UnZiploc: A bug hunter’s journey from 0-click to platform compromise

Daniel Komaromy
TASZK Security Labs

Lorant Szabo
TASZK Security Labs

TASZK
WE FIND NEEDLES

$whoami

- Mobile security researchers @ TASZK Security Labs
- 50+ CVEs from Mobile OEMs, Pwn2Own, BH/REcon/EkoParty etc.
- QPSI alumn - my 5th Qualcomm Security Summit :)

TASZK
WE FIND NEEDLES

Motivation

- non-baseband, non-app based 0-click
- logic bugs instead of .*-bypass memory corruption chains
- HarmonyOS Bug Bounty Program

TASZK
WE FIND NEEDLES

Attack Vector: Boot chain

- Done: BootROM, Xloader, Fastboot
- Not done: OTA Update

TASZK
WE FIND NEEDLES

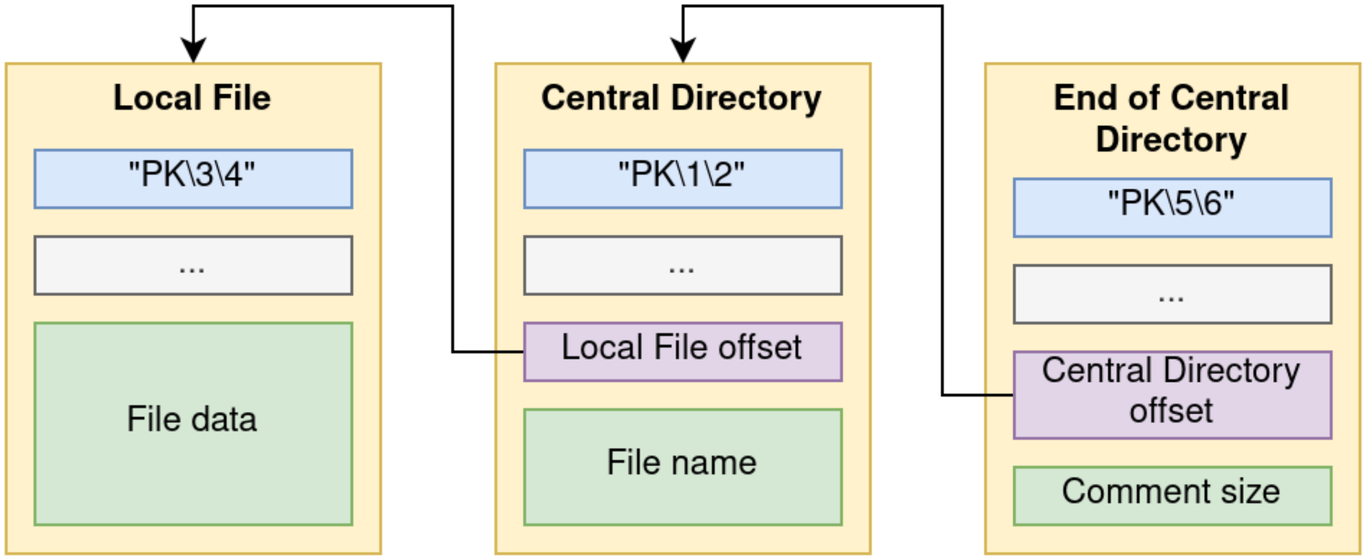
Huawei Firmware Update

- Update method is based on the AOSP OTA procedures
- Update file format is also from AOSP
- Legacy update scheme (no A/B partition)
- The update itself takes place in the recovery/erecovery mode
  - Recovery: only installs an already downloaded update
  - eRecovery: downloads by itself the update via wifi (e.g. when the phone does not boot)
- Dedicated recovery environment (same kernel, but separate ramdisk with tools)

TAS/ZK
WE FIND NEEDLES

Update file format

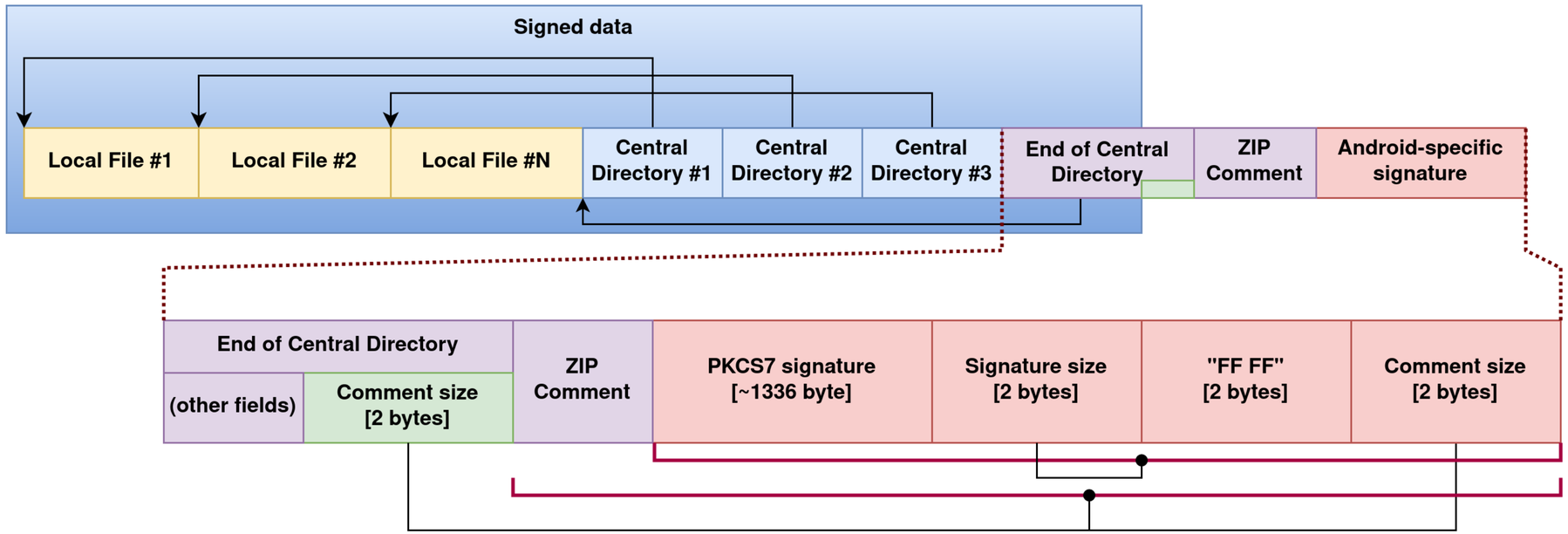
- Format described at AOSP: bootable/recovery/install/verifier.cpp
- Outside: ZIP with extra signature field
- Inside: JAR-like file structure (e.g. META-INF)
- ZIP: a very flexible container format
- Local File: metadata for the actual content
- Central Directory: file listing, points to file
- End of Central Directory: points to central dir



