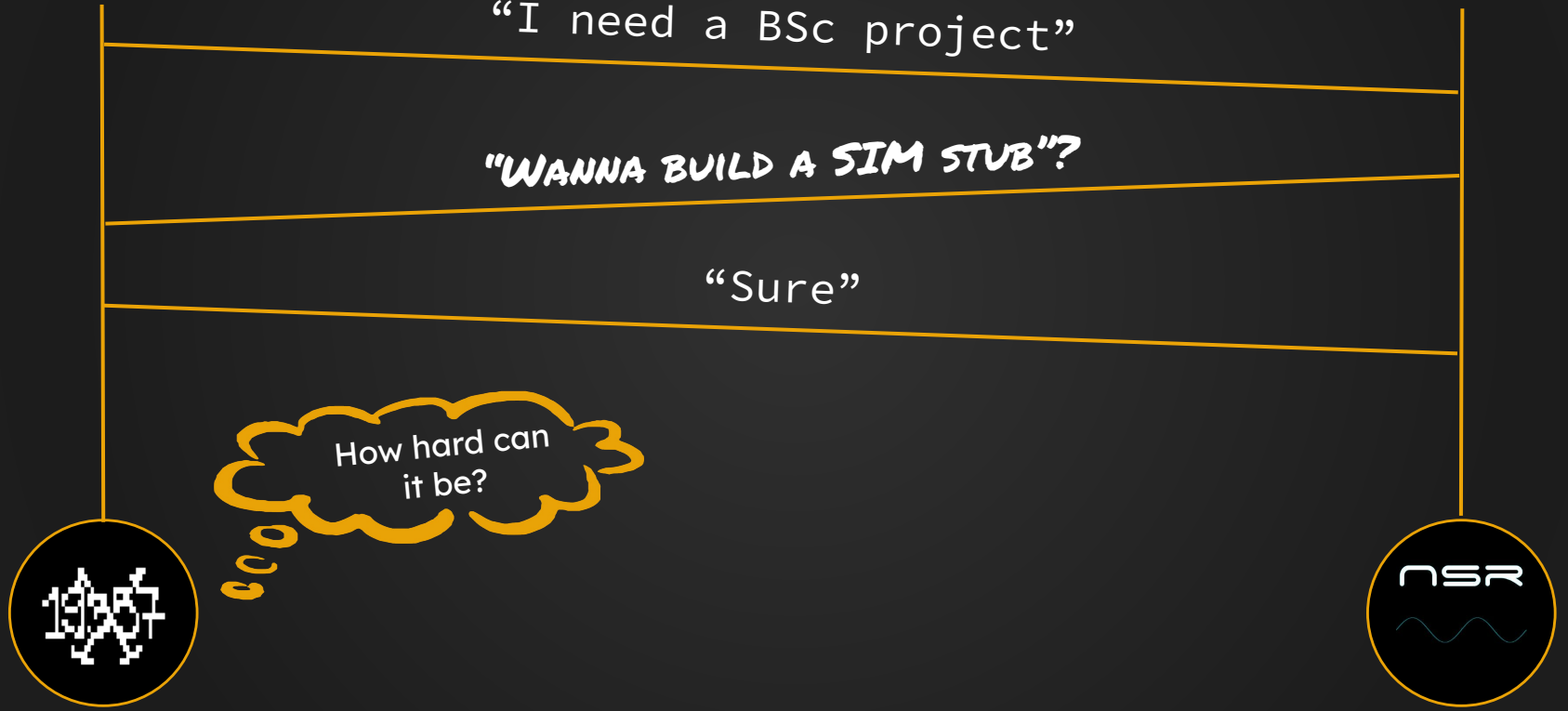


https://github.com/FirmWire/FirmWire

2022: Upcoming Release of FirmWire



First Contact




Always has been...
So a SIM card is just a Smart Card?

4.3 Mechanical strength (of a card and contacts)

The card should resist damage to its surface and to any components contained in it and should remain intact during normal use, storage and handling.

Each contact surface and contact area (entire galvanic surface) shall not be damaged by a working pressure equivalent to a steel ball of diameter 1 mm applying a force of 1,5 N.

4.4 Electrical resistance (of contacts)

The contact resistance of a card connector assembly should be sufficiently low to enable a good contact between card and reader contacts.

When a d.c. current of any value between 50 µA and 100 mA is applied, the surface resistance between any two points on the same contact pad shall not exceed 0,5 Ω at a distance of 1,5 mm between contact points. The contact pad area is defined in ISO/IEC 7816-2.

4.5 Electromagnetic interference (between magnetic stripe and integrated circuit)

If the card carries a magnetic stripe, the IC card shall not be damaged, malfunction or be altered after reading, writing or erasing of the magnetic stripe. Conversely, the writing or reading of the IC shall not cause a malfunction of the magnetic stripe or its associated reading, writing or handling mechanisms.

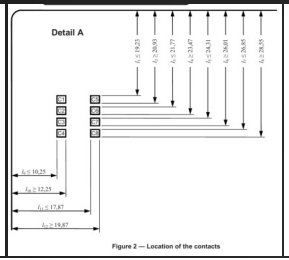


Figure 2 — Location of the contacts

— RST shall be put to state L, see 5.2.2.

— VCC shall be powered, see 5.2.1.

— I/O in the interface device shall be put in reception mode, see 5.2.5. The interface device shall ignore the state on I/O during activation.

— CLK shall be provided with a clock signal, see 5.2.3.

NOTE 1 The delays between powering VCC, setting I/O in reception mode and providing the clock signal on CLK are not defined.

NOTE 2 The interface device may perform a deactivation due to short circuits.

Figure 1 summarizes activation (before time T_A) and cold reset (after time T_A).

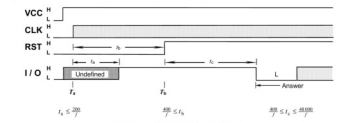


Figure 1 — Activation and cold reset

N_c denotes the number of bytes in the command data field. The L_c field encodes N_c.

— If the L_c field is absent, then N_c is zero.

— A short L_c field consists of one byte not set to '00'. From '01' to 'FF', the byte encodes N_c from one to 255.

— An extended L_c field consists of three bytes: one byte set to '00' followed by two bytes not set to '0000'. From '0001' to 'FFFF', the two bytes encode N_c from one to 65 535.

Table 1 — Command-response pair (C-RP)

Field	Description	Number of bytes	Direction
Command header	Class byte denoted CLA	1	To the card
	Instruction byte denoted INS	1	
	Parameter bytes denoted P1-P2	2	
L_c field	Absent for encoding $N_c = 0$, present for encoding $N_c > 0$	0, 1 or 3	
Command data field	Absent if $N_c = 0$, present as a string of N_c bytes if $N_c > 0$	N_c	
L_e field	Absent for encoding $N_e = 0$, present for encoding $N_e > 0$	0, 1, 2 or 3	
Response data field	Absent if $N_r = 0$, present as a string of N_r bytes if $N_r > 0$	N_r (at most N_e)	From the card
Response trailer	Status bytes denoted SW1-SW2	2	

N_r denotes the maximum number of bytes expected in the response data field, regardless of any structure of the data within this field. The L_e field encodes N_e.

— If the L_e field is absent, then N_e is zero.

— A short L_e field consists of one byte with any value.

— From '01' to 'FF', the byte encodes N_e from one to 255.

12.2.2 Case 1

The command APDU is mapped onto the command TPDU with P3 = '00'.

Command APDU	CLA INS P1 P2
Command TPDU	CLA INS P1 P2 {P3 = '00'}

The response TPDU is mapped onto the response APDU without any change.

Response TPDU	SW1 SW2
Response APDU	SW1 SW2

12.2.3 Case 2S

The short L_e field consists of one byte: C(5) with any value for encoding N_e from 1 to 256 ('00' means the maximum, 256). The command APDU is mapped onto the command TPDU without any change.

Command APDU	CLA INS P1 P2	{L_e field = C(5)}
Command TPDU	CLA INS P1 P2 {P3 = C(5)}	

The response TPDU is mapped onto the response APDU according to the acceptance of N_e and according to the processing of the command.

— Data object (DO):

The DOs store data. A DO may be the parent of other structures, whose types shall belong to the following set {DO}. These other structures are said to be immediately under the DO.

Two types of logical organization are provided.

— Figure 3 illustrates a hierarchy of DFs with its corresponding security architecture (see Clause 9). In such a card organization, the DF at the root is called the master file (MF); any DF may be an application DF, with or without its own hierarchy of DFs.

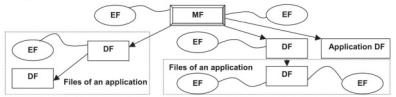


Figure 3 — Example of hierarchy of DFs

— Figure 4 illustrates application DFs in parallel, with no MF seen at the interface, i.e. without any apparent hierarchy of DFs. Such an organization supports independent applications in the card

Part 7: Interindustry commands for Structured Cards Query Language (SCQL)

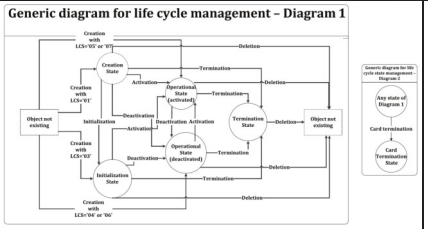
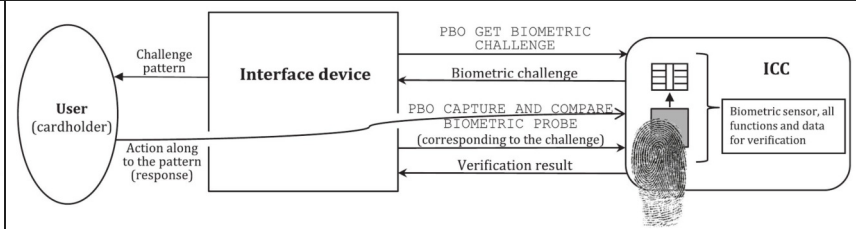


Figure 1 — Generic diagram for life cycle status



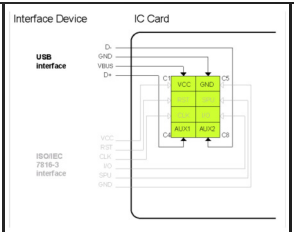


Figure 1 — Assignment of contacts for a USB integrated circuit card

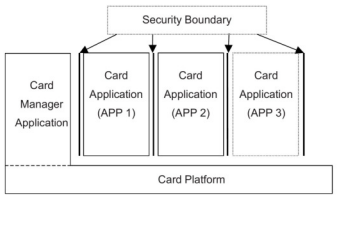
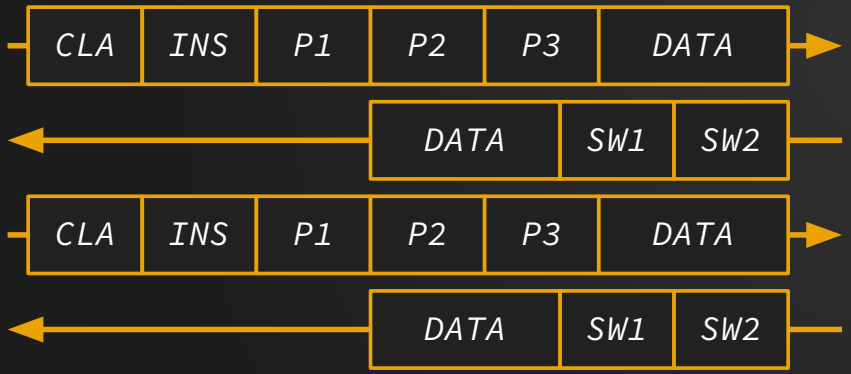


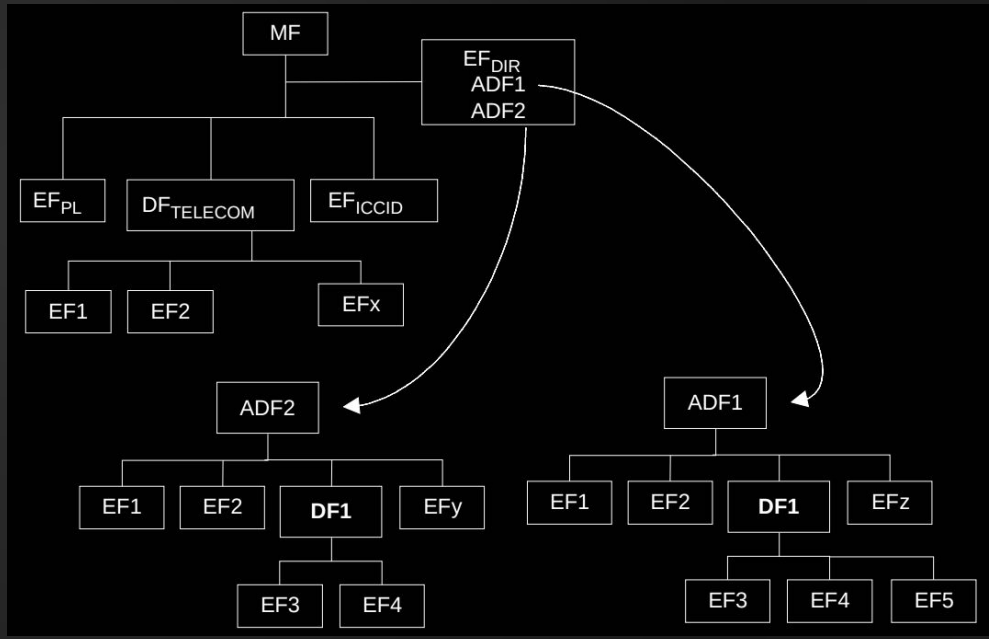
Figure 1 — Possible structure of a multi-application card

swICC - Smart Card Emulator

Commands



File System



swSIM - SIM Emulator

- SIM is ~superset of the ICC

- More command classes (CLA)
- More commands (INS)
- Different behaviour of some existing commands
- Authentication: **Milenage**, **COMP128**

