Checkmate Mate30

Attack the bootrom of Huawei Smartphones

Slipper & Guanxing Wen

BIO

- Guanxing Wen is member of Pangu Team. In recent years, his primary interests are low-level security, such as kernel, bootloader, trustzone. Previously, his work has been presented at Infiltrate, Black Hat EU, 44CON, etc.

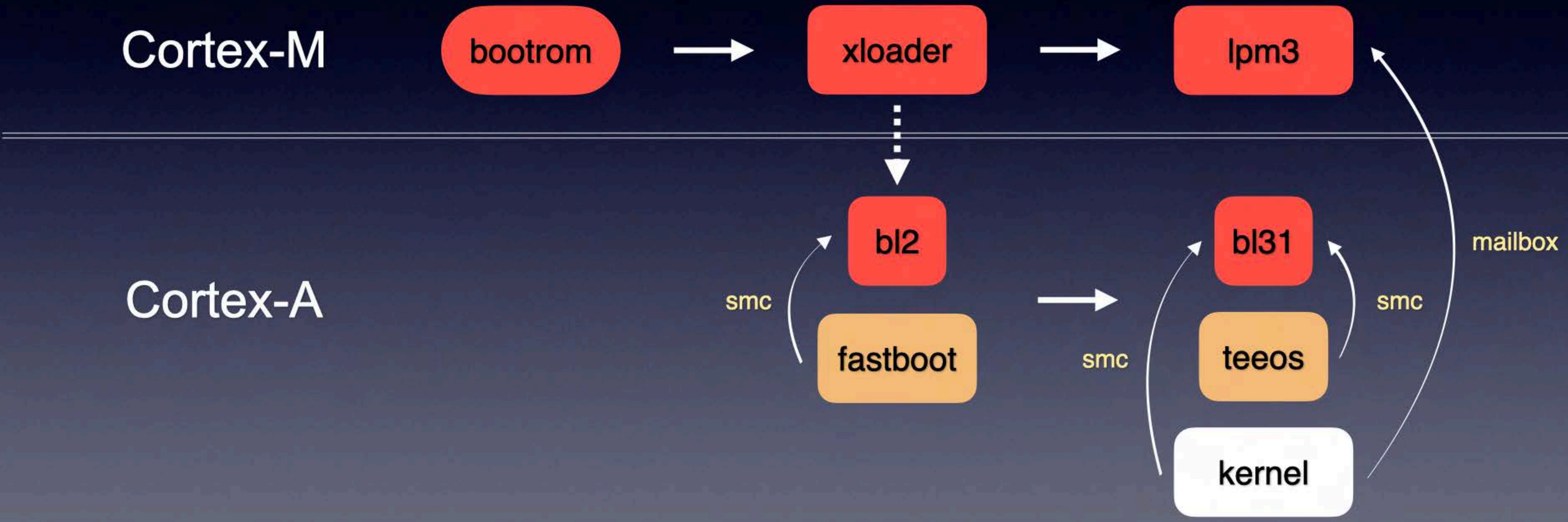
- Slipper, a security researcher from Pangu Team. He used to play in hacking games like Pwn2Own/DEFCON CTF/GeekPWN/TianfuCup. He has hacked many targets in public: iPhone8/PlayStation4/Cisco ASA/ Safari/Firefox/MacOS/Docker/Cent OS/Ubuntu/Adobe Reader. Sometimes he livestreams hackings.