TASZK
WE FIND NEEDLES

Update file signature

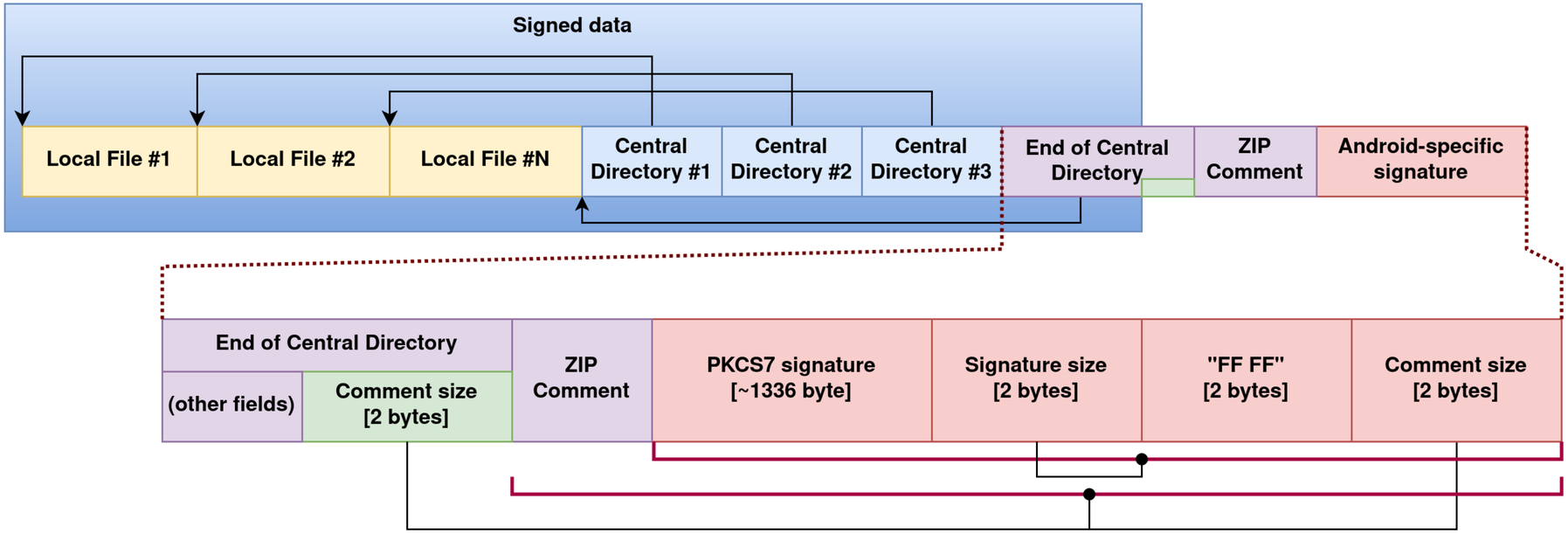
- Located in the comment field, transparent for a ZIP extractor
- signs the whole file except the comment size field
- SHA-256 data hash signed in an RSA-PKCS7 format



TASZK
WE FIND NEEDLES

Signature verification

- Verifier gets the signature based on the footer fields
- Verifies that by going back “comment size” there is in fact a valid EOCD block
- Also checks integrity of the footer and finally the validity of the signature
- minzip handles the actual data extraction of a verified update archive



TASZK
WE FIND NEEDLES

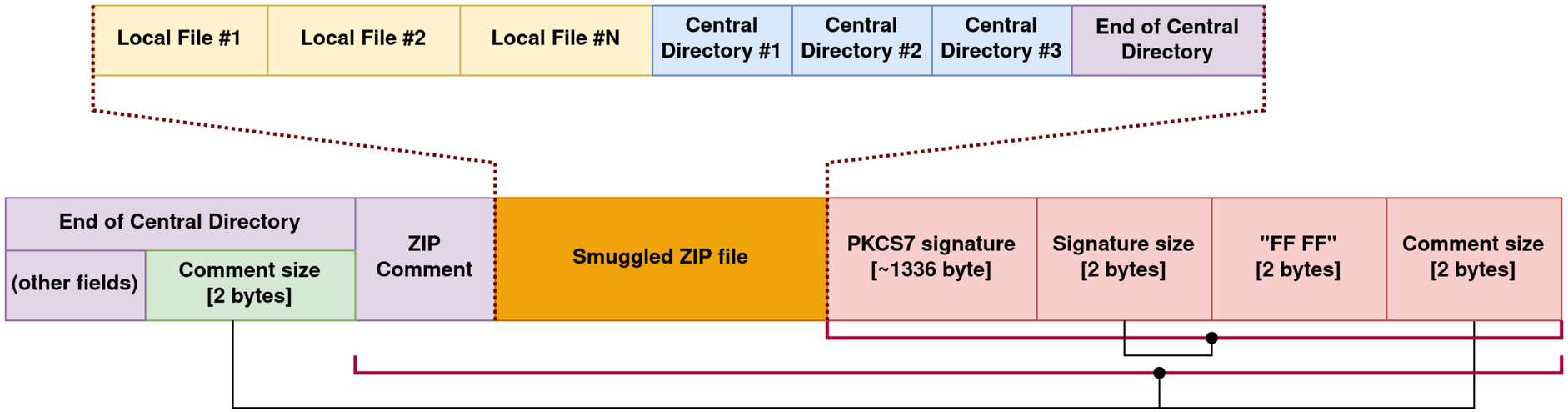
Vulnerability #1: EOCD Smuggling

- ZIP parsers' heuristic discrepancy on finding EOCD signature:
  - verifier code: where the comment size field points (the original, signed one)
  - minzip code: the closest EOCD marker to the end of the ZIP file
- Comment size and signature size are independent
- A gap can be made between the signature and the EOCD block
- Let's smuggle a valid (offset-aligned) ZIP file into the gap

TASZK
WE FIND NEEDLES

Vulnerability #1: EOCD Smuggling

- Split-brain state:
  - The verifier finds the original, signed ZIP
  - The extractor finds the EOCD of the smuggled ZIP