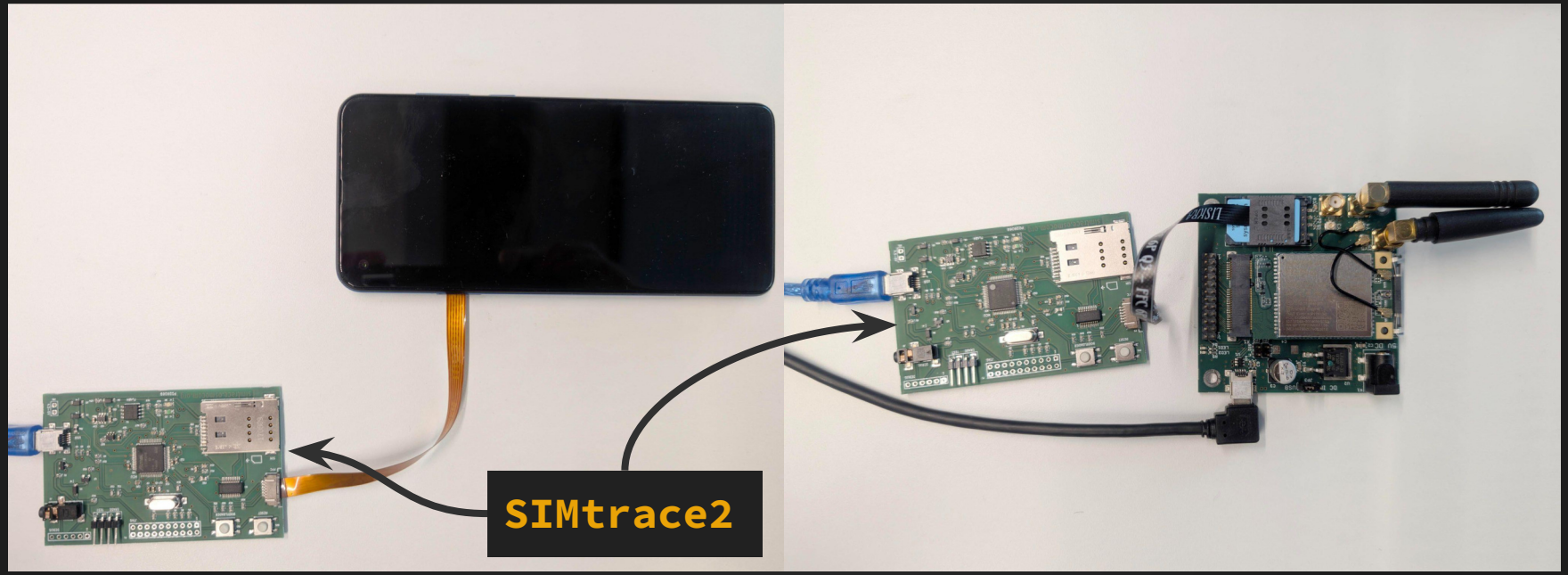
- Card **Application Toolkit** (CAT)

- **USIM “Application”**

COMMAND	CLA	INS
Command APDUs		
SELECT FILE	'0X' or '4X' or '6X'	'A4'
STATUS	'8X' or 'CX' or 'EX'	'F2'
READ BINARY	'0X' or '4X' or '6X'	'B0'
UPDATE BINARY	'0X' or '4X' or '6X'	'D6'
READ RECORD	'0X' or '4X' or '6X'	'B2'
UPDATE RECORD	'0X' or '4X' or '6X'	'DC'
SEARCH RECORD	'0X' or '4X' or '6X'	'A2'
INCREASE	'8X' or 'CX' or 'EX'	'32'
RETRIEVE DATA	'8X' or 'CX' or 'EX'	'CB'
SET DATA	'8X' or 'CX' or 'EX'	'DB'
VERIFY PIN	'0X' or '4X' or '6X'	'20'
CHANGE PIN	'0X' or '4X' or '6X'	'24'
DISABLE PIN	'0X' or '4X' or '6X'	'26'
ENABLE PIN	'0X' or '4X' or '6X'	'28'
UNBLOCK PIN	'0X' or '4X' or '6X'	'2C'
DEACTIVATE FILE	'0X' or '4X' or '6X'	'04'
ACTIVATE FILE	'0X' or '4X' or '6X'	'44'
AUTHENTICATE	'0X' or '4X' or '6X'	'88', '89'
GET CHALLENGE	'0X' or '4X' or '6X'	'84'
TERMINAL CAPABILITY	'8X' or 'CX' or 'EX'	'AA'
TERMINAL PROFILE	'80'	'10'
ENVELOPE	'80'	'C2'
FETCH	'80'	'12'
TERMINAL RESPONSE	'80'	'14'
MANAGE CHANNEL	'0X' or '4X' or '6X'	'70'
MANAGE SECURE CHANNEL	'0X' or '4X' or '6X'	'73'
TRANSACT DATA	'0X' or '4X' or '6X'	'75'
SUSPEND UICC	'80'	'76'

COMMAND	CLA	INS
Command APDUs		
GET IDENTITY	'8X' or 'CX' or 'EX'	'78' (see note)
EXCHANGE CAPABILITIES	'80'	'7A' (see note)
MANAGE LSI	'80'	'7C'
Transmission oriented APDUs applying to the above commands		
GET RESPONSE	'0X' or '4X' or '6X'	'C0'
NOTE: These INS values are also used by GlobalPlatform (for the commands END R-MAC SESSION and BEGIN R-MAC SESSION, see [i.1] and [i.2]). See also note 2 below.		

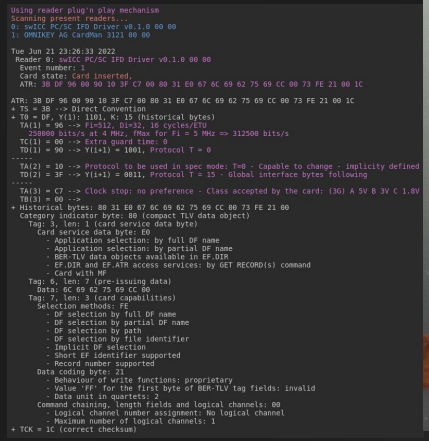
Dev Setup



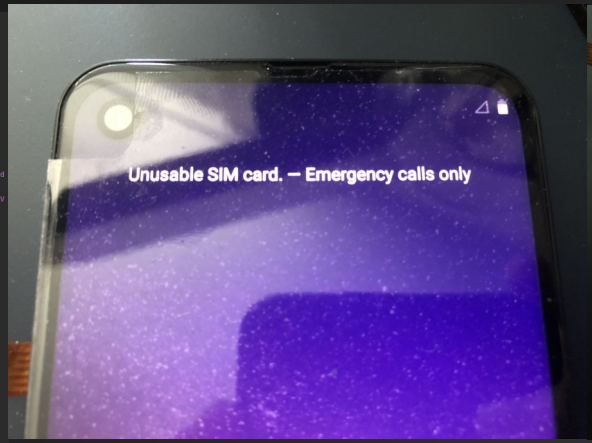
```
Using default server IP: '127.0.0.1'.
swSIM:
  swICC FS at './tmp.swiccfs'.
  FS JSON at  './data/usim.json'.
  Connect to   127.0.0.1:55556.

Tree: Allocated more memory, retrying.
Press ctrl-c to exit.
RX:
(Message
    (Header (Size 9))
    (Data
        (Control 0x02)
        (Cont 0x00000000
        (BufLenExp 0)
        (Buf [ ])))
TX:
(Message
    (Header (Size 34))
    (Data
        (Control 0xF0)
        (Cont 0x00006000)
        (BufLenExp 0)
        (Buf [ 3B DF 96 00 90 10 3F 07 00 80 31 E0 67 73 77 69 63 63 00 00 73 FE 21 00 06 ])))
RX:
(Message
    (Header (Size 14))
    (Data
        (Control 0x00)
```

Demo 0

Progress Over Time

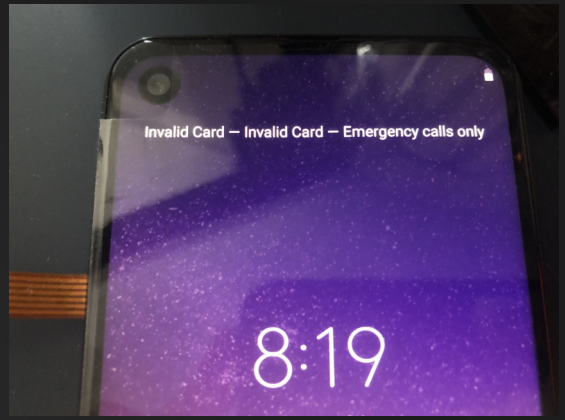


2022/06/21

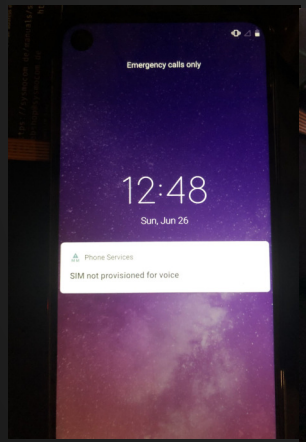


2022/06/23

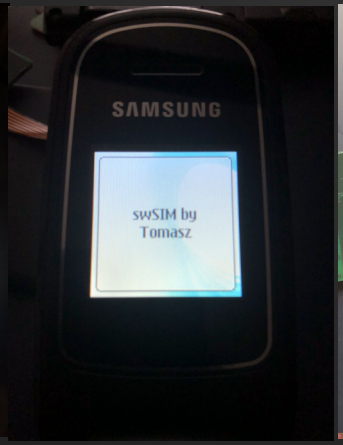
2022/06/24



2022/06/24



2022/06/25



2022/06/25

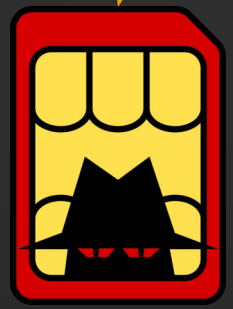
2022/06/27

2022/11/16

Chapter 1:
Diving Into Hostile SIMs

SIMs can do WHAT?

Hostile SIM Threat Model

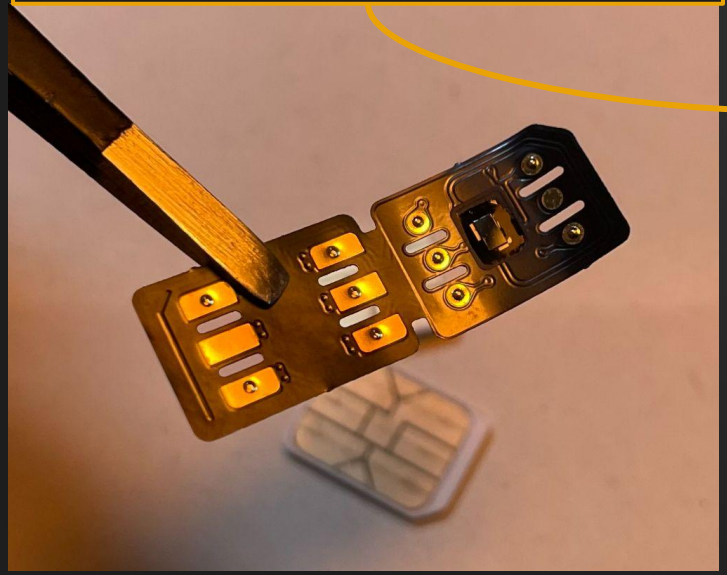


1.SIM SW Stack Vulns: Allowing remote exploitation by attackers.

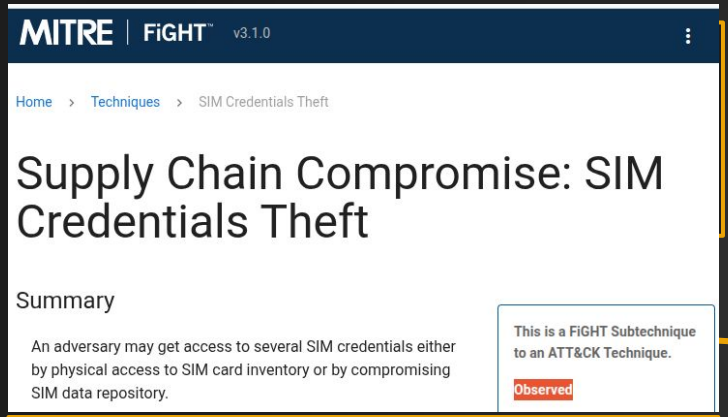
Hostile SIM Threat Model

1.SIM SW Stack Vulns:
Allowing remote exploitation by attackers.