AGENDA

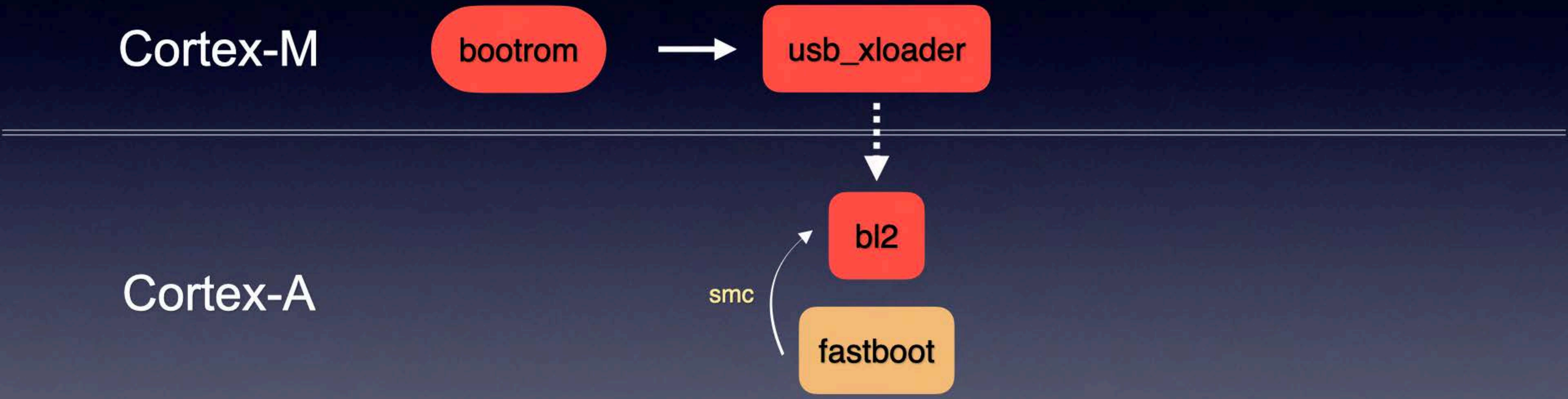
- SECURE BOOTCHAIN
- BOOTROM 101
- VULNERABILITIES
- DEMOS
  - UNLOCK BOOTLOADER
  - EL3 ROOT SHELL
  - JTAG DEBUGGING

Secure Boot Chain

Secure Boot Chain



Secure Boot Chain (download)



Bootrom 101

Bootrom 101

- OCR (On-Chip Rom) for Huawei
  - PBL for Qualcomm
  - SecureROM for Apple
- Cortex M3
  - Thumb
  - SRAM only

Bootrom 101

```
00000000                AREA ROM, CODE, READWRITE, ALIGN=0
00000000                CODE16
00000000                DCD 0x5D3FC             ; stack_base + sizeof(stack_base)
00000004                DCD reset_handler+1     ; ResetISR
00000008                DCD dead_loop+1         ; NmiSR
0000000C                DCD dead_loop+1         ; FaultISR
```

Start	End	Usage	Permission
0x00000000	0x00020000	bootrom	RE
0x00020000	0x00020E38	usb_dev	RWE
0x00020E38	0x00022000	data	RWE
0x00022000	0x0005D000	xloader	RWE
0x0005D000	0x0005D400	stack	RWE