TA$ZK
WE FIND NEEDLES

Vulnerability #1: EOCD Smuggling

- This was known to be a possible vulnerability by upstream
- Huawei used a modified version of the verifier, which didn't have a check
- CVE-2021-40055 (Severity: Critical)

```
187   for (size_t i = 4; i < eocd_size - 3; ++i) {
188     if (eocd[i] == 0x50 && eocd[i + 1] == 0x4b && eocd[i + 2] == 0x05 && eocd[i + 3] == 0x06) {
189       // If the sequence $50 $4b $05 $06 appears anywhere after the real one, libziparchive will
190       // find the later (wrong) one, which could be exploitable. Fail the verification if this
191       // sequence occurs anywhere after the real one.
192       LOG(ERROR) << "EOCD marker occurs after start of EOCD";
193       return VERIFY_FAILURE;
194     }
195   }
```

https://android.googlesource.com/platform/bootable/recovery/+/refs/heads/android12-release/install/verifier.cpp

TASZK
WE FIND NEEDLES

Unzip to RCE

- Once all checks are passed, the actual update is not carried out by the /sbin/recovery executable itself!
- The META-INF/com/google/android/update-binary from update.zip gets executed as root
- The smuggled zip can carry arbitrary content, so RCE is achieved

TASZK
WE FIND NEEDLES

Unzip to RCE

- But first: further verifications on the update.zip contents
  - auth-token: the update server issues this for the recovery on an OTA update
    => create a “skipauth_pkg.tag” empty file in the update archive
  - “META-INF/CERT.RSA” must be present
    => in fact only its size checked, so create a file filled with dummy content
  - The firmware images must contain valid VRL (secboot) headers
    => if there’s no firmware images, no checks happens
  - Firmware version must be applicable for the given device hw/sw revision
    => create a “SOFTWARE_VER_LIST.mbn” file filled with device prefixes

TASZK
WE FIND NEEDLES

Triggering with Local Access

- Memory card (SD/NM card, USB Flash Drive) based update

- Trigger options:

  1) ProjectMenu in Android UI ”4. Software Update → 1. Memory card update” (Dialer: *#*#2846579#*#*)

  2) during boot, button press combo to boot recovery + ”Update mode → Memory card/ OTG update mode”

- Automatically tries to mount external storage -> connecting a USB flash drive with dload/update_sd_base.zip, recovery binary triggers automatically

- Note: this means that physical access (w/o knowing user unlock secret) is enough to get code execution as root in recovery!

TASZK
WE FIND NEEDLES

Triggering Remotely

- Huawei OTA download can happen in 3 ways:
  - eRecovery Wi-Fi update triggered by user
  - Android update triggered by user
  - Android update triggered automatically

TASZK
WE FIND NEEDLES

Vulnerability #2: SSL Bypass

- Android OTA and eRecovery OTA update both ask Huawei query server for available updates
- Served via SSL from query.hicloud.com with cert pinning
- BUT: the query server returns CDN server URLs (update.dbankcdn.com) that are plain HTTP!
- <base URL>/full/filelist.xml contains MD5 hashes of images - also served via HTTP
- Authentication token served via HTTPS – but it can be skipped with the “skipauth_pkg.tag” file
- CVE-2021-40055

TASZK
WE FIND NEEDLES

eRecovery HOTA

- Update query over HTTPS

No.	Time	Source	Destination	Protocol	Length	Info
181	5.913919420	10.42.0.231	10.42.0.1	DNS	77	Standard query 0xffad A query.hicloud.com
182	5.914120166	10.42.0.1	10.42.0.231	DNS	93	Standard query response 0xffad A query.hicloud.com A 80.158.19.121
183	5.922317579	10.42.0.231	80.158.19.121	TCP	74	59608 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=1474245 TSecr=0 WS=256
184	5.952899696	80.158.19.121	10.42.0.231	TCP	66	443 → 59608 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1452 SACK_PERM=1 WS=512
185	5.959320000	10.42.0.231	80.158.19.121	TCP	54	59608 → 443 [ACK] Seq=1 Ack=1 Win=87808 Len=0
186	5.960536961	10.42.0.231	80.158.19.121	TLSv1.2	571	Client Hello
187	5.991392132	80.158.19.121	10.42.0.231	TCP	54	443 → 59608 [ACK] Seq=1 Ack=518 Win=30720 Len=0
188	5.992137743	80.158.19.121	10.42.0.231	TLSv1.2	1506	Server Hello
189	5.992558126	80.158.19.121	10.42.0.231	TCP	1506	443 → 59608 [ACK] Seq=1453 Ack=518 Win=30720 Len=1452 [TCP segment of a reassembled PDU]
190	5.992647001	80.158.19.121	10.42.0.231	TLSv1.2	449	Certificate, Server Key Exchange, Server Hello Done
191	5.996051560	10.42.0.231	80.158.19.121	TCP	54	59608 → 443 [ACK] Seq=518 Ack=1453 Win=90624 Len=0
192	5.998226219	10.42.0.231	80.158.19.121	TCP	54	59608 → 443 [ACK] Seq=518 Ack=2905 Win=93440 Len=0
193	5.998668896	10.42.0.231	80.158.19.121	TCP	54	59608 → 443 [ACK] Seq=518 Ack=3300 Win=96512 Len=0
194	6.052105003	10.42.0.231	80.158.19.121	TLSv1.2	147	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message