2.Physical Access:
- Exchange the SIM
- Install an interposer



Hostile SIM Threat Model



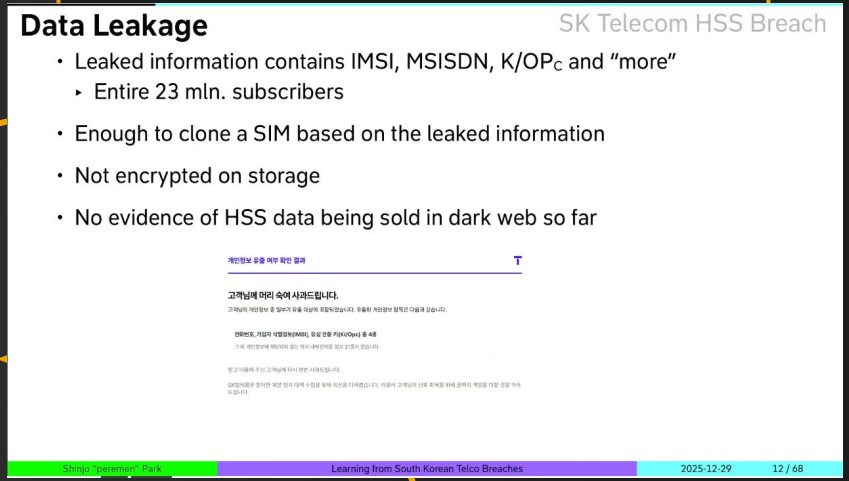
1.SIM SW Stack Vulns: Allowing remote exploitation by attackers.

3.Supply-Chain Attacks: SIM back-doors added anywhere between manufacturing and distribution.

Hostile SIM Threat Model

2.Physical Access:
- Exchange the SIM
- Install an interposer

3.Supply-Chain Attacks:
SIM back-doors added anywhere between manufacturing and distribution.



4.SIM Remote Admin:
Available to operators. Allows for OTA updates and initiating attacks.

Hostile SIM Threat Model

1.SIM SW Stack Vulns:
Allowing remote exploitation by attackers.

2.Physical Access:
- Exchange the SIM
- Install an interposer

3.Supply-Chain Attacks:
SIM back-doors added anywhere between manufacturing and distribution.

4.SIM Remote Admin:
Available to operators. Allows for OTA updates and initiating attacks.

SIM Attack Surface

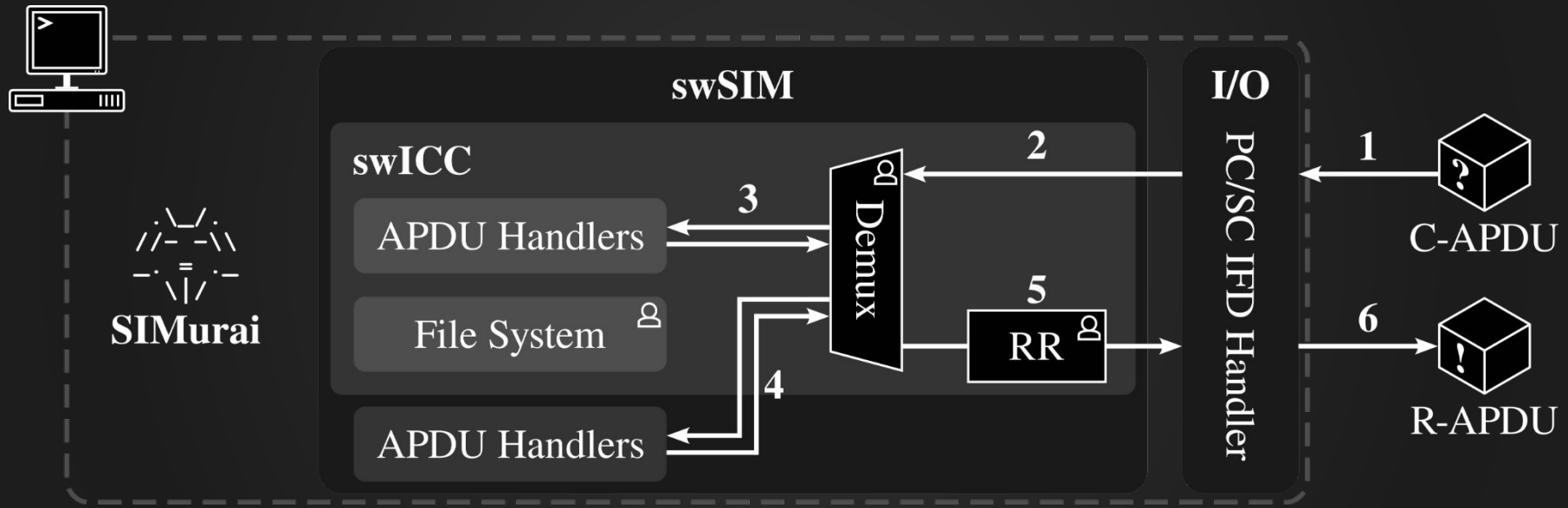
- Others laid the groundwork.

- Let’s explore it further using FirmWire + swSIM!

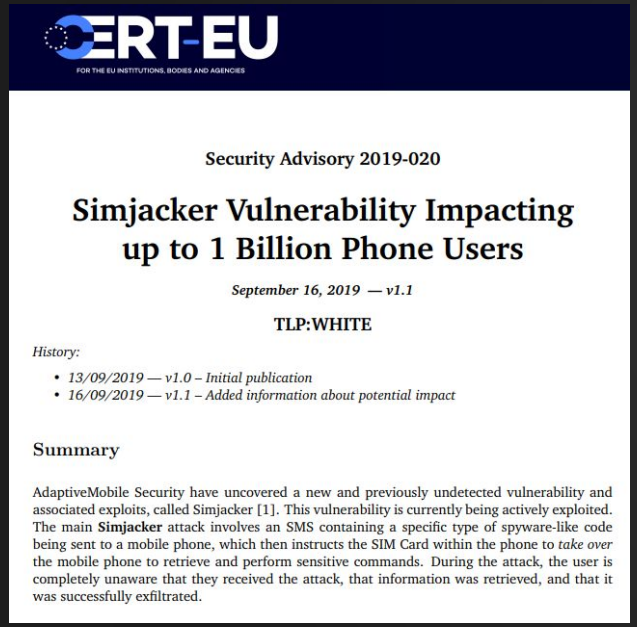
SIMurai:

- Our new security-centric tool
- Not always in spec

SIMurai Architecture/Setup



SIMurai: **Re-Implementing Simjacker Spyware**



CERT-EU
FOR THE EU INSTITUTIONS, BODIES AND AGENCIES

Security Advisory 2019-020

Simjacker Vulnerability Impacting up to 1 Billion Phone Users

September 16, 2019 — v1.1

TLP:WHITE

History:

- *13/09/2019 — v1.0 – Initial publication*
- *16/09/2019 — v1.1 – Added information about potential impact*

Summary

AdaptiveMobile Security have uncovered a new and previously undetected vulnerability and associated exploits, called Simjacker [1]. This vulnerability is currently being actively exploited. The main **Simjacker** attack involves an SMS containing a specific type of spyware-like code being sent to a mobile phone, which then instructs the SIM Card within the phone to *take over* the mobile phone to retrieve and perform sensitive commands. During the attack, the user is completely unaware that they received the attack, that information was retrieved, and that it was successfully exfiltrated.

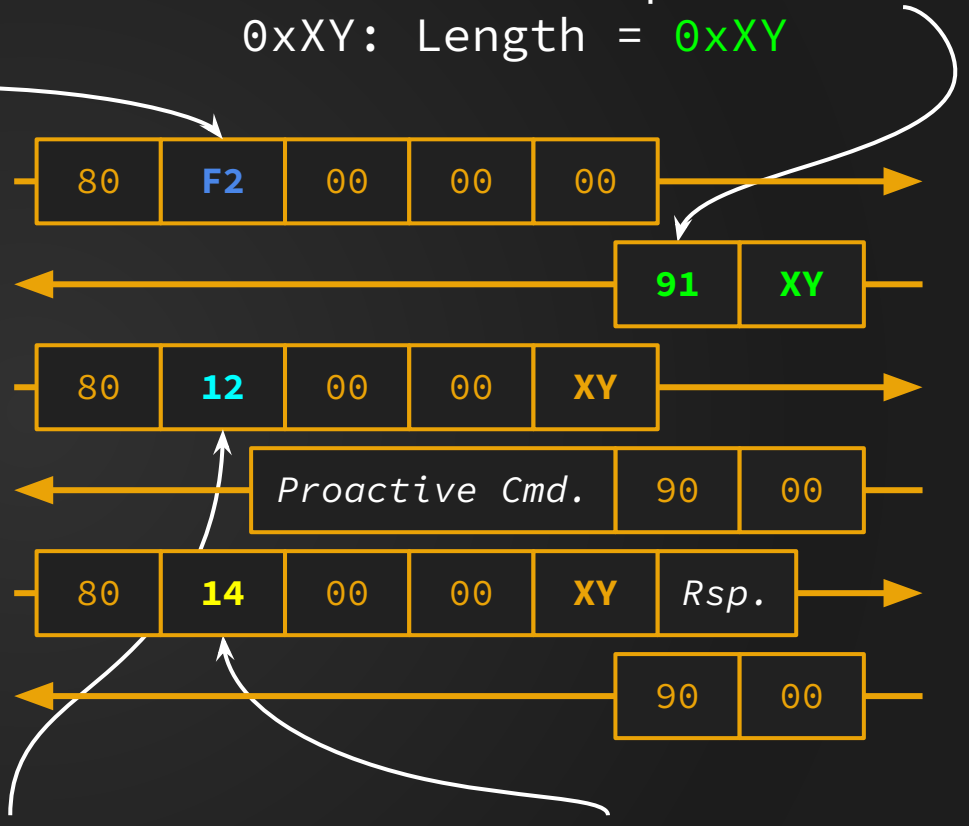
- SIM Spyware based on proactive commands
- Initial delivery via vulnerabilities in S@T Browser
- Variations still in use today![1]
- Our code cannot be installed on commercial cards. Not weaponized!

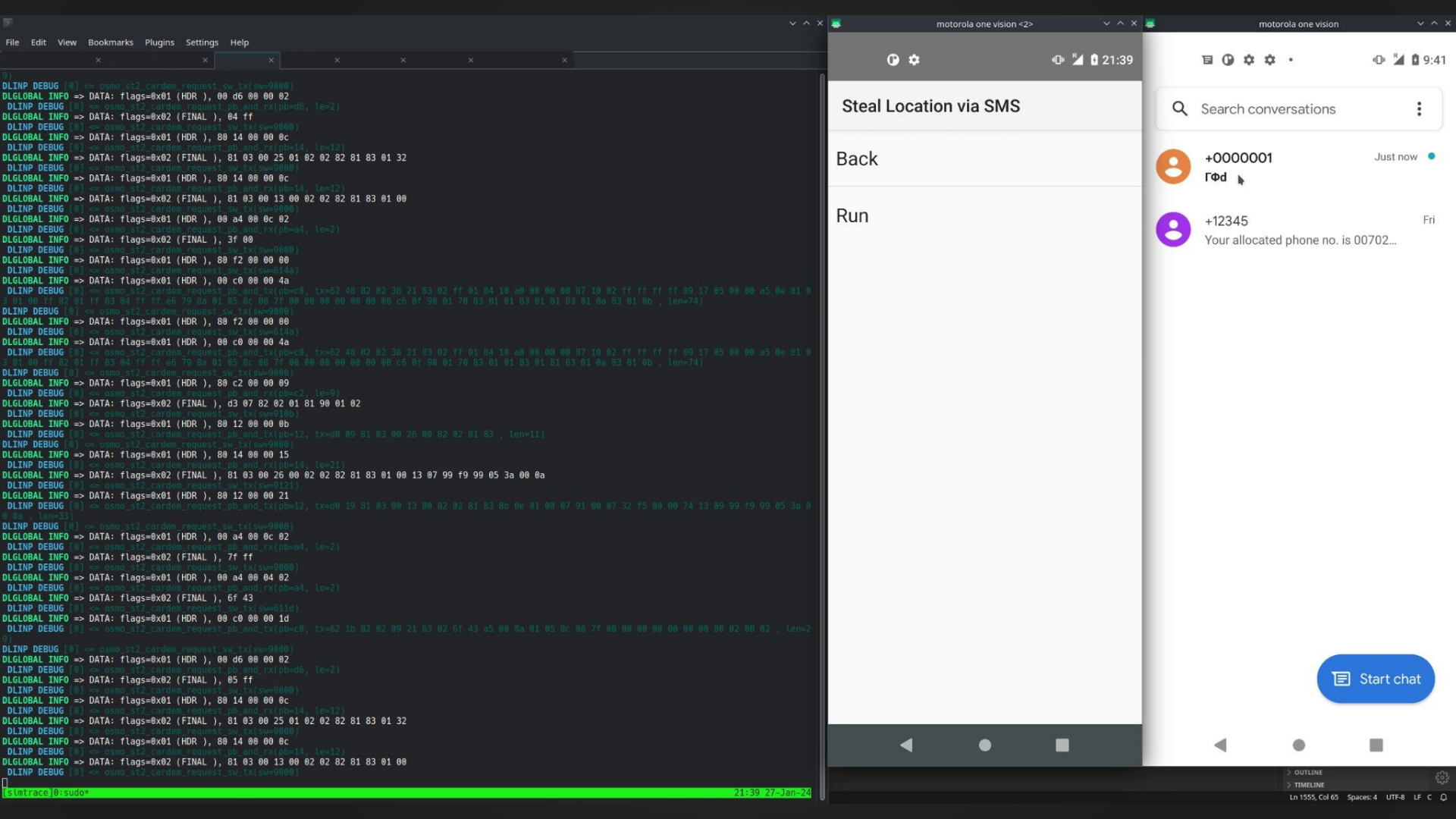
[1] https://citizenlab.ca/research/uncovering-global-telecom-exploitation-by-covert-surveillance-actors/#STA2-the-sim-as-the-spy

Proactive Commands

- “STATUS” (polling)
- OR any other APDU with a success response 0x9000

1. **SIM** card **prepares** commands
2. **Terminal fetches** commands
3. **Terminal sends** back **result** eventually.