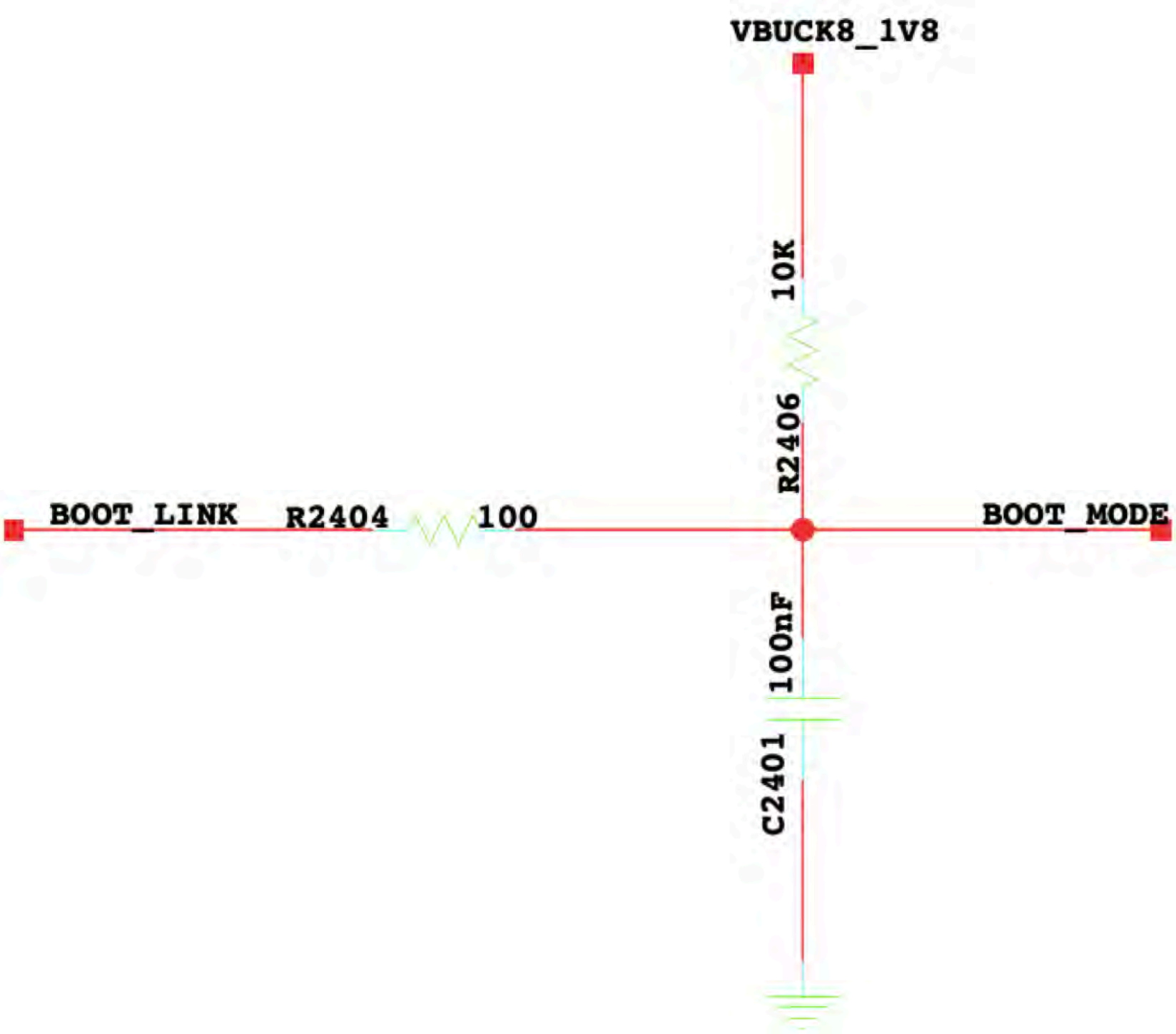
Bootmode

```
if ( is_download )
{
  set_boot_state(0x1F);
  is_normal = 0;
  is_download = 1;
}
else
{
  is_normal = check_bootmode();
}
if ( is_normal == 1 )
{
  if ( !load_disk(v3) )
    goto LABEL_12;
  goto LABEL_10;
}
if ( !is_normal )
{
  while ( 1 )
  {
    v9 = usb_download(0x22000);
    if ( v9 )
    {
      v18 = cprintf("usb download err:%d\n", v9);
```