- Download OTA update over HTTP

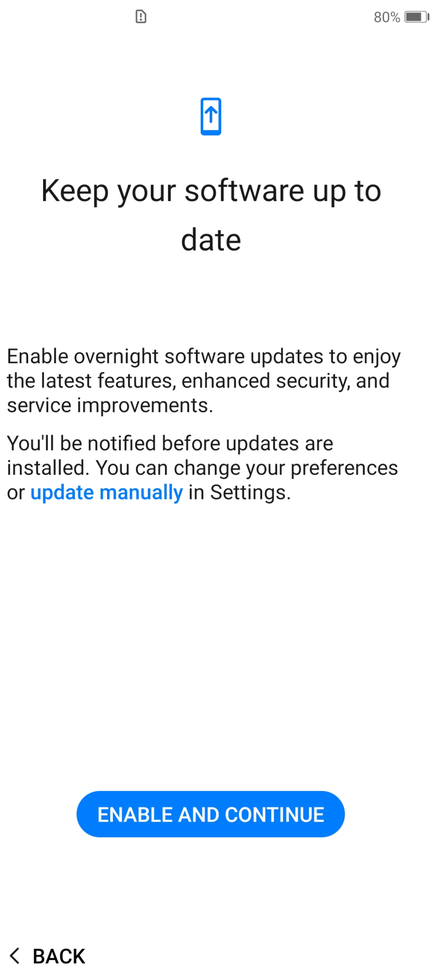
tcp.stream eq 15

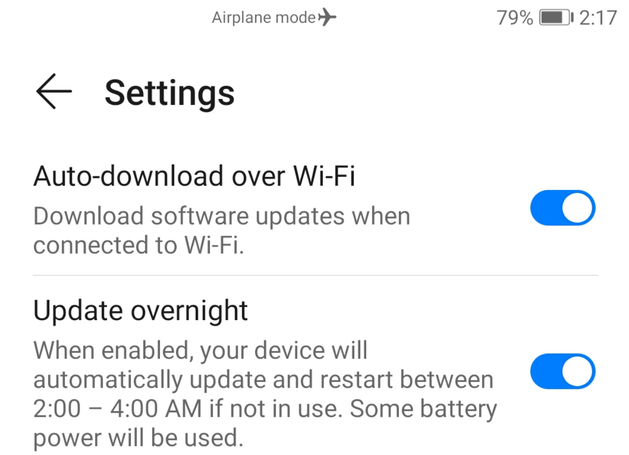
No.	Time	Source	Destination	Protocol	Length	Info
886	134.527627635	10.42.0.231	10.42.0.1	TCP	74	35032 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 TSval=1602853 TSecr=0 WS=256
890	134.574477943	10.42.0.1	10.42.0.231	TCP	66	80 → 35032 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1452 SACK_PERM=1 WS=1024
891	134.584662381	10.42.0.231	10.42.0.1	TCP	54	35032 → 80 [ACK] Seq=1 Ack=1 Win=87808 Len=0
892	134.585921404	10.42.0.231	10.42.0.1	HTTP	245	GET /download/data/pub_13/HWHOTA_hota_900_9/58/v3/Sdd5h01LThCk2OOBkSwD_Q/full/update_base.zip HTTP/1.1
898	134.631348853	10.42.0.1	10.42.0.231	TCP	54	80 → 35032 [ACK] Seq=1 Ack=192 Win=30720 Len=0
900	134.634572157	10.42.0.1	10.42.0.231	TCP	1506	80 → 35032 [ACK] Seq=1 Ack=192 Win=30720 Len=1452 [TCP segment of a reassembled PDU]
901	134.634934077	10.42.0.1	10.42.0.231	TCP	1506	80 → 35032 [ACK] Seq=1453 Ack=192 Win=30720 Len=1452 [TCP segment of a reassembled PDU]
902	134.634944417	10.42.0.1	10.42.0.231	TCP	1506	80 → 35032 [ACK] Seq=2905 Ack=192 Win=30720 Len=1452 [TCP segment of a reassembled PDU]
903	134.635228890	10.42.0.1	10.42.0.231	TCP	1506	80 → 35032 [ACK] Seq=4357 Ack=192 Win=30720 Len=1452 [TCP segment of a reassembled PDU]
904	134.635248417	10.42.0.1	10.42.0.231	TCP	1506	80 → 35032 [ACK] Seq=5809 Ack=192 Win=30720 Len=1452 [TCP segment of a reassembled PDU]
905	134.635255825	10.42.0.1	10.42.0.231	TCP	1506	80 → 35032 [ACK] Seq=7261 Ack=192 Win=30720 Len=1452 [TCP segment of a reassembled PDU]
906	134.635262682	10.42.0.1	10.42.0.231	TCP	1506	80 → 35032 [ACK] Seq=8713 Ack=192 Win=30720 Len=1452 [TCP segment of a reassembled PDU]
907	134.635269634	10.42.0.1	10.42.0.231	TCP	1506	80 → 35032 [ACK] Seq=10165 Ack=192 Win=30720 Len=1452 [TCP segment of a reassembled PDU]
908	134.635276582	10.42.0.1	10.42.0.231	TCP	1506	80 → 35032 [ACK] Seq=11617 Ack=192 Win=30720 Len=1452 [TCP segment of a reassembled PDU]
909	134.635283807	10.42.0.1	10.42.0.231	TCP	1506	80 → 35032 [ACK] Seq=13069 Ack=192 Win=30720 Len=1452 [TCP segment of a reassembled PDU]

TASZK
WE FIND NEEDLES

Remote Attack: Automatic Trigger

- Timezone automatic determination: GPS, mobile carrier country code of serving cell
- NTP server updates happen daily & after reboots
- GSM network can push time setting with network IE
- All these can be spoofed