File Edit View Bookmarks Plugins Settings Help
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=9000)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 00 d6 00 00 02
DLINP DEBUG [0] <= osmo_st2_cardem_request_pb_and_rx(pb=d6, le=2)
DLGLOBAL INFO => DATA: flags=0x02 (FINAL), 04 ff
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=9000)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 80 14 00 00 0c
DLINP DEBUG [0] <= osmo_st2_cardem_request_pb_and_rx(pb=14, le=12)
DLGLOBAL INFO => DATA: flags=0x02 (FINAL), 81 03 00 25 01 02 02 82 81 83 01 32
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=9000)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 80 14 00 00 0c
DLINP DEBUG [0] <= osmo_st2_cardem_request_pb_and_rx(pb=14, le=12)
DLGLOBAL INFO => DATA: flags=0x02 (FINAL), 81 03 00 13 00 02 02 82 81 83 01 00
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=9000)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 00 a4 00 0c 02
DLINP DEBUG [0] <= osmo_st2_cardem_request_pb_and_rx(pb=a4, le=2)
DLGLOBAL INFO => DATA: flags=0x02 (FINAL), 3f 00
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=9000)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 80 f2 00 00 00
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=614a)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 00 c0 00 00 4a
DLINP DEBUG [0] <= osmo_st2_cardem_request_pb_and_tx(pb=c0, tx=62 48 82 02 38 21 83 02 ff 01 84 10 a0 00 00 00 87 10 02 ff ff ff ff 89 17 05 00 00 a5 0e 81 03 01 00 ff 82 01 ff 83 04 ff ff e6 79 8a 01 05 8c 08 7f 00 00 00 00 00 00 00 c6 0f 90 01 70 83 01 01 83 01 81 83 01 0a 83 01 0b , len=74)
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=9000)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 80 f2 00 00 00
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=614a)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 00 c0 00 00 4a
DLINP DEBUG [0] <= osmo_st2_cardem_request_pb_and_tx(pb=c0, tx=62 48 82 02 38 21 83 02 ff 01 84 10 a0 00 00 00 87 10 02 ff ff ff ff 89 17 05 00 00 a5 0e 81 03 01 00 ff 82 01 ff 83 04 ff ff e6 79 8a 01 05 8c 08 7f 00 00 00 00 00 00 00 c6 0f 90 01 70 83 01 01 83 01 81 83 01 0a 83 01 0b , len=74)
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=9000)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 80 c2 00 00 09
DLINP DEBUG [0] <= osmo_st2_cardem_request_pb_and_rx(pb=c2, le=9)
DLGLOBAL INFO => DATA: flags=0x02 (FINAL), d3 07 82 02 01 81 90 01 02
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=6100)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 80 12 00 00 0b
DLINP DEBUG [0] <= osmo_st2_cardem_request_pu_and_tx(pb=12, tx=d0 09 81 03 00 26 00 82 02 81 83 , len=11)
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=9000)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 80 14 00 00 15
DLINP DEBUG [0] <= osmo_st2_cardem_request_pb_and_rx(pb=14, le=21)
DLGLOBAL INFO => DATA: flags=0x02 (FINAL), 81 03 00 26 00 02 02 82 81 83 01 00 13 07 99 f9 99 05 3a 00 0a
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=9121)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 80 12 00 00 21
DLINP DEBUG [0] <= osmo_st2_cardem_request_pb_and_tx(pb=12, tx=d0 19 81 03 00 13 00 82 02 81 83 8b 0e 01 00 07 91 00 07 32 f5 00 00 74 13 09 99 f9 99 05 3a 0 0 0a , len=33)
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=9000)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 00 a4 00 0c 02
DLINP DEBUG [0] <= osmo_st2_cardem_request_pb_and_rx(pb=a4, le=2)
DLGLOBAL INFO => DATA: flags=0x02 (FINAL), 7f ff
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=9000)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 00 a4 00 04 02
DLINP DEBUG [0] <= osmo_st2_cardem_request_pb_and_rx(pb=a4, le=2)
DLGLOBAL INFO => DATA: flags=0x02 (FINAL), 6f 43
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=611d)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 00 c0 00 00 1d
DLINP DEBUG [0] <= osmo_st2_cardem_request_pb_and_tx(pb=c0, tx=62 1b 82 02 09 21 83 02 6f 43 a5 08 8a 01 05 8c 08 7f 00 00 00 00 00 00 00 80 02 00 02 , len=29)
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=9000)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 00 d6 00 00 02
DLINP DEBUG [0] <= osmo_st2_cardem_request_pb_and_rx(pb=d6, le=2)
DLGLOBAL INFO => DATA: flags=0x02 (FINAL), 05 ff
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=9000)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 80 14 00 00 0c
DLINP DEBUG [0] <= osmo_st2_cardem_request_pb_and_rx(pb=14, le=12)
DLGLOBAL INFO => DATA: flags=0x02 (FINAL), 81 03 00 25 01 02 02 82 81 83 01 32
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=9000)
DLGLOBAL INFO => DATA: flags=0x01 (HDR), 80 14 00 00 0c
DLINP DEBUG [0] <= osmo_st2_cardem_request_pb_and_rx(pb=14, le=12)
DLGLOBAL INFO => DATA: flags=0x02 (FINAL), 81 03 00 13 00 02 02 82 81 83 01 00
DLINP DEBUG [0] <= osmo_st2_cardem_request_sw_tx(sw=9000)
[simtrace]0:sudo*
21:39 27-Jan-24
motorola one vision <2>
21:39
Steal Location via SMS
Back
Run
motorola one vision
9:41
Search conversations
+0000001
Just now
ГФd
+12345
Fri
Your allocated phone no. is 00702...
Start chat
OUTLINE
TIMELINE
Ln 1555, Col 65
Spaces: 4
UTF-8
LF
C

SIMurai: **Fuzzing**

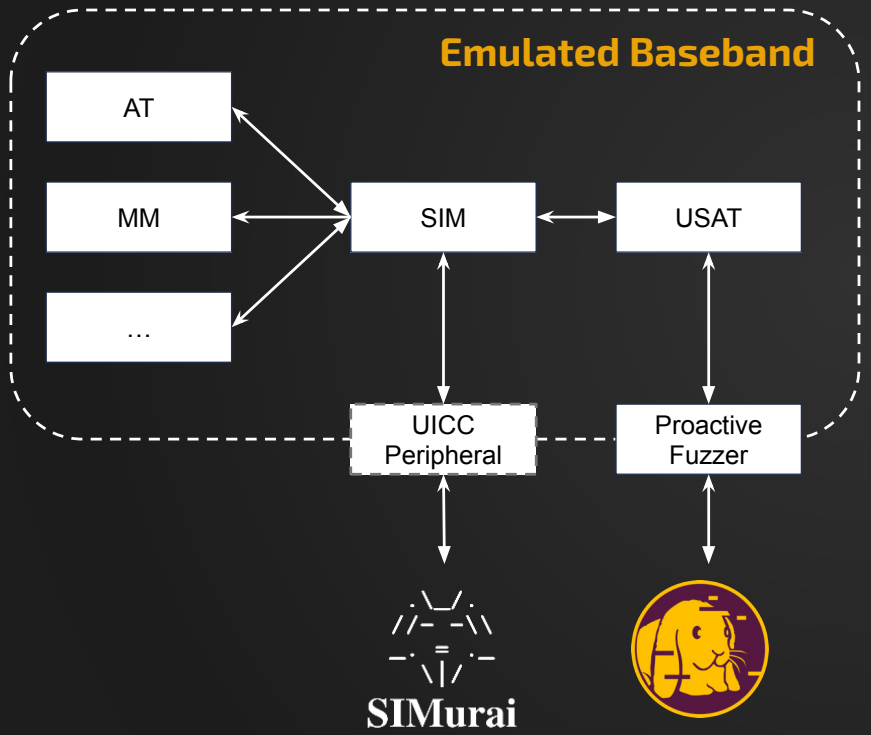
💡Inputs from Hostile SIMs need to be parsed.

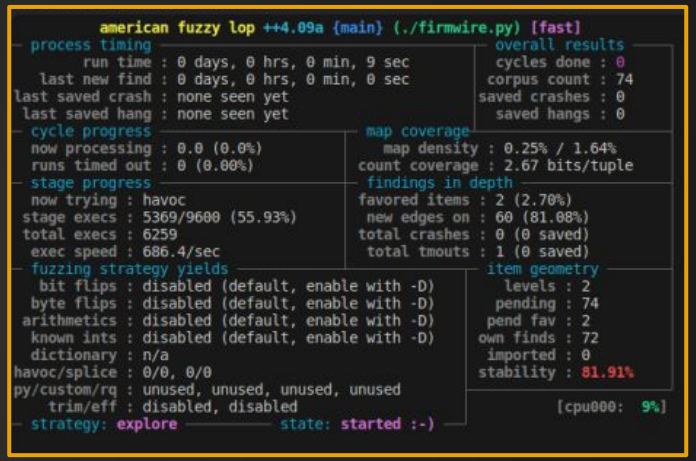
~~Attempt #1: SIM File-based fuzzing~~

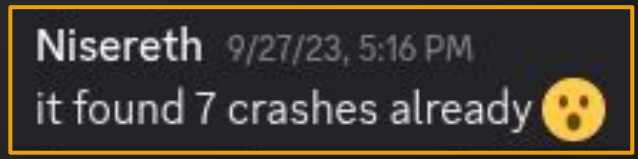
~~Attempt #2: Fuzzing real-hardware~~

Attempt #3: Fuzzing Proactive commands in FimWire

The Final Fuzzing Setup








[0.01517][USAT] 0x41bdefeb 0b100: [../../../PSS/StackService/USAT/Code/Src/usat_UsimIntfManagement.c] - [USAT_0] In usat_CheckCmdSuppInTermProf - Curr Rat - 255, cmd typ
[0.01551][USAT] 0x41be4c7b 0b100: [../../../PSS/StackService/USAT/Code/Src/usat_ProactiveCmdHandler.c] - [USAT_0] usat_CurrentCmdDetails[0],[1],[2] 7f,40,80
[0.01575][USAT] 0x41bdfb63 0b100: [../../../PSS/StackService/USAT/Code/Src/usat_UsimIntfManagement.c] - [USAT_0] CheckCmdSuppInTermProf: result :1
[0.01603][USAT] 0x41be3d0d 0b100: [../../../PSS/StackService/USAT/Code/Src/usat_TimerManagement.c] - [USAT_0] StartTimer: USAT_TIMER_PROACTIVECMD_TR, duratn 600000 ms
[0.01816][USAT] 0x41bdceeb 0b101: [../../../PSS/StackService/USAT/Code/Src/usat_main.c] - [USAT_0] Get :FlagIndex 12 result 1
[0.01841][USAT] 0x41bf06cb 0b101: [../../../PSS/StackService/USAT/Code/Src/usat_BipConnIntfManagement.c] - [USAT_0] [USAT BIP] usat_BipConnectionCmdHandler
[0.01896][USAT] 0x41bf04ed 0b101: [../../../PSS/StackService/USAT/Code/Src/usat_BipConnIntfManagement.c] - [USAT_0] [USAT BIP] usat_OpenChannel
[0.01933][USAT] 0x41bf02a5 0b101: [../../../PSS/StackService/USAT/Code/Src/usat_BipConnIntfManagement.c] - [USAT_0] [USAT BIP] usat_CheckAndUpdatePreOpenConnStateData
[0.01965][USAT] 0x41bf031b 0b101: [../../../PSS/StackService/USAT/Code/Src/usat_BipConnIntfManagement.c] - [USAT_0] [USAT BIP]Bearer Type : 0 is not supported
[0.01987][USAT] 0x41befbcf 0b101: [../../../PSS/StackService/USAT/Code/Src/usat_BipConnIntfManagement.c] - [USAT_0] [USAT BIP] usat_HandleOpenChannelTr, result:32
[0.02049][USAT] 0x41bdceeb 0b101: [../../../PSS/StackService/USAT/Code/Src/usat_main.c] - [USAT_0] Get :FlagIndex 13 result 0
[0.02081][USAT] 0x41be4d6d 0b101: [../../../PSS/StackService/USAT/Code/Src/usat_ProactiveCmdHandler.c] - [USAT_0] CreateAndSendTerminalResponse: TR len= 0x14
[0.02132][USAT] 0x41bdebe7 0b100: [../../../PSS/StackService/USAT/Code/Src/usat_UsimIntfManagement.c] - [USAT_0] SendTerminalResponse:
[0.02168][USAT] 0x41bde0ad 0b100: [../../../PSS/StackService/USAT/Code/Src/usat_pducodec.c] - [USAT_0] AllocateMemoryForMsg: MsgLength = 16
[0.02217][USAT] 0x41bde66b 0b11: [../../../PSS/StackService/USAT/Code/Src/usat_pducodec.c] - [USAT_0] 8. USAT ==> SIM_TERMINAL_RESPONSE_REQ [USIM]
[0.02232][USAT] 0x41bde68d 0b10: [../../../PSS/StackService/USAT/Code/Src/usat_pducodec.c] - [USAT_0] ------------Dis
Hex Dump Of Message....--------------
[0.02260][USAT] pal_MsgSendTo+0x3ff (0x41004
) pal_
endTo(SIM (46)) - PALMsg<0x2f71, USAT (bc) -> SIM (2e), 8 bytes>
[0.02284][SIM] 0x40e9f3bf 0b101: [../../../P
StackSe
/USIM/Code/Src/u
_0]
USIM TASK ----------
[0.02312][SIM] 0x409c7d67 0b101: [../../../P
StackSe
/USIM/Code/Src/u
.c]
USIM
--- USIM Tim