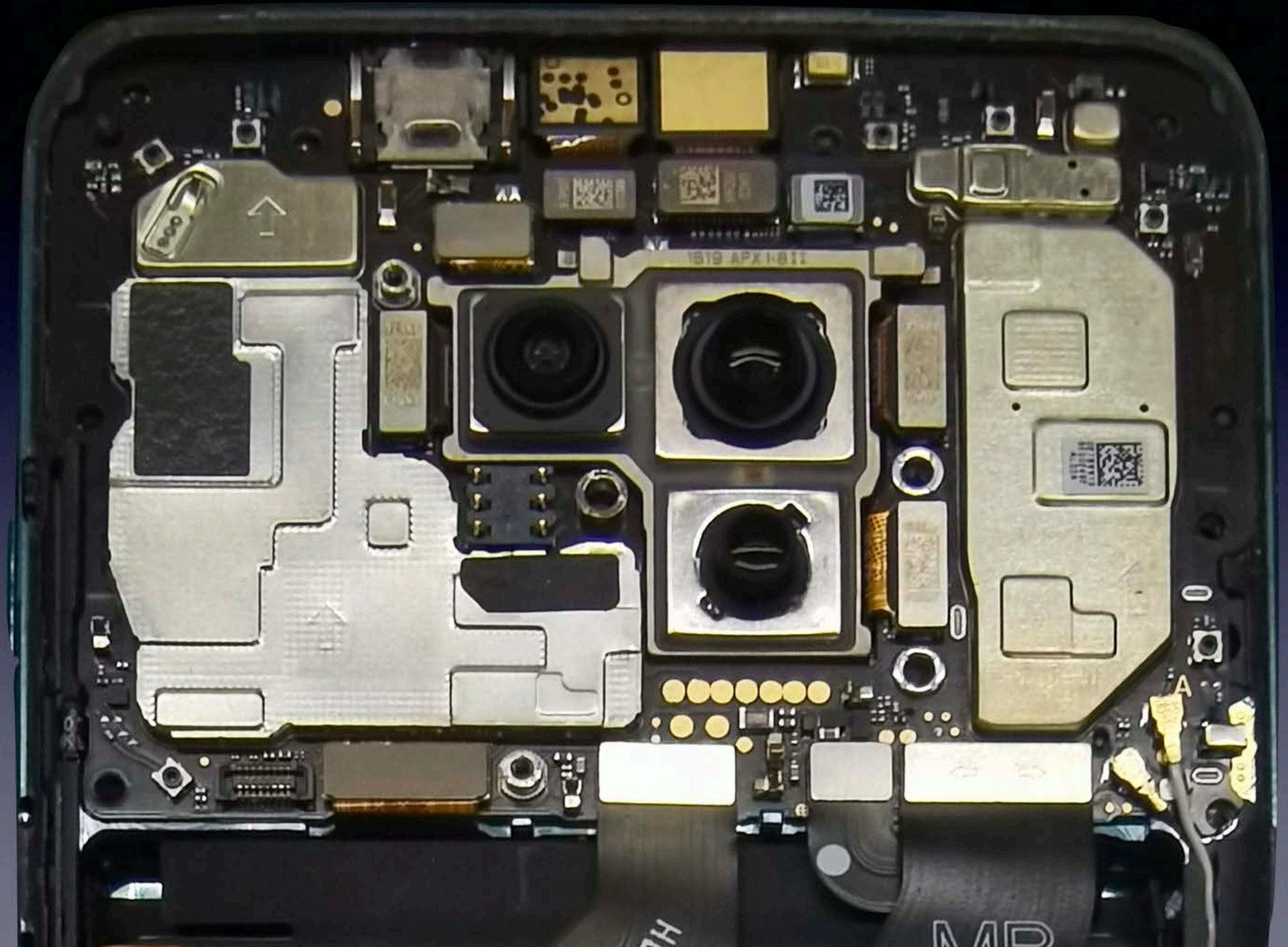
Bootmode

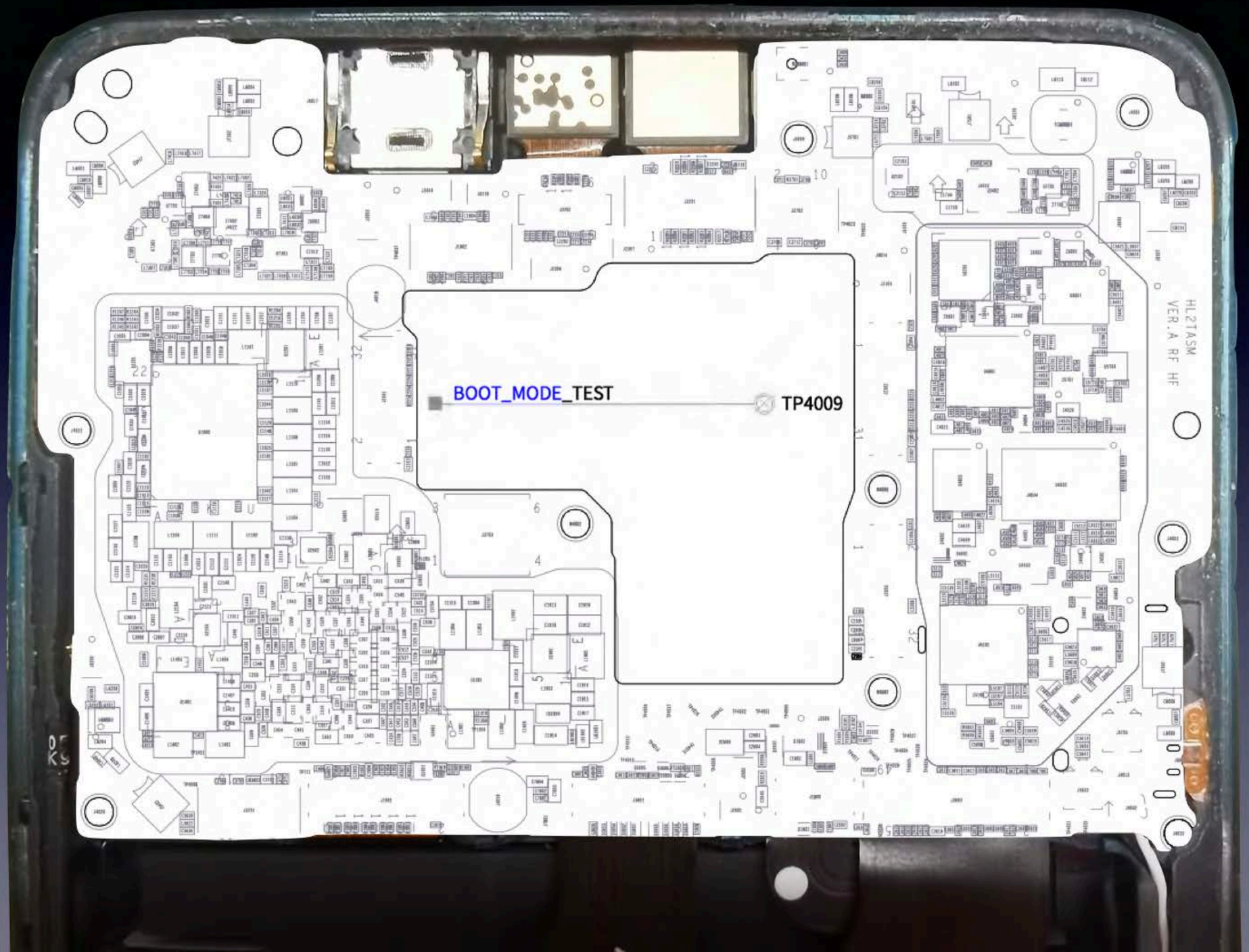
```
signed int check_bootmode()
{
  if ( (MEMORY[0x40285114] & 3) != 0 )
  {
    cprintf("normal\n");
    return 1;
  }
  else
  {
    cprintf("download \n");
    set_boot_state(0x1F);
    return 0;
  }
}
```