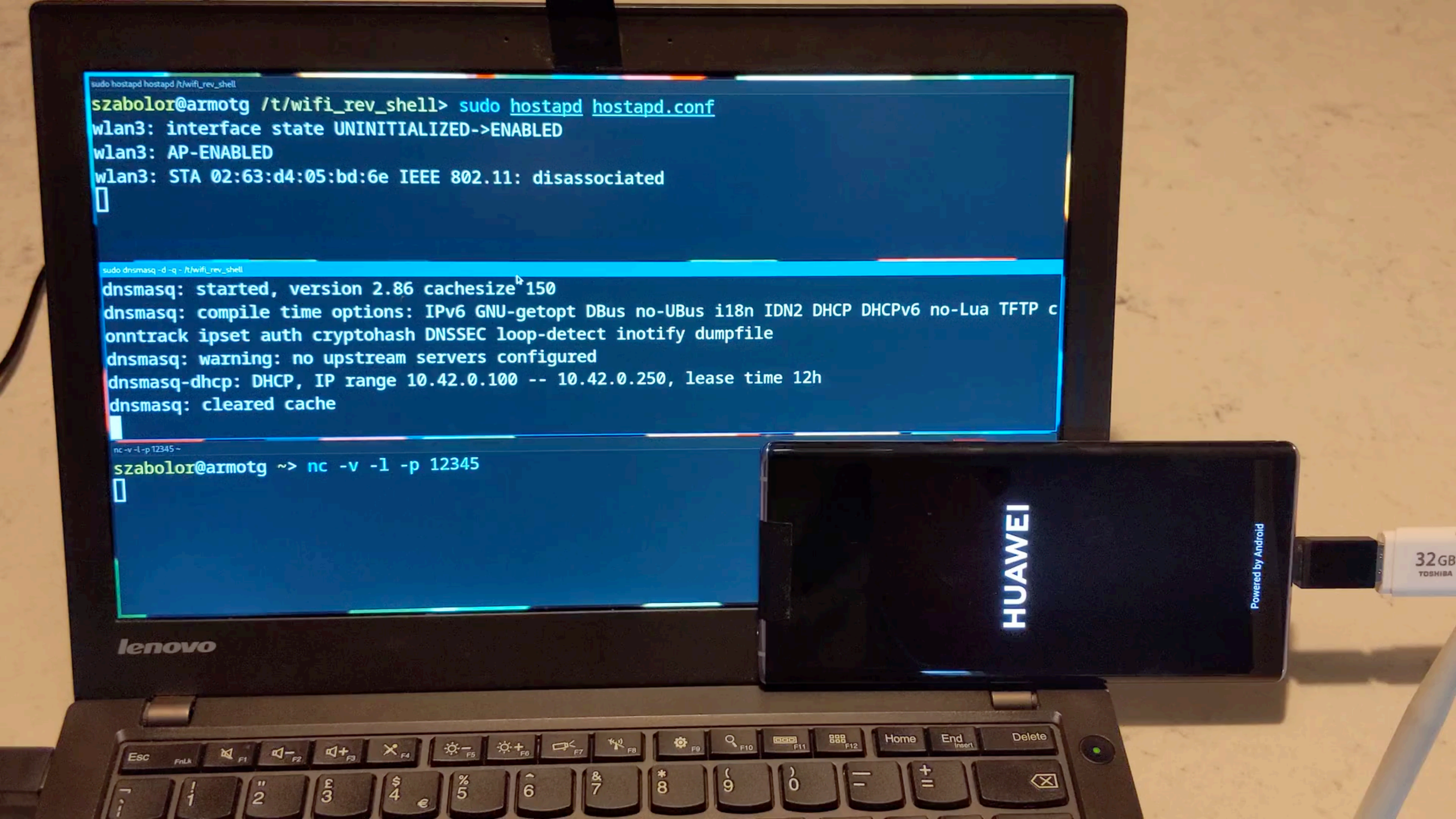




TA$ZK
WE FIND NEEDLES

Remote shell vs wpa-supplicant

- In the Android OTA cases, the entire download happens in Android not recovery
- Recovery is allowed to access Wi-Fi (to support eRecovery OTA), so the attacker update-binary can bring up Wi-Fi itself
- But wpa-supplicant_hisi checks ro.enter_erecovery system property and exits if it is not eRecovery -> SD card or Android OTA update mode can’t bring up Wi-Fi
- Since the modified update-binary runs as root -> patch wpa_supplicant_hisi binary or use LD_PRELOAD to bypass the getprop check


sudo hostapd hostapd /t/wifi_rev_shell
szabolor@armotg /t/wifi_rev_shell> sudo hostapd hostapd.conf
wlan3: interface state UNINITIALIZED->ENABLED
wlan3: AP-ENABLED
wlan3: STA 02:63:d4:05:bd:6e IEEE 802.11: disassociated
sudo dnsmasq -d -q - /t/wifi_rev_shell
dnsmasq: started, version 2.86 cachesize 150
dnsmasq: compile time options: IPv6 GNU-getopt DBus no-UBus i18n IDN2 DHCP DHCPv6 no-Lua TFTP c
onntrack ipset auth cryptohash DNSSEC loop-detect inotify dumpfile
dnsmasq: warning: no upstream servers configured
dnsmasq-dhcp: DHCP, IP range 10.42.0.100 -- 10.42.0.250, lease time 12h
dnsmasq: cleared cache
nc -v -l -p 12345 ~
szabolor@armotg ~> nc -v -l -p 12345
lenovo
HUAWEI
Powered by Android
32GB
TOSHIBA

TASZK
WE FIND NEEDLES

Escalation Options

- # … now what?
- Recovery != Android
- Goal: user data
- FDE keys derived in TrustZone
- TrustZone takeover possible pathways
  - Attack TAs: SELinux allows update-recovery to talk to TAs by design
  - Overwrite low-level images on storage - secure boot prevents this
  - Take over the kernel runtime, attack the platform from kernel

TASZK
WE FIND NEEDLES

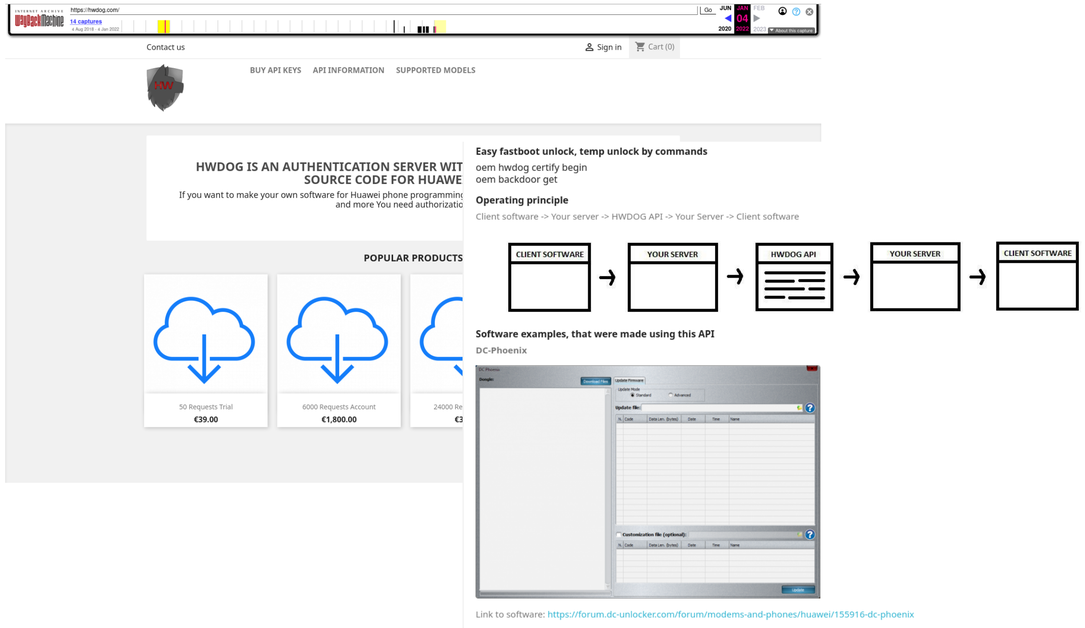
Escalation Via Kernel Route

- Old-school options (kexec, insmod) - stopped by SELinux
- Patch kernel image on storage - stopped by AVB
- N-day kernel LPE + old kernel - stopped by … ?
- 0-day kernel LPE