[0.02324][SIM] 0x409c7d99 0b101: [../../../P
StackSe
/USIM/Code/Src/u_Time
nager
.c]
USIM_
USIM_CARD_PR
T_POLL_TIMER
[0.02338][SIM] 0x409c7e15 0b101: [../../../P
StackSe
/USIM/Code/Src/u_Time
nager
.c]
USIM_
Timer 0 - RU
G
[0.02349][SIM] 0x409c7d99 0b101: [../../../P
/USIM/Code/Src/u_Time
nager
.c]
USIM_
USIM_PBM_SMS
CHING_TIMER_1
[0.02357][SIM] 0x409c7e15 0b101: [../../../P
/USIM/Code/Src/u_Time
nager
.c]
Timer 1 - RU
G
[0.02365][SIM] 0x409c7d99 0b101: [../../../PSS/StackService/USIM/Code/Src/usim_TimerManagement.c] - [USIM_0] USIM_PBM_SMS_CACHING_TIMER_2
[0.02376][SIM] 0x409c7dc3 0b101: [../../../PSS/StackService/USIM/Code/Src/usim_TimerManagement.c] - [USIM_0] Timer 2 - STOPPED
[0.02384][SIM] 0x409c7d99 0b101: [../../../PSS/StackService/USIM/Code/Src/usim_TimerManagement.c] - [USIM_0] USIM_TEMPORARY_UNLOCK_TIMER
[0.02393][SIM] 0x409c7dc3 0b101: [../../../PSS/StackService/USIM/Code/Src/usim_TimerManagement.c] - [USIM_0] Timer 3 - STOPPED
[0.02401][SIM] 0x409c7d99 0b101: [../../../PSS/StackService/USIM/Code/Src/usim_TimerManagement.c] - [USIM_0] USIM_RECOVERY_RESTART_TIMER
[0.02409][SIM] 0x409c7dc3 0b101: [../../../PSS/StackService/USIM/Code/Src/usim_TimerManagement.c] - [USIM_0] Timer 4 - STOPPED
[0.02418][SIM] 0x409c7d99 0b101: [../../../PSS/StackService/USIM/Code/Src/usim_TimerManagement.c] - [USIM_0] USIM_SCHEDULE_SIM_INIT_TIMER
[0.02426][SIM] 0x409c7dc3 0b101: [../../../PSS/StackService/USIM/Code/Src/usim_TimerManagement.c] - [USIM_0] Timer 5 - STOPPED
[0.02433][SIM] 0x409c7d99 0b101: [../../../PSS/StackService/USIM/Code/Src/usim_TimerManagement.c] - [USIM_0] USIM_SYNC_MSG_RSP_TIMER
[0.02441][SIM] 0x409c7dc3 0b101: [../../../PSS/StackService/USIM/Code/Src/usim_TimerManagement.c] - [USIM_0] Timer 6 - STOPPED
[0.02449][SIM] 0x409c7d99 0b101: [../../../PSS/StackService/USIM/Code/Src/usim_TimerManagement.c] - [USIM_0] USIM_VSIM_SWITCH_TO_PHYSIM_TIMER
[0.02457][SIM] 0x409c7dc3 0b101: [../../../PSS/StackService/USIM/Code/Src/usim_TimerManagement.c] - [USIM_0] Timer 7 - STOPPED
[0.02473][SIM] 0x40e9d32b 0b101: [../../../PSS/StackService/USIM/Code/Src/usim_main.c] - [USIM_0] ---USIM Flags---
[0.02494][SIM] 0x40e9d363 0b101: [../../../PSS/StackService/USIM/Code/Src/usim_main.c] - [USIM_0] No Flags are Set !
[0.02512][SIM] 0x40e9d029 0b11: [../../../PSS/StackService/USIM/Code/Src/usim_main.c] - [USIM_0] --- Start Displaying USIM Message Log ---
[0.02533][SIM] 0x40e9d137 0b11: [../../../PSS/StackService/USIM/Code/Src/usim_main.c] - [USIM_0] --- End Displaying USIM Message Log ---
[0.02595][USAT] 0x41bdceeb 0b101: [../../../PSS/StackService/USAT/Code/Src/usat_main.c] - [USAT_0] Get :FlagIndex 11 result 0
[0.02668][USAT] 0x41bee621 0b101: [../../../PSS/StackService/USAT/Code/Src/usat_BipConnIntfManagement.c] - [USAT_0] [USAT BIP] Deep Freed [pSimAtkBipConnState] [4139c5a
[ERROR] firmwire.vendor.shannon.hooks: EXCEPTION: PREFETCH ABORT (USAT) - Faulting PC: 0xaaaaaaae
Shutting down...
Demo1

Crash Analysis - 2 Key Root Causes

Memory corruptions in parsing incoming proactive commands:

PROACTIVE_SEND_SMS

- DATA_ABORT

→ Null Pointer Dereference

PROACTIVE_SEND_SS

- PREFETCH ABORTS
- MEM_GUARD_CORRUPTION

→ Heap Overflow

*Replicated on (newer) hardware, reported, and **fixed**!*

Detour 0: Accidental Lockscreen Bypass

Sure Dave, I can let you do this.

A Special Proactive Command

6.4.26 LAUNCH BROWSER

This clause applies if class "ab" is supported.

For a terminal of type ND or NK the support of this command is optional.

This command is used to request a browser inside a browser-enabled terminal to interpret the content corresponding to a URL.

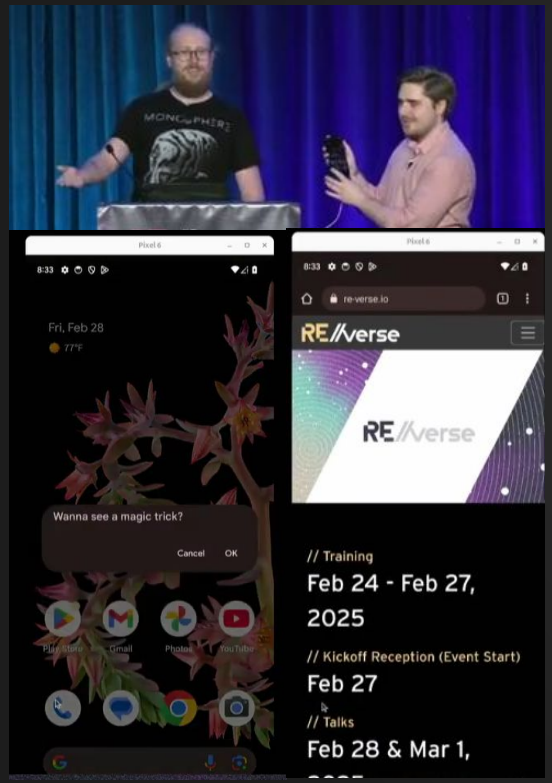
Upon receiving this command, the terminal shall decide if it is able to execute the command. Examples are given below, but the list is not exhaustive:

- if the command is rejected because the browser on the terminal is busy or not available, the terminal informs the UICC using TERMINAL RESPONSE (launch browser generic error - browser unavailable);
- if the command is rejected because the bearer provided in the command is not available, the terminal informs the UICC using TERMINAL RESPONSE (launch browser generic error - bearer unavailable).

If the terminal is able to execute the command:

- the terminal shall inform the UICC that the command has been successfully taken into account, using TERMINAL RESPONSE;
- the UICC shall end the proactive UICC session;
- then the terminal shall request content using the URL;
- if an error occurs when accessing the resource indicated in the URL, the terminal shall send to the UICC a browsing status event reporting the error (if the browsing status event is part of the event list).

Accidental Findings



https://www.youtube.com/watch?v=2ou_MxSmiVo

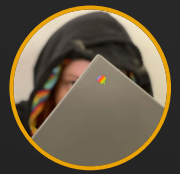
Preparing for the presentation

Tomasz Lisowski 2:30 PM
oh wow
I found a bypass to the menu
no clicks needed now

Marius Muench 2:42 PM
Wait what? SO launch browser without menu?

Tomasz Lisowski 2:42 PM
yes
let me show you

Brainstorming with friends



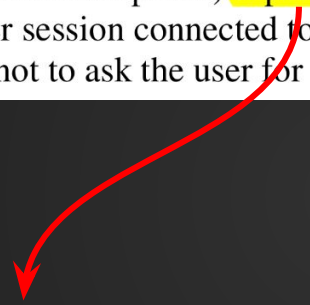
“How does this interact with lockscreens?”

Demo 2

LAUNCHing BROWSER Under the Lockscreen

Why does this work?

Unless a terminal is of type NK or type ND, the terminal shall ask the user for confirmation using the Alpha Identifier/Icon Identifier (user confirmation phase) if present, when it receives a LAUNCH BROWSER command which requests the existing browser session connected to a new URL or to terminate a browser session. It is allowed for a terminal of type NK or type ND not to ask the user for confirmation.



```
Missing (or NULL) Alpha Identifier/Icon Identifier

    → Undefined Behavior
```

Where does this work?

At the time of testing (March 2025):

Without User Interaction:

- Galaxy A14
- Galaxy S22
- Motorola G32
- Motorola One Vision
- Pixel 6
- Pixel 8
- Pixel 9

User Interaction Required:

- Galaxy S9
- Galaxy S10e
- Galaxy A41
- ZTE Blade A54
- Huawei P40 Lite

The Fix

Google Git

Sign in

android / platform / frameworks / opt / telephony / **fee68bcdcf029e8f40980616d09367610544bc62**

commit	fee68bcdcf029e8f40980616d09367610544bc62	[log]
author	arunvoddu <arunvoddu@google.com>	Tue Jul 01 16:48:48 2025 +0000
committer	Android Build Coastguard Worker <android-build-coastguard-worker@google.com>	Fri Oct 10 14:19:42 2025 -0700
tree	a351c8b7482c4452b109e0f8c35856add0ab5cf1	
parent	c4041d9ea314c8226fdfde2fca0eaff223da1805 [diff]	

```
[Telephony][Security Fix] Launch Browser only if device is unlocked.

Ignore the launch browser proactive cmd from modem to STK if the device screen is locked.

Bug: 404254549
Flag: EXEMPT Bugfix.
Test: Verified manually with TestApk
(cherry picked from https://googleplex-android-
review.googlesource.com/q/commit:dcf5c112a93dc8fcc67d65434707e205fd79cee2)
Cherrypick-From: https://googleplex-android-review.googlesource.com/q/commit:d3774ac65a38121b242f75388a08971883cc05d2
Merged-In: I9f651c7f1c5674df5774f5a7609f6d3749b5c50c
Change-Id: I9f651c7f1c5674df5774f5a7609f6d3749b5c50c
```

src/java/com/android/internal/telephony/cat/CommandParamsFactory.java [diff]
src/java/com/android/internal/telephony/cat/ResultException.java [diff]

2 files changed

Chapter 2: Equipping the CATana

AT commands in SIM spec? Of course!

Another Special Proactive Command

6.4.23 RUN AT COMMAND

This clause applies if class "b" is supported by the terminal and enabled by the subscriber through the terminal.

If this feature is enabled, the UICC uses this command to send an AT Command to the terminal as though initiated by an attached TE. The terminal shall then return an AT Response within a TERMINAL RESPONSE to the UICC. The terminal shall respond with "command performed successfully" upon completion of the command, regardless of the actual response of the AT Command which is provided to the UICC in the AT Response data object.

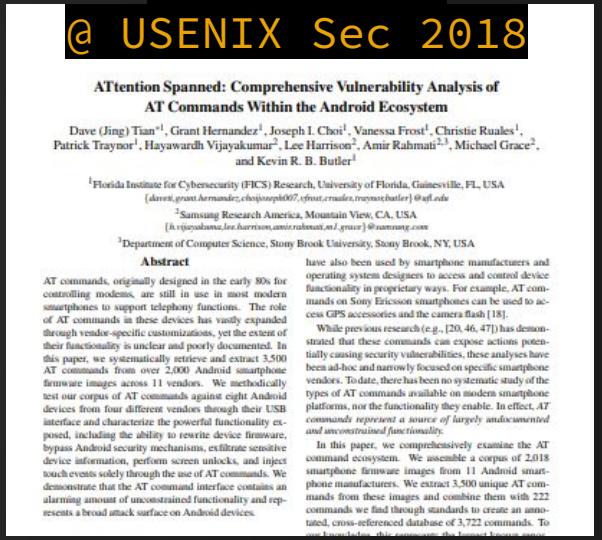
If this feature is disabled in the card or in the terminal, or the terminal does not support the RUN AT COMMAND, then if the CAT receives an instruction from the network to issue the command, the CAT should return an error to the network.

Optionally, the UICC may include in this command an alpha identifier. The use of this alpha identifier by the terminal is described below:

- if the alpha identifier is provided by the UICC and is not a null data object, the terminal shall use it to inform the user. This is also an indication that the terminal should not give any other information to the user on the fact that the terminal is performing an AT command. If an icon is provided by the UICC, the icon indicated in the command may be used by the terminal to inform the user, in addition to, or instead of the alpha identifier, as indicated with the icon qualifier (see clause 6.5.4);
- if the alpha identifier is provided by the UICC and is a null data object (i.e. length = '00' and no value part), this is an indication that the terminal should not give any information to the user on the fact that the terminal is performing an AT command;
- if the alpha identifier is not provided by the UICC, the terminal may give information to the user concerning what is happening.