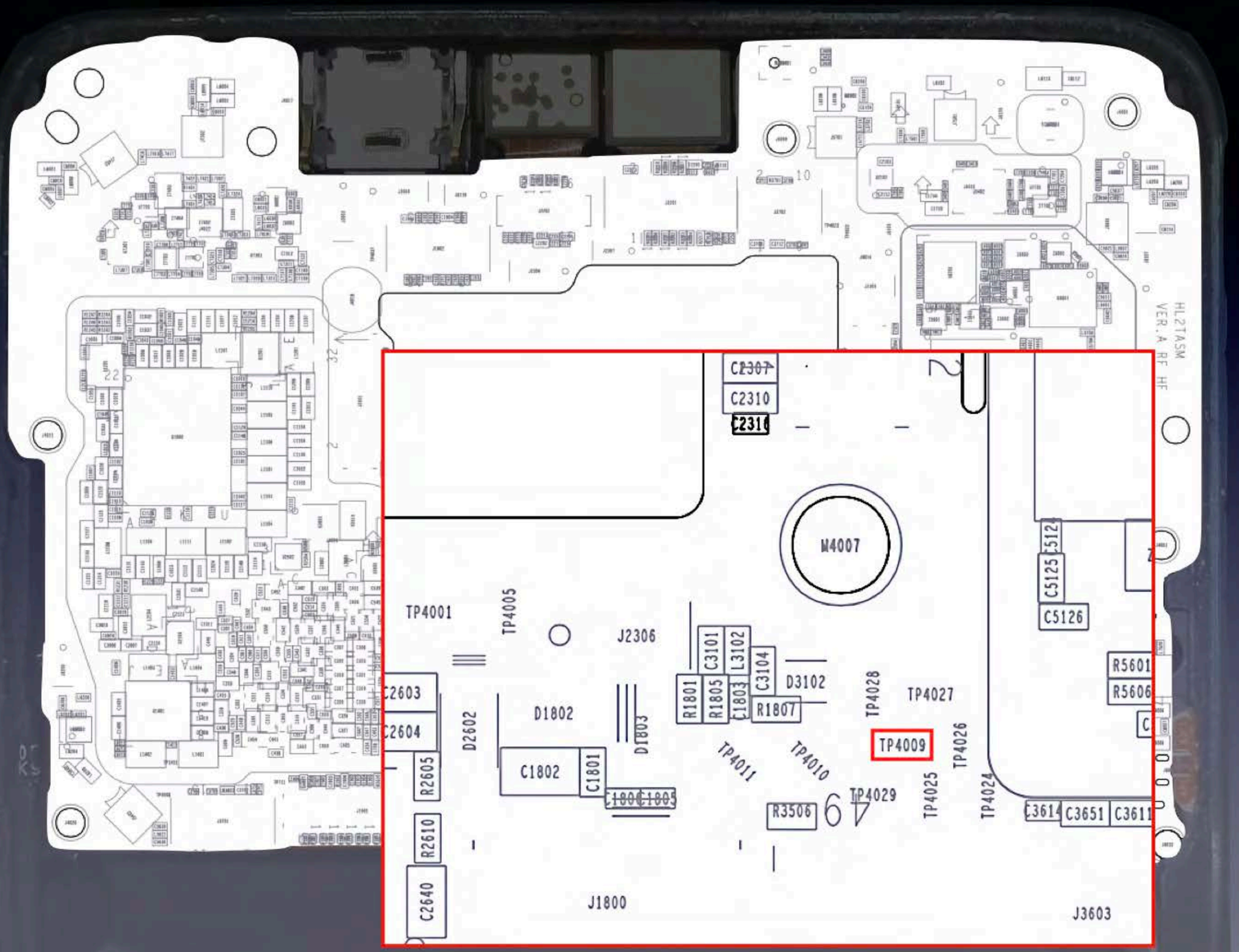
```
SOC_CRGPERIPH_PERI_STAT1_ADDR(0x40285114)
SOC_CRGPERIPH_PERI_STAT1_bootmode_START (0)
```