TASZK
WE FIND NEEDLES

Fastboot Unlock

- Known Secure Boot vulnerabilities [BH 2021]: needed USB path, patched
- Fastboot bootloader unlock: flags (FBLock, userlock) stored in NVME partition
- Writable - but signed with an RSA key, public key burnt into Fastboot image
- RSA private key was …. leaked?
- … let’s try something legal instead :)



TASZK
WE FIND NEEDLES

Kernel N-day

- Find a suitable known vulnerability
  - 2021 June: NPU driver vulnerabilities (CVE-2021-22388, CVE-2021-22389, CVE-2021-22390, CVE-2021-22412, CVE-2021-22415)
  - https://labs.taszk.io/articles/post/exploiting_huaweis_npu_driver/
  - Device node restricted - but we are running as #
- The bug is fixed - but we can write all partitions
- Image rewrite needs reboot to take effect: how do we preserve code exec?
- Solution: if update-recovery fails, after reboot recovery is started again

TASZK
WE FIND NEEDLES

Kernel Flashing vs AVB

- Android Verified Boot uses vbmeta partition for boot.img verification
- vbmeta partition is protected by Huawei secure boot (VRL Header); contains public keys for signing kernel, ramdisk, recovery, etc.
- But the vbmeta partition only changes with major OS updates!
  - e.g. Android 10 & 10.1 are the same
- Simply flash previous (e.g. 12 months old) boot.img

TASZK
WE FIND NEEDLES

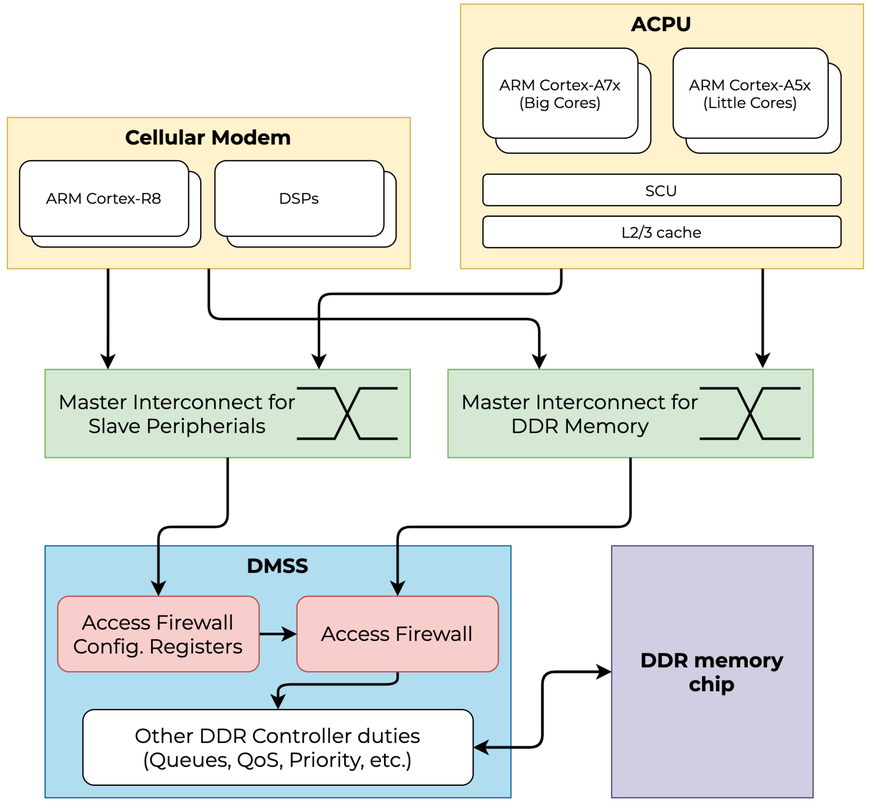
Kernel 0-day

- https://consumer.huawei.com/en/support/bulletin/2022/4/
- CVE-2022-22252: UAF vulnerability in the DFX module (Severity: Critical)
- Vulnerability details not public yet

TASZK
WE FIND NEEDLES

Return of the DMSS bypass I.

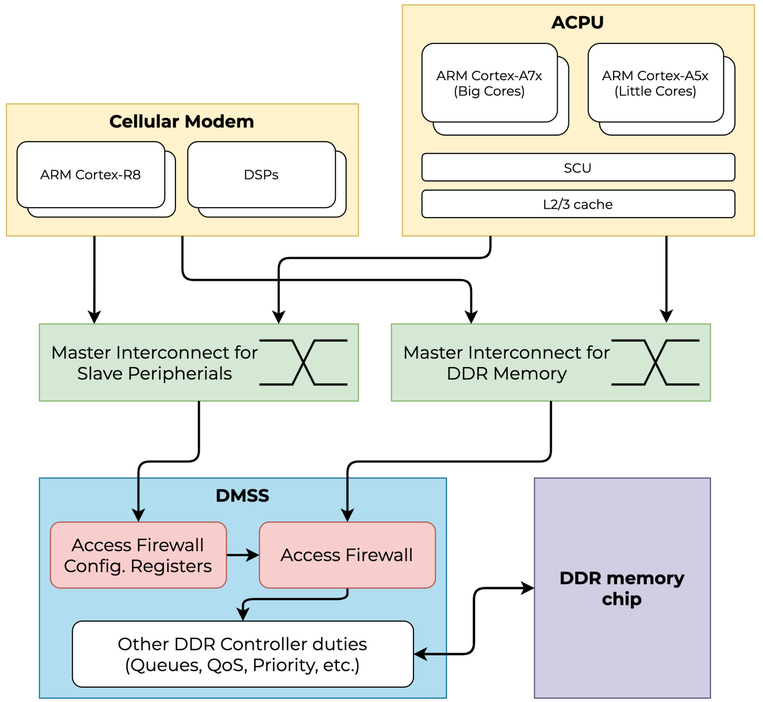
- nSW EL1 write - now what? We want the 90s, not exploit mitigations
- DMSS: a DDR Controller with a “Memory Firewall” built-in
- Programmed via ASI entries
- CVE-2021-22432: IOMCU DMA allowed to access TrustZone memory [BH 2021]
- Huawei patch fixed the ASI entry for IOMCU DMA



TASZK
WE FIND NEEDLES

Return of the DMSS bypass I.