A terminal of type ND shall ignore any alpha identifier provided together with this command and shall respond with "command performed successfully" upon successful completion of the command. A terminal of type ND shall also ignore any icon provided together with this command and shall respond with "command performed successfully but

Tian et al.
@ USENIX Sec 2018



ATtention Spanned: Comprehensive Vulnerability Analysis of AT Commands Within the Android Ecosystem

Dave (Jing) Tian*[1], Grant Hernandez[1], Joseph I. Choi[1], Vanessa Frost[1], Christie Ruales[1], Patrick Traynor[1], Hayawardh Vijayakumar[2], Lee Harrison[2], Amir Rahmati[2,3], Michael Grace[2], and Kevin R. B. Butler[1]

[1]Florida Institute for Cybersecurity (FICS) Research, University of Florida, Gainesville, FL, USA
{daven1,grant.hernandez,choijoseph1007,vfrost,cruales,traynor,butler}@ufl.edu

[2]Samsung Research America, Mountain View, CA, USA
{h.vijayakumar,lee.harrison,amir.rahmati,m1.grace}@samsung.com

[3]Department of Computer Science, Stony Brook University, Stony Brook, NY, USA

Abstract

AT commands, originally designed in the early 80s for controlling modems, are still in use in most modern smartphones to support telephony functions. The role of AT commands in these devices has vastly expanded through vendor-specific customizations, yet the extent of their functionality is unclear and poorly documented. In this paper, we systematically retrieve and extract 3,500 AT commands from over 2,000 Android smartphone firmware images across 11 vendors. We methodically test our corpus of AT commands against eight Android devices from four different vendors through their USB interface and characterize the powerful functionality exposed, including the ability to rewrite device firmware, bypass Android security mechanisms, exfiltrate sensitive device information, perform screen unlocks, and inject touch events solely through the use of AT commands. We demonstrate that the AT command interface contains an alarming amount of unconstrained functionality and represents a broad attack surface on Android devices.

have also been used by smartphone manufacturers and operating system designers to access and control device functionality in proprietary ways. For example, AT commands on Sony Ericsson smartphones can be used to access GPS accessories and the camera flash [18].

While previous research (e.g., [20, 46, 47]) has demonstrated that these commands can expose actions potentially causing security vulnerabilities, these analyses have been ad-hoc and narrowly focused on specific smartphone vendors. To date, there has been no systematic study of the types of AT commands available on modern smartphone platforms, nor the functionality they enable. In effect, *AT commands represent a source of largely undocumented and unconstrained functionality.*

In this paper, we comprehensively examine the AT command ecosystem. We assemble a corpus of 2,018 smartphone firmware images from 11 Android smartphone manufacturers. We extract 3,500 unique AT commands from these images and combine them with 222 commands we find through standards to create an annotated, cross-referenced database of 3,722 commands. To

Limited attack surface?

RUN AT barely supported by phones:

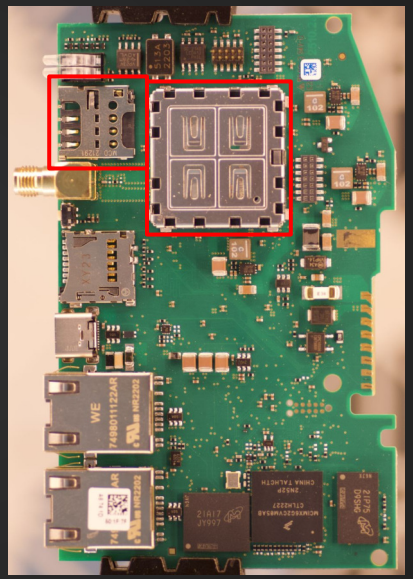
- Found only one instance in over 20 models
- “Interesting, but not worth further investigation …”

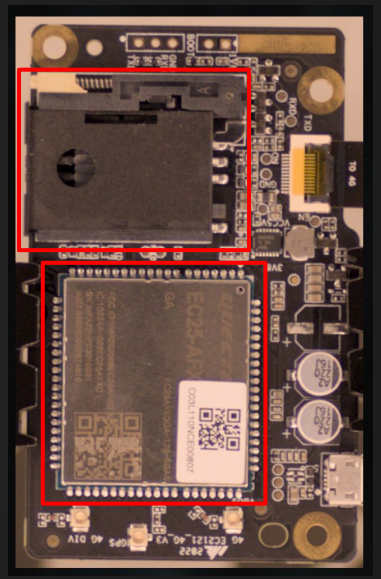
End of 2025:

“💡What about IoT devices?”

Where to find good target IoT devices?

- Interesting, high profile targets
- Car TCUs, EV Chargers, Automotive OSs
- Documented SIM slots!

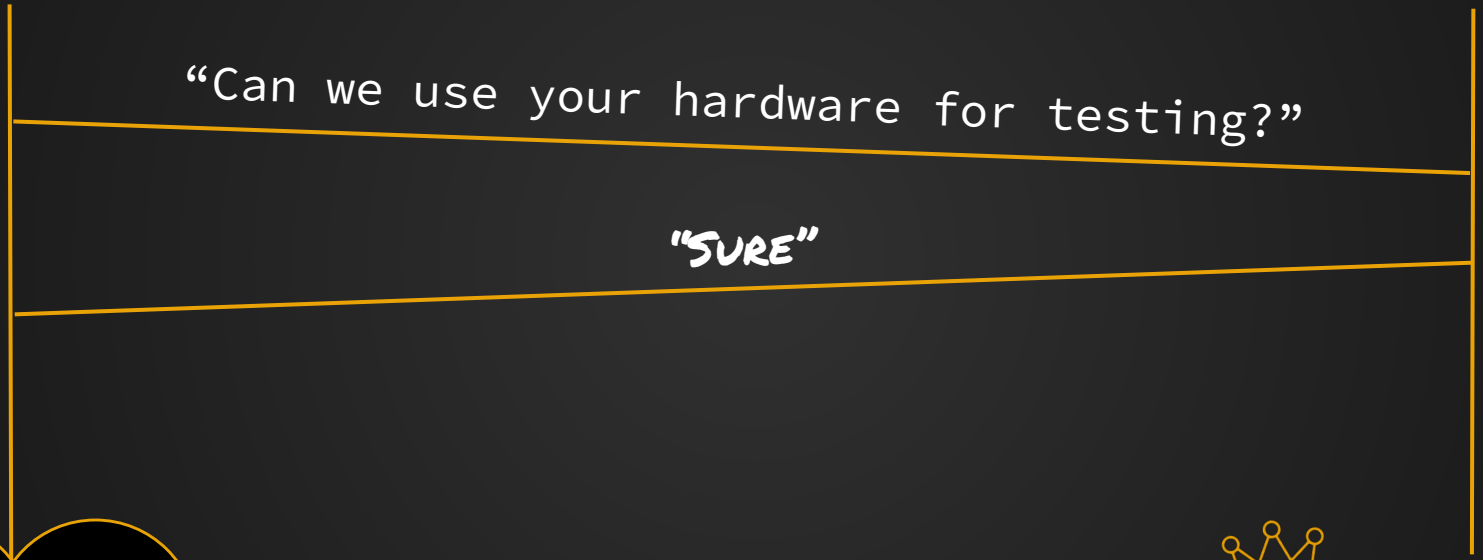


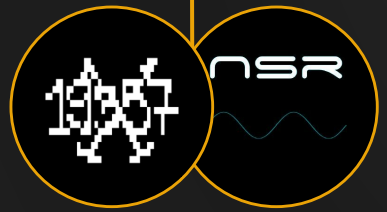


Todd Manning. “A Detailed Look at Pwn2Own Automotive EV Charger Hardware.” [1]

[1] https://www.thezdi.com/blog/2023/11/28/a-detailed-look-at-pwn2own-automotive-ev-charger-hardware

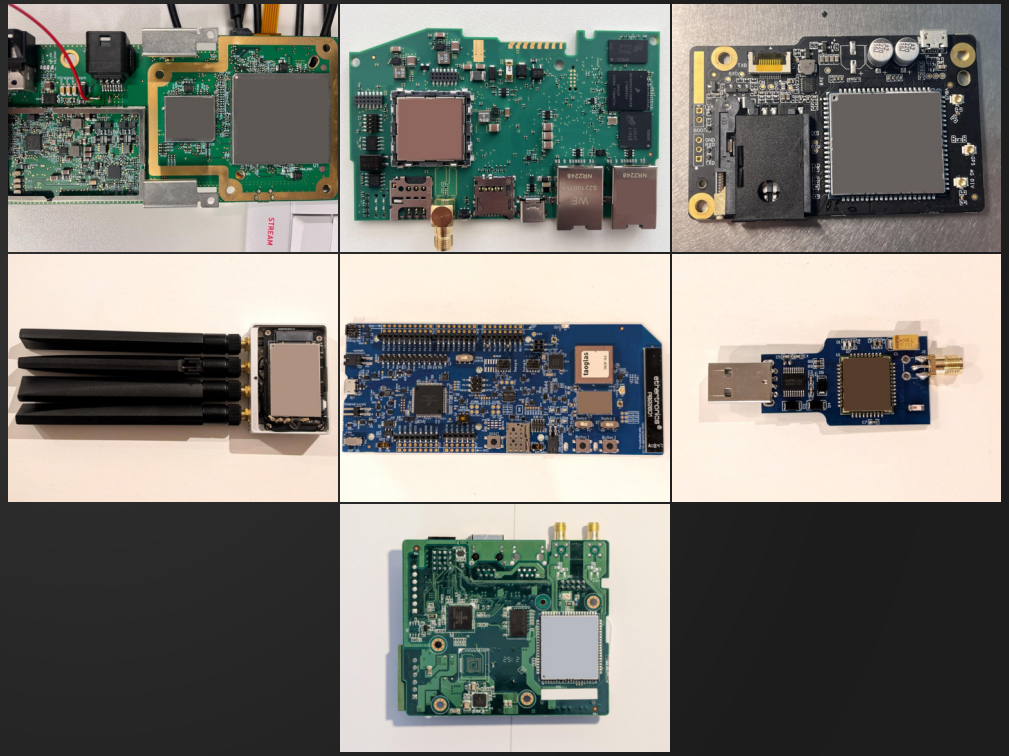
Where to find good target IoT devices?



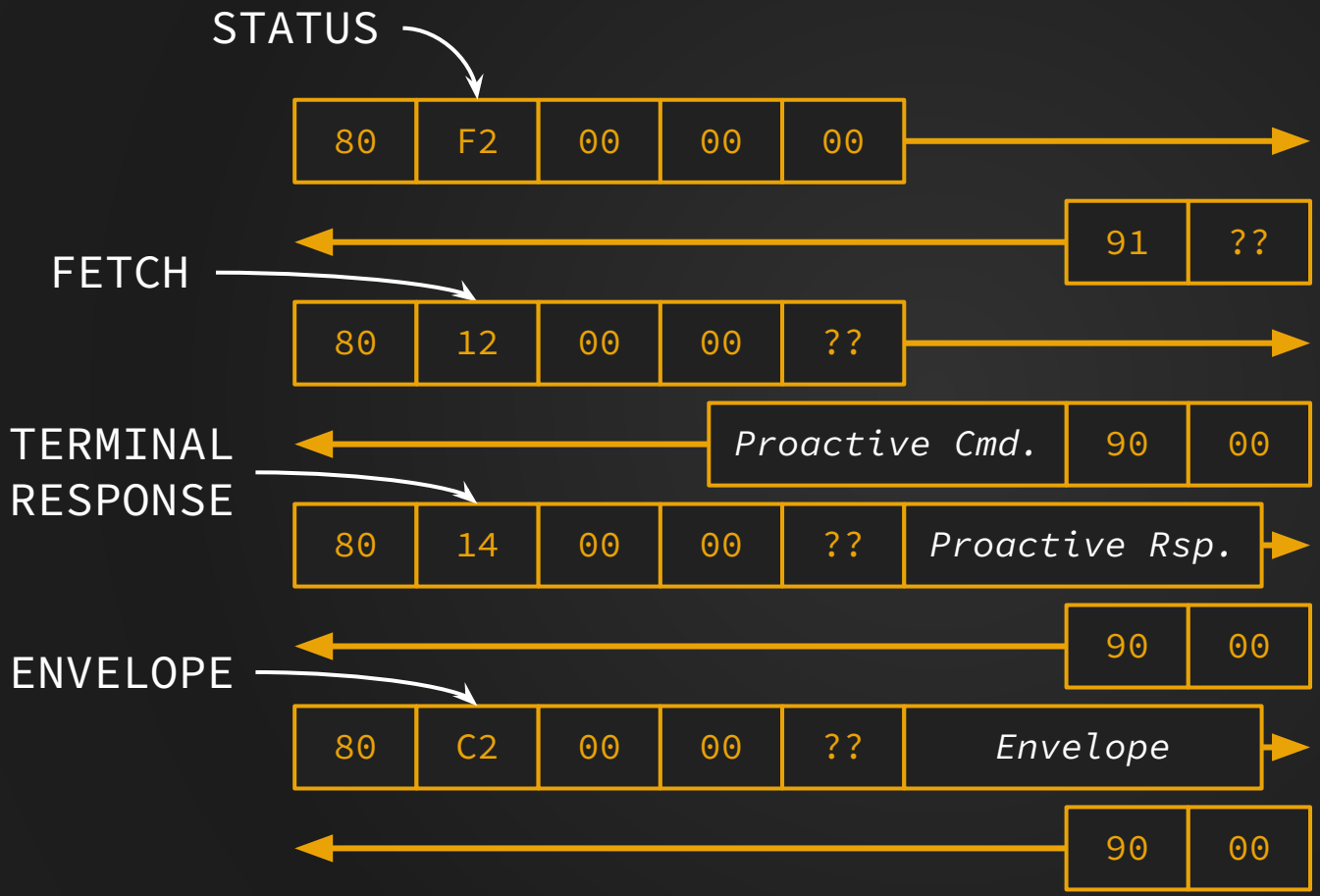


Testing for SIM AT Support

Vendor A - Device #1	✓
Vendor A - Device #2	✓
Vendor A - Device #3	✓
Vendor A - Device #4	✓
Vendor B - Device #1	
Vendor C - Device #1	✓
Vendor D - Device #1	