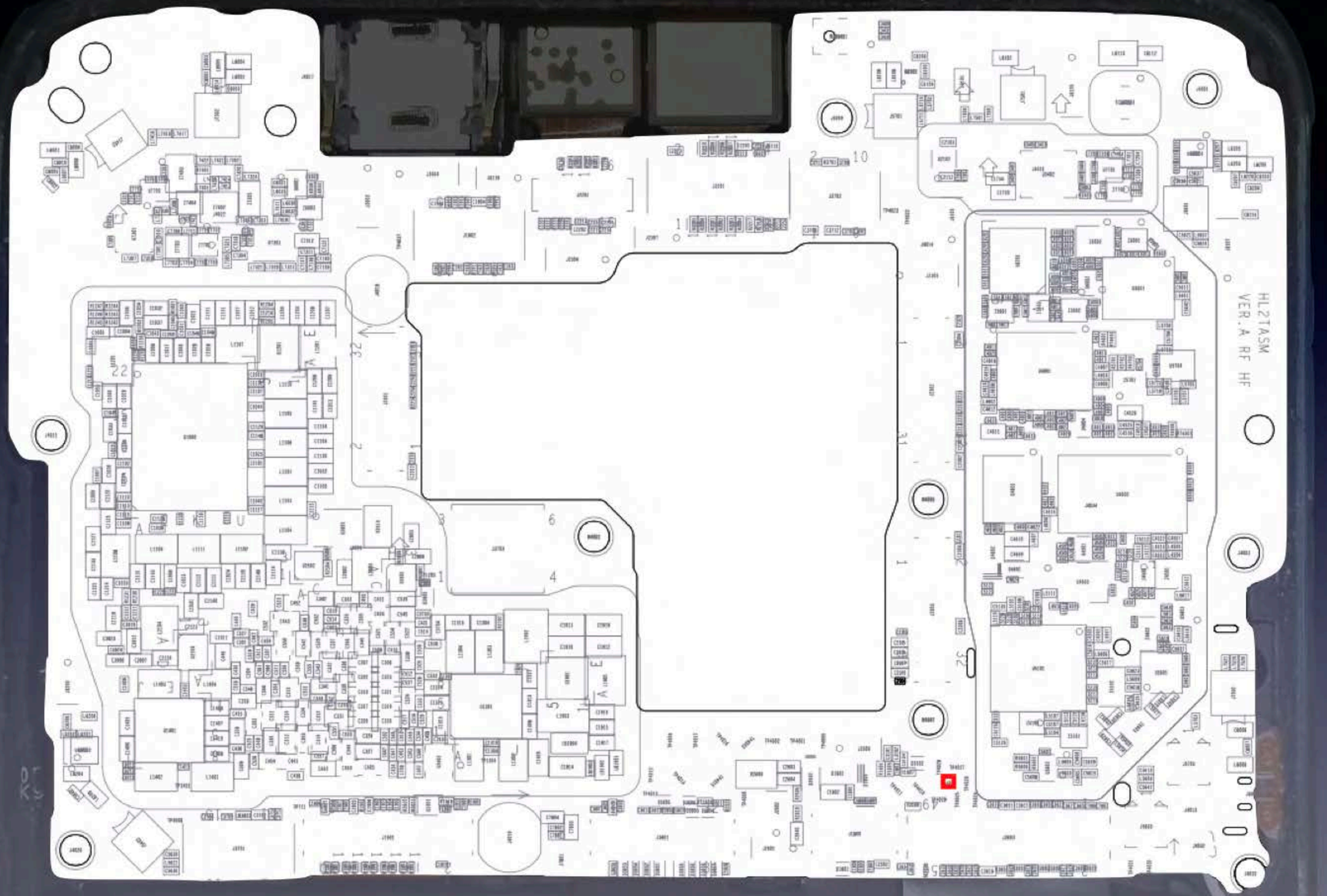

VBUCK8_1V8
10K
R2406
BOOT_LINK
R2404
100
BOOT_MODE
100nF
C2401

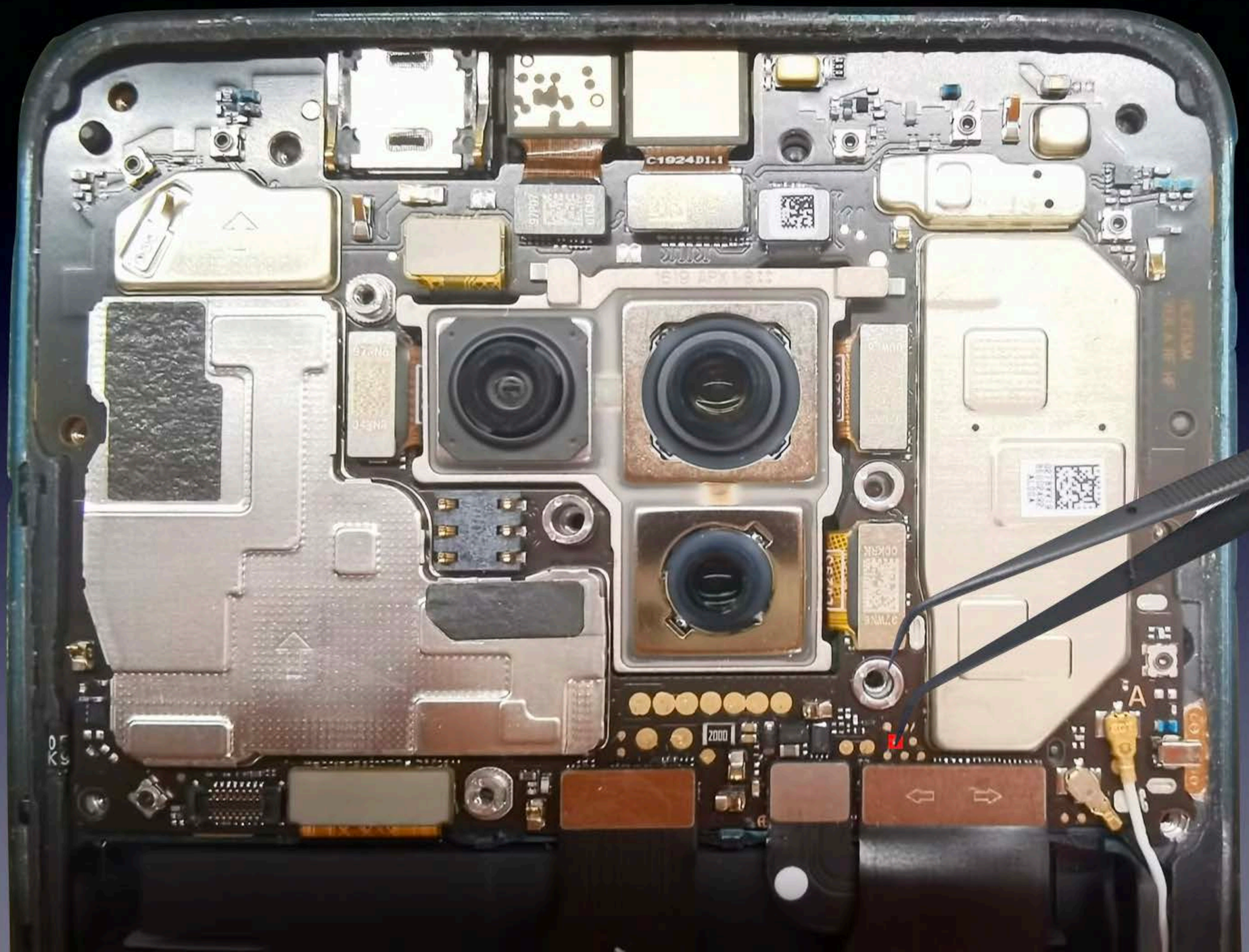

MB
HLITAURUSFLA VER.C
BF 9F
先撕开①②
再单独向上拉出③
①
②
③
A 1928 . 40
HLITAURUSFLA VER.B
SUB




BOOT_MODE_TEST
TP4009
HL2TASM
VER.A RF HF


C2307
C2310
C2310
M4007
TP4001
TP4005
J2306
C3101
L3102
C3104
D3102
C2603
C2604
D1802
D2602
D1803
R1801
R1805
C1803
R1807
TP4028
TP4027
TP4009
TP4026
TP4011
TP4010
TP4029
TP4025
TP4024
C1802
C1801
C1804
C1805
R3506
R2605
R2610
C2640
J1800
J3603
C5125
C5126
R5601
R5606
C3614
C3651
C3611
HL2TASM
VER.A RF HF


HL2TASM
VER.A RF HF


C1924D1.1
2000
A

Download mode

```
res = usb_download(0x22000);
if ( res )
{
  v18 = cprintf("usb download err:%d\n", res);
  usb3_reset(v18, v19);
  goto LABEL_10;
}
12:
ver_mode = get_ver_mode(0x22000, (unsigned int *)&vermode);
v11 = "get ver mode err:0x%x\n";
if ( ver_mode )
  goto LABEL_30;
if ( !is_download )
{
  ver_mode = GM_SecureInit(vermode);
  if ( ver_mode )
    break;
}
_vermode = vermode;
v22 = 0x50C7A5C3;
if ( vermode )
  v13 = 1;
else
  v13 = 2;
count = 0;
while ( 2 )
{
  v15 = Image_verify(0x22000, _vermode, count);
```

```
is_elf = get_file_format(0x22000) << 28;
if ( is_elf )
  entry_point = (int (__fastcall *)(int))(dword_23018 + 1);
else
  entry_point = (int (__fastcall *)(int))(&dword_23000 + 1);
v7 = cprintf("exc_xloader!\n", is_elf);
return entry_point(v7);
```

Download Procedure

- usb_download
  - fetch usb_xloader.img from PC to 0x22000
- verify usb_xloader.img
- boot into xloader
- Xloader has a similar procedure for bl2.img and fastboot.img …

USB Communication

- UART Interface
  - packet structure