- What do other ASI entries allow?
- What additional cores / DMA-capable peripherals are there in the SoC?
- ASI entry reversing: core and peripheral Master IDs found in drivers/hisi/ap/platform/kirin990/soc_mid.h




```
DMSS entry address | index | range begin-end | (N)S: (non)secure R/W | AXI Master ID for (W)rite and (R)ead
0xffe82e30 19: 0x13000000 - 0x135fffff SR SW W00010020 R00010020
0xffe82e40 20: 0x13600000 - 0x192fffff SR SW W00010420 R00010420
```

TASZK
WE FIND NEEDLES

More Insecure ASI Configuration

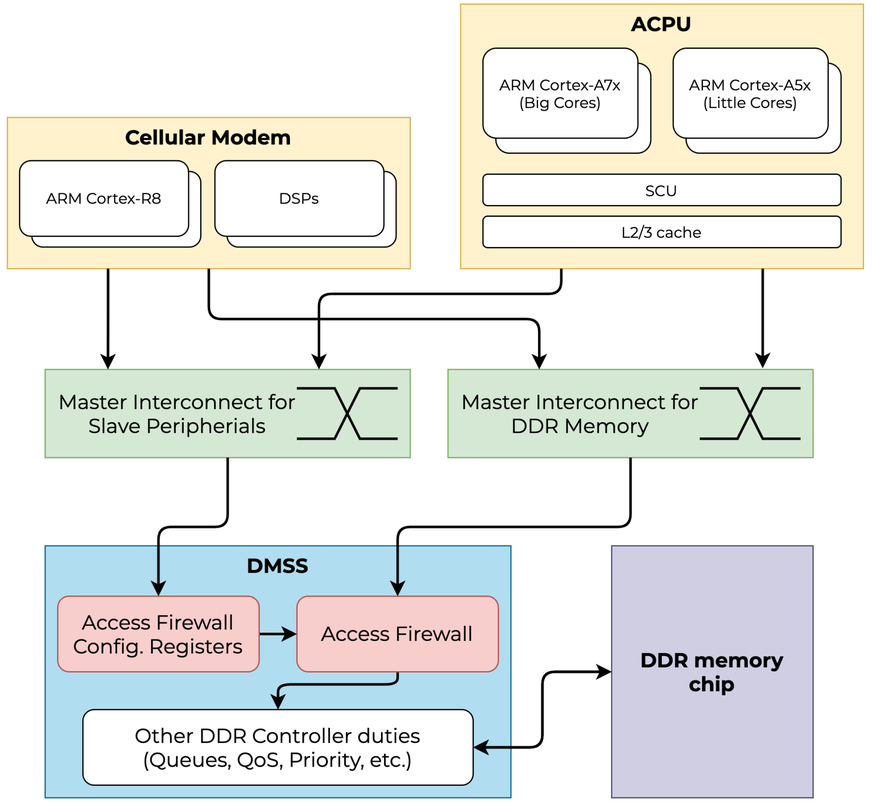
- Peripheral DMA (drivers/dma/hisi_dma_64.c)
- Full secure read-write access to the trustfirmware (0x13000000-0x135fffff) and the teeos and the trustlets (0x13600000-0x192fffff) memory ranges
- CVE-2021-39992

```
static struct dma_async_tx_descriptor *hisi_dma_prep_memcpy(
  struct dma_chan *chan, dma_addr_t dst, dma_addr_t src, size_t
len, unsigned long flags)
{
  ...
  if (!c->ccfg) {
    /* default is memtomem, without calling device_control */
    c->ccfg = CCFG_SRCINCR | CCFG_DSTINCR | CCFG_EN;
    /* burst : 16 */
    c->ccfg |= (0xf << DMA_CH_CFG_DL) | (0xf << DMA_CH_CFG_SL);
    /* width : 64 bit */
    c->ccfg |= (DMA_2BITS_MASK << DMA_CH_CFG_DW) |
      (DMA_2BITS_MASK << DMA_CH_CFG_SW);
  }
  ...
}
```

TASZK
WE FIND NEEDLES

Return of the DMSS bypass II.

- CVE-2021-22431: Modem had direct access to ASI entries [BH 2021]
- ASI entries control DDR access but are not in DDR so the protection of it seemed to be a hole in the SoC fabric security design itself
- However: the Huawei patch also removed the modem's access to DMSS entries
- Clearly we missed something the first time



TASZK
WE FIND NEEDLES

The DMSS Access Patch

- Diff the Huawei patches
  - https://labs.taszk.io/articles/post/huawei_kirin990_bootrom_patch/
- New code appeared in EL3 secure monitor
- Copies a predefined table to the “CFGBUS_Service_Target” register region (Kernel source to the rescue again)
- Resembles three distinct groups (experimentally verified):
  - 1: removes modem access to DMSS config registers
  - 2: removes modem access to CFGBUS_Service_Target registers
  - 3: still unknown
- Looks like a NoC structure used inside the SoC, where the modem CPU and the main memory (DDR) are a source and a target node
- Looks like the NoC groups are tied to Master IDs - perhaps the patch only thought of adding restrictions for the modem?

```
#define SOC_ACPU_DMA_NOC_Service_Target_BASE_ADDR (0xFE260000)
#define SOC_ACPU_CFGBUS_Service_Target_BASE_ADDR (0xFE250000)
```

```
…
[0xFE252080]: 0x257000
[0xFE252084]: 0x7FFE00
[0xFE252088]: 0x0
[0xFE25208C]: 0x0
[0xFE252090]: 0x14
[0xFE252094]: 0x0
[0xFE252098]: 0x0
[0xFE25209C]: 0x3
[0xFE2520A0]: 0x3
[0xFE2520A4]: 0xF
[0xFE2520A8]: 0x0
[0xFE2520AC]: 0x10000
[0xFE2520B0]: 0x0
…
```

```
          unknown        DMSS       CFGBUS
+0x00:   0x257000    0x3D7000    0x002000 | offset
+0x04:   0x7FFE00    0x7FFE00    0x03FE00 | base
+0x08:        0x0         0x0        0x2000 |
+0x0C:        0x0         0x0           0x0 |
+0x10:       0x14        0x14           0x0A | log2(size)
+0x14:        0x0         0x0           0x0 |
+0x18:        0x0         0x0           0x0 |
+0x1C:        0x3         0x3           0x2 | {!write,!read}
+0x20:        0x3         0x3           0x3 |
+0x24:        0xF         0xF           0xF |
+0x28:        0x0         0x0           0x0 |
+0x2C:    0x10000     0x10000       0x10000 |
+0x30:        0x0         0x0           0x0 |
```