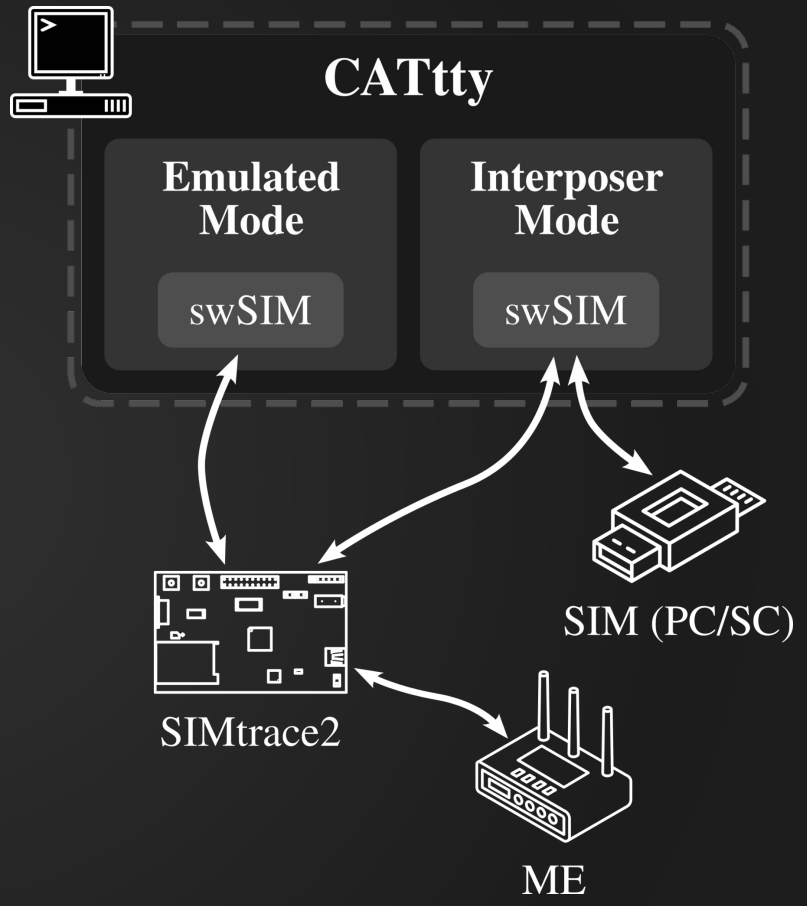
Sending AT from SIM



- Everything is **BER-TLV** encoded
- **Async**hronous process
- **Notifications** sent after AT handling are **not delivered**
- AT **data mode** most likely **not possible**

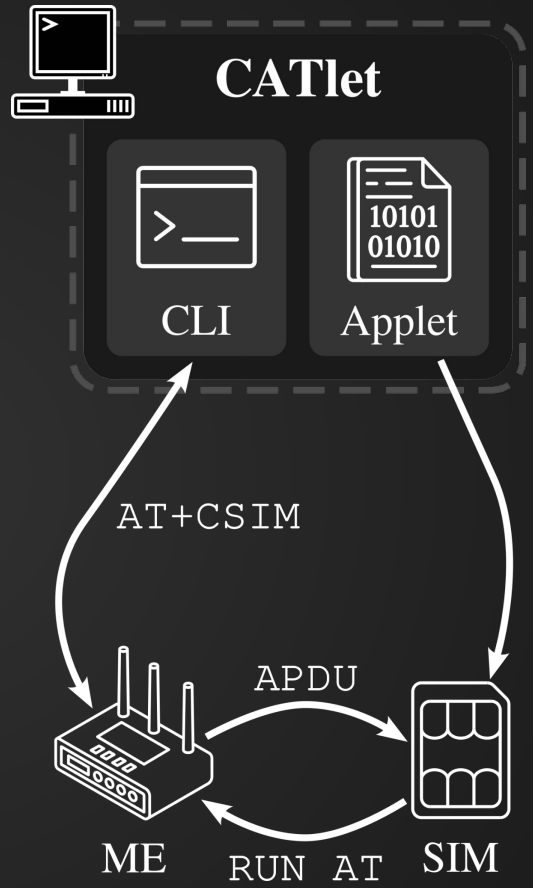
CATana: **CATtty**

- Custom **proactive app** for swSIM **to send AT** commands
- Testing only requires a **SIMtrace2 + swSIM**
- Real card not needed
- **Interposing built-in** if a real connection is needed for experiments



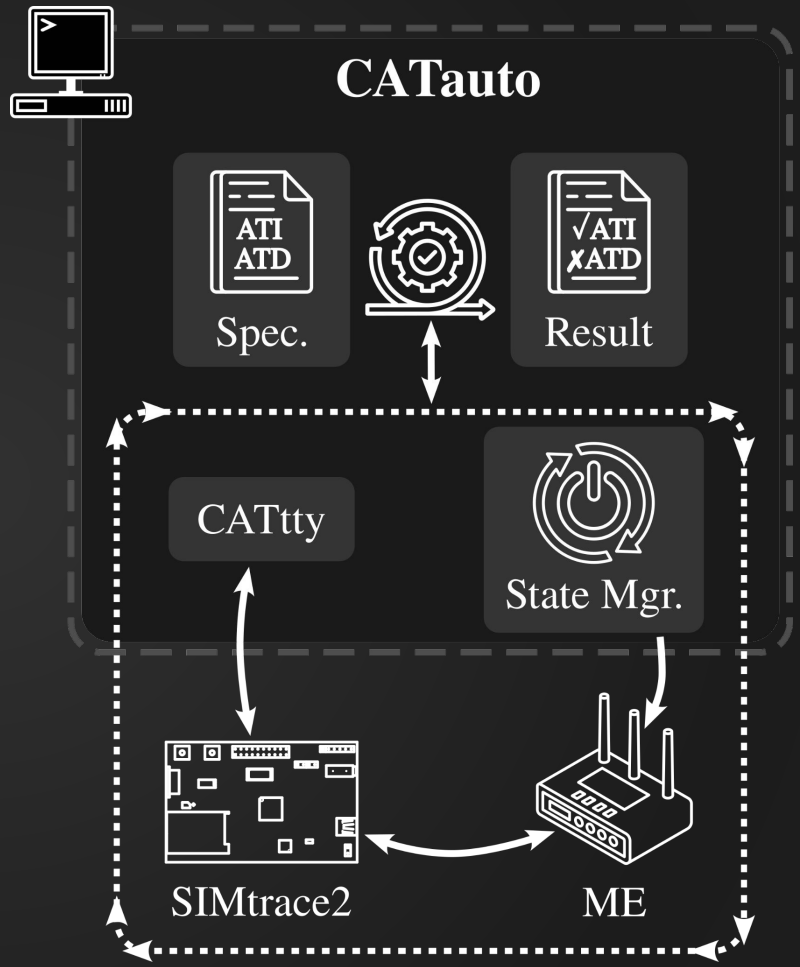
CATana: **CATlet**

- Java Card **Applet**
- **Host app** for sending commands to the applet.
- **AT+CSIM** can deliver raw APDUs
- Bi-directional communication



CATana: **CATauto**

- **Automating** AT command dispatch with CATtty
- Collecting **results**
- **Analysis** using an AT command **definition file**
- Machine-readable AT command definitions



```
info(catlet__host): The "help" command can be used to display usage instructions.
catlet$ sim at "AT"
info(catlet__host): Packet command "at_command": data=CF03024154.
info(catlet__host): SIM envelope transceive: response="910F".
info(catlet__host): Packet command "at_response": data=CF0103.
info(catlet__host): SIM envelope transceive: response="01060D0A4F4B0D0A9000".
info(catlet__host): SIM envelope transceive: response_escape="\x01\x06\r\nOK\r\n\x9
catlet$
```

Demo 3

```
OK

+CSIM: 4,"6108"

OK

+CSIM: 20,"01060D0A4F4B0D0A9000"

OK
```

Did you find interesting attacks?

YES.

(Still under embargo. More information in August.)

Chapter 3:
SIM Exploration++

Thanks NLNet

Learning new Skills

The Dream of Custom SIM Hardware

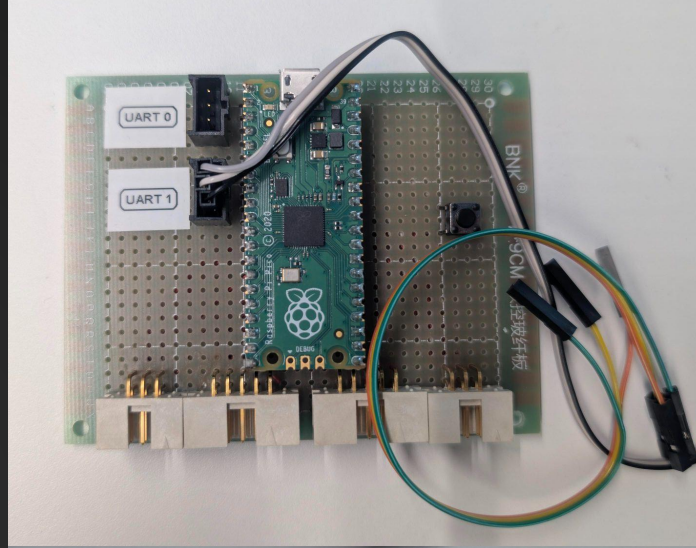
SIMcurity: **SIMuscope**

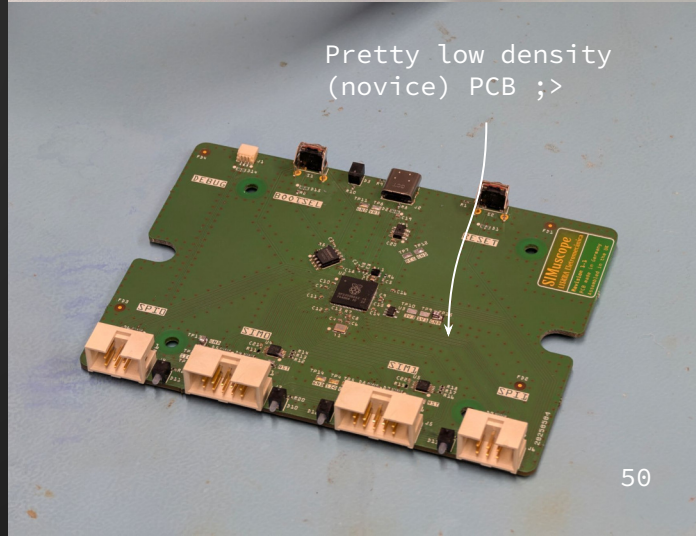
Goal:

- **Physical layer** with **PIO**
- Current sensing, voltage sensing, forcing different voltages from card.

Result:

- RP2350 and RP2040 have different **GPIO block** (fail)
- Voltages too small for RP2350…



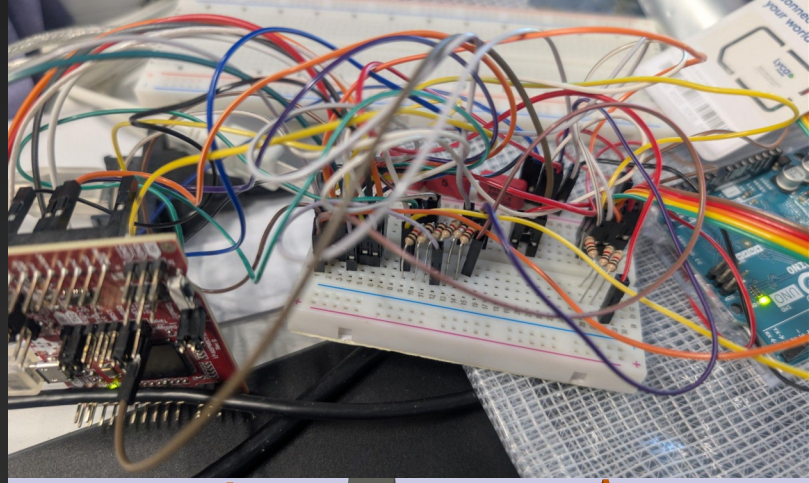


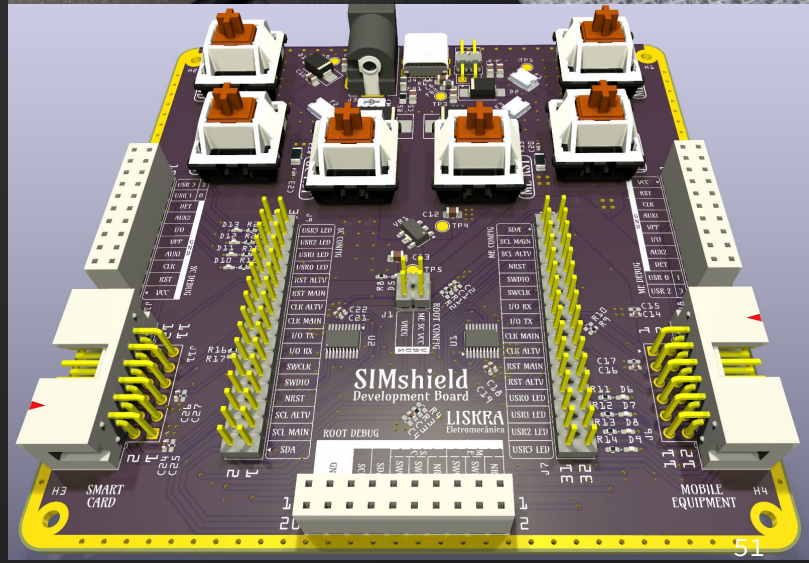
SIMcurity: **SIMshield Dev Board**

Goal: Open Programmable Interposers

- **Hard to flash** with firmware
- 8 pins require **multiplexing** (easy to brick)