TASZK
WE FIND NEEDLES

More Insecure ASI Access

- CVE-2021-39991: The Peri DMA can rewrite ASI entries
- CVE-2021-39986: The ASP DMA can rewrite ASI entries
- CVE-2021-37115: The modem can overwrite the SRAM of the LPMCU …. and the LPMCU can rewrite ASI entries

```
CVE-2021-39992: Improper security permission configuration vulnerability on ACPU Severity: High
CVE-2021-39991: Unauthorized rewriting vulnerability with the memory access management module on ACPU Severity: High
CVE-2021-39986: Unauthorized rewriting vulnerability with the memory access management module on ACPU Severity: High
CVE-2021-37115: Unauthorized rewriting vulnerability with the memory access management module on ACPU Severity: High
CVE-2021-37109: Security protection bypass vulnerability with the modem Severity: High
CVE-2021-37107: Improper memory access permission configuration on ACPU Severity: High
```

TASZK
WE FIND NEEDLES

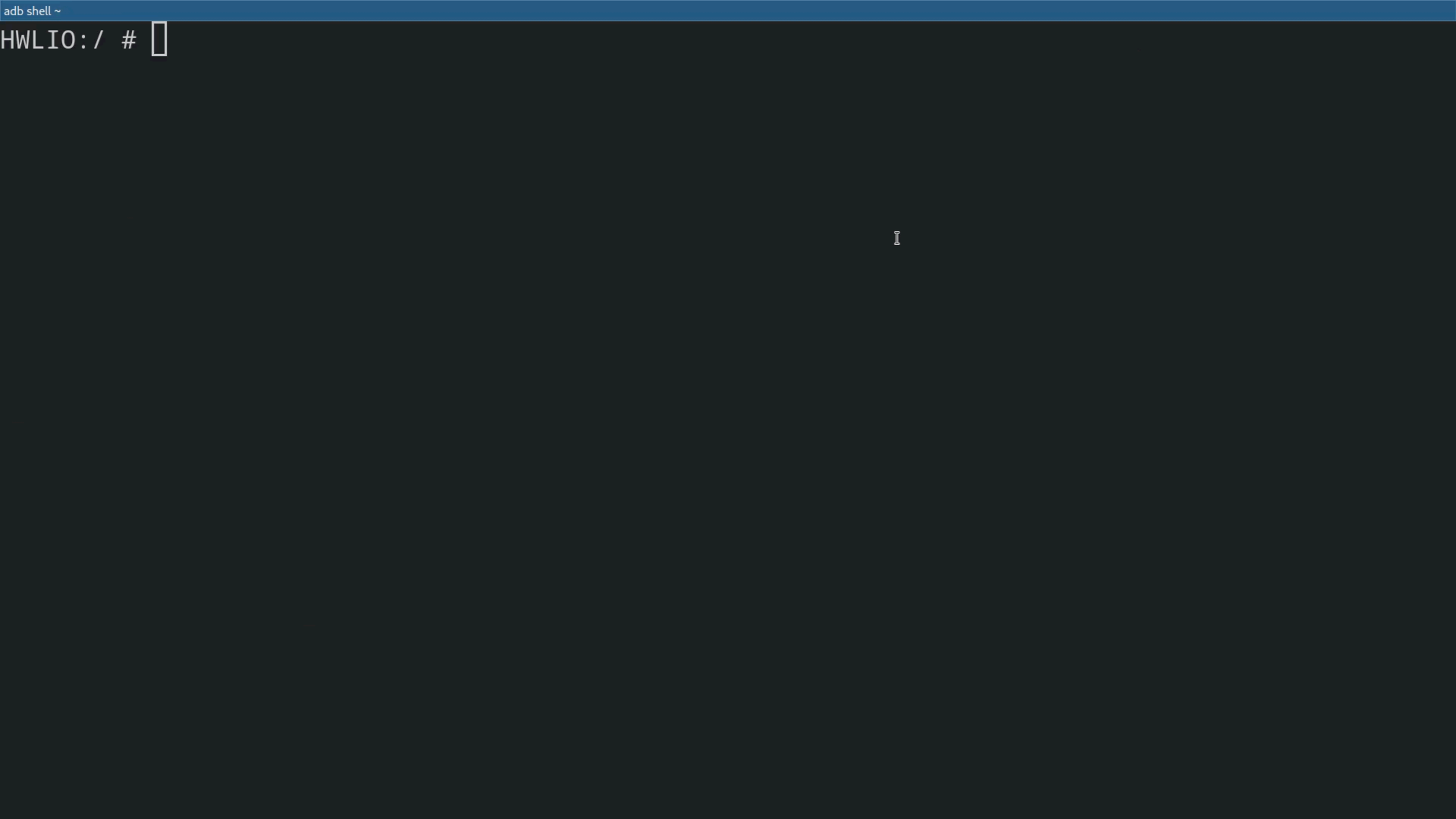
But we are not the modem?

- Unlike BH, we are not looking for baseband SBX vulnerabilities, needed to trigger from the ACPU side
- The modem EDMA can write all modem AND LPMCU memory
- This not only bypasses the modem MPU … (CVE-2021-37109)
- The EDMA control registers are programmable by the kernel too (CVE-2021-37107)
- Fine print: need to have modem powered and overcome the limited memory mapping view of the LPMCU

TASZK
WE FIND NEEDLES

TZ code exec: now what?

- Deriving FDE keys in Huawei TrustZone [See WOOT 2020 Busch et al.]
- Once we have code execution in the TEE, we can offline bruteforce the PIN input and recover FDE keys


adb shell ~
HWLIO:/ #

TASZK
WE FIND NEEDLES

Disclosure Process

- Huawei Bug Bounty Program
- Every reported bug fixed via Huawei OTA update (security bulletins: 2022 February, March, April)

TASZK
WE FIND NEEDLES

Thank you!

Daniel Komaromy
daniel@taszk.io
Twitter: @kutyacica

Lorant Szabo
szabolor@taszk.io
Twitter: @szabolor