- Dev board needed to prove out the tech and develop software
- Mechanical switches instead of buttons ;>



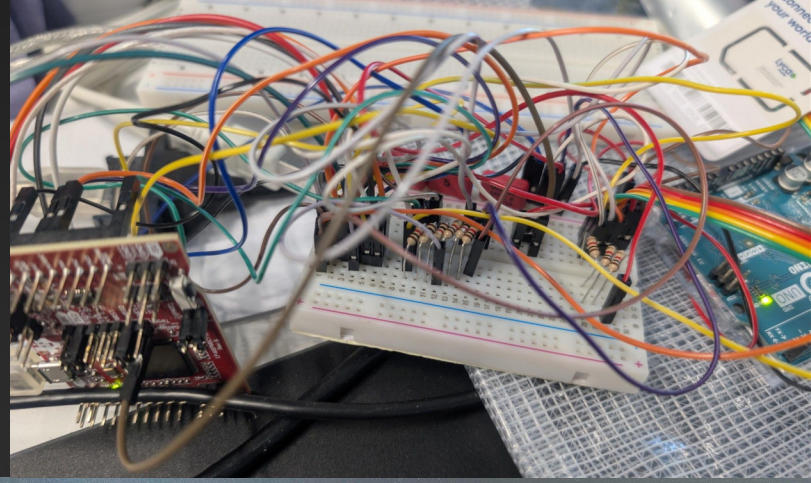


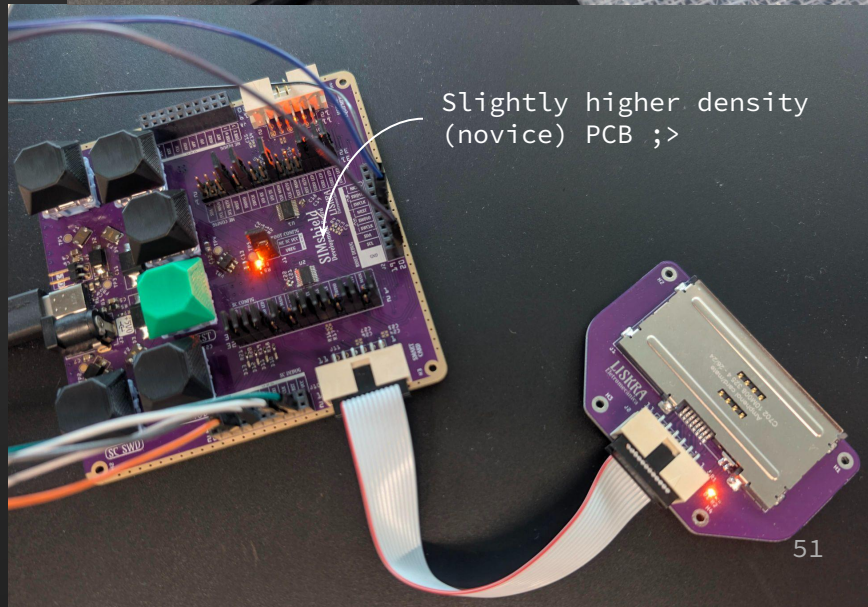
SIMcurity: **SIMshield Dev Board**

Goal: Open Programmable Interposers

- **Hard to flash** with firmware
- 8 pins require **multiplexing** (easy to brick)

- Dev board needed to prove out the tech and develop software
- Mechanical switches instead of buttons ;>

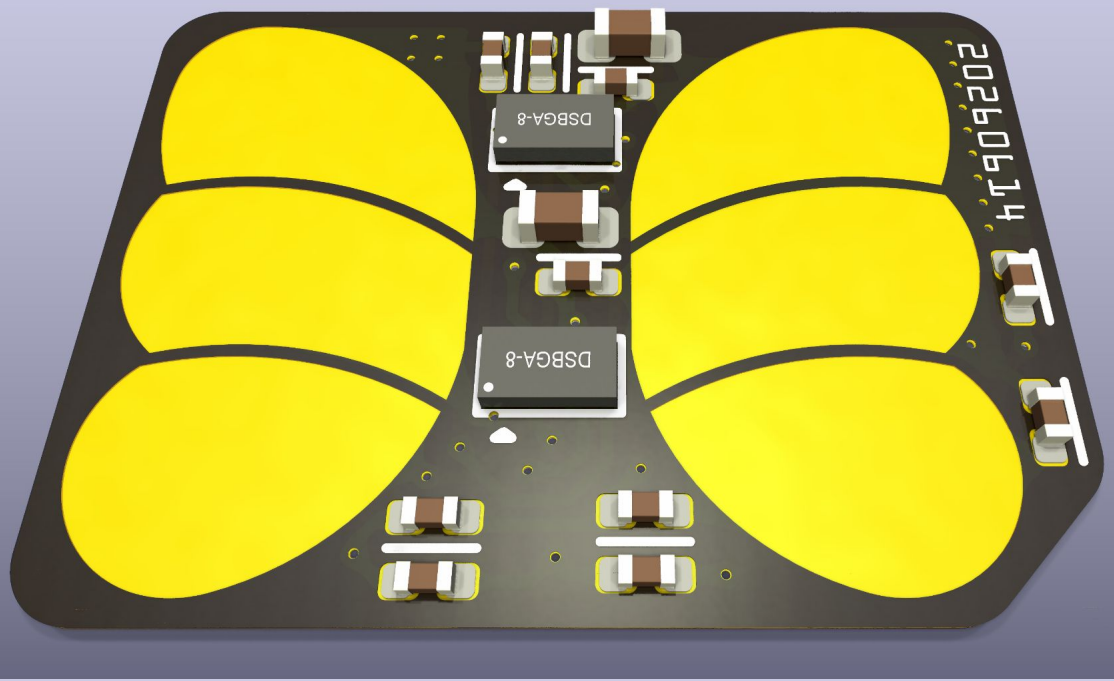


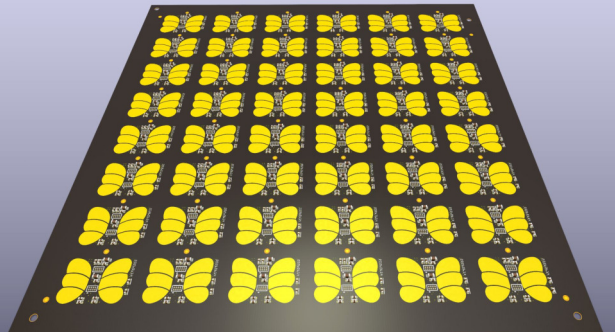


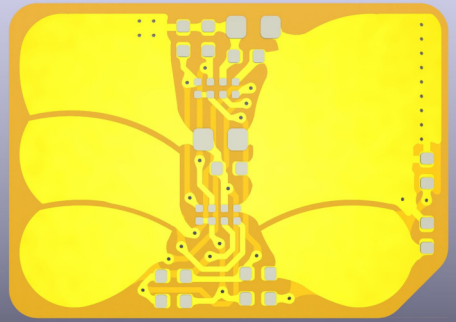
SIMcurity: **SIMshield**

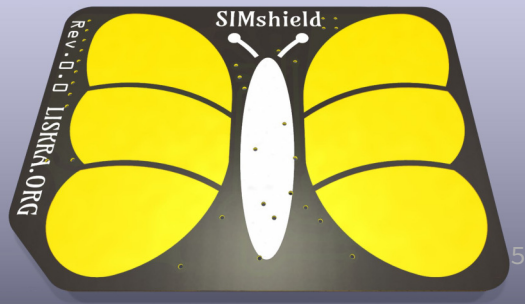
- Sticker that attaches to SIM cards
- Two MCUs forwarding back and forth

Bonus: Learning how to properly panelize PCBs.









Chapter 4: The End?

...

Recap

swICC & swSIM:

- A Smart Card & SIM emulation framework

CATana:

- Making the SIM-AT interface ~~exploitable~~ explorable

SIMurai:

- Framework for exploring hostile SIMs

SIMcurity:

- New REsearch tooling for the SIM interface

Takeaways

SIMs are an attack surface!

- As seen in real-world campaigns

Devices connected to SIMs were not built with this in mind

- Decades of work, yet vulnerabilities remain

Hardware design is fun!

- You should try it too!

Shout-outs & Thanks

Merlin Chlosta:

- SIM expertise (SIMurai & Interposer)

Jinjin Wang:

- Compatibility testing SIMurai

Jiska Classen:

- Brainstorming on LAUNCHing BROWSERs in creative ways

Jacob, Jacqui, Matt:

- Creative help

Osmocom & Sysmocom:

- Tools & guides, kickstarting our journey

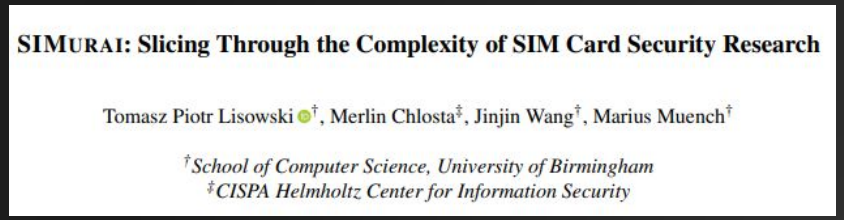
Fuzzware Team:

- Access to IoT devices while developing CATana

NLNet:

- Funding our latest adventures

Where can I learn more?


SIMURAI: Slicing Through the Complexity of SIM Card Security Research

Tomasz Piotr Lisowski[†], Merlin Chlosta[‡], Jinjin Wang[†], Marius Muench[†]

[†]School of Computer Science, University of Birmingham
[‡]CISPA Helmholtz Center for Information Security

SIMurai @ USENIX Sec 2024

SIMcurity @ NLNet

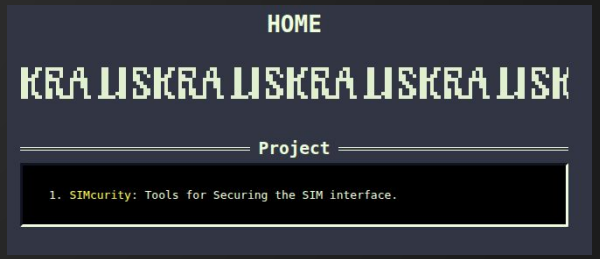

CATANA: On the Dangers of SIM-Originating AT Commands

Tomasz Piotr Lisowski
University of Birmingham

Kristian Covic
Ruhr University Bochum

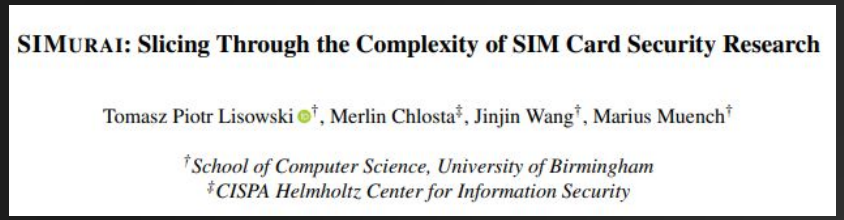
Marius Muench
University of Birmingham

CATana @ WOOT 2026
(To be released)



LISKRA.ORG

Questions?


SIMURAI: Slicing Through the Complexity of SIM Card Security Research

Tomasz Piotr Lisowski[†], Merlin Chlosta[‡], Jinjin Wang[†], Marius Muench[†]

[†]School of Computer Science, University of Birmingham
[‡]CISPA Helmholtz Center for Information Security

SIMurai @ USENIX Sec 2024

SIMcurity @ NLNet

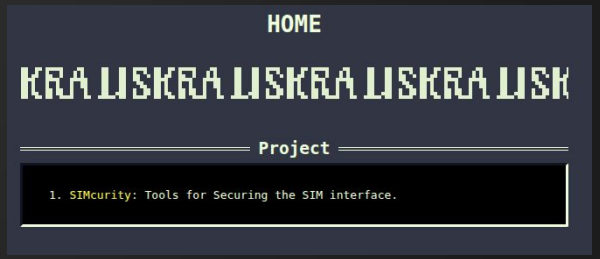

CATANA: On the Dangers of SIM-Originating AT Commands

Tomasz Piotr Lisowski
University of Birmingham

Kristian Covic
Ruhr University Bochum

Marius Muench
University of Birmingham

CATana @ WOOT 2026
(To be released)



LISKRA.ORG

Vulnerabilities in This Presentation

Identifier	Affected Vendor	Protocol Stack	Severity	Estimated No. Affected Chipset Models
CVE-2023-50806	Samsung	Modem	High	14
CVE-2024-45184	Samsung	Modem	High	18
CVE-2024-27209	Google	Modem	High	4
CVE-2025-48618	Google	Android	High	>100