    - | op | seq | 255 - seq | payload … | crc16(payload) |
  - ops
    - 0xCD, 0xFE, 0xDA, 0xED

USB Communication

op	usage	extra bytes
CD	**Inquire**, check current states	
FE	**File**, choose where to download	Type | Capacity | Address
DA	**Download**, send image data	Seq | Data
ED	**End**, usb_download cycle will then return	

Command handler *oxFE* configures the downloading parameters

```
if ( cmd == 0xFE )
{
  if ( seq || actual != 14 || (v22 = a1->ss_bulk_buf[3], (unsigned __int8)(v22 - 1) > 1u)
  {
    seq = 1;
  }
  else
  {
    v23 = _byteswap_ulong(*(_DWORD *)&a1->ss_bulk_buf[8]);
    v24 = _byteswap_ulong(*(_DWORD *)&a1->ss_bulk_buf[4]);
    a1->file_type = v22;
    a1->file_capacity = v24;
    a1->file_address = v23;
    a1->file_complete = v23;
    if ( v23 == 0x22000 )
    {
      if ( (v24 & 0x3FF) != 0 )
        v25 = 2;
      else
        v25 = 1;
      a1->file_received = 0;
      a1->file_curr_frame = 0;
      a1->file_total_frame = v25 + (v24 >> 10);
      a1->file_next_frame = 1;
    }
    else
    {
      cprintf("[UE]file addr err:%x\n", v23);
      seq = 7;
    }
  }
  goto LABEL_26;
```

The *file_address* must be **0x22000**

Command handler *oxDA* saves the bulk buffer into *file_address*

```
if ( cmd == 0xDA )
{
  if ( seq != (unsigned __int8)next_frame )
  {
    if ( seq != a1->file_curr_frame )
    {
LABEL_25:
      seq = 1;
      a1->file_received -= actual;
      goto LABEL_26;
    }
    cprintf("[UE]transfer again [%d, %d]\n", seq, seq);
    v17 = a1->file_curr_frame - 1;
    next_frame = a1->file_next_frame - 1;
    a1->file_received = a1->file_received + 5 - actual;
    a1->file_curr_frame = v17;
    a1->file_next_frame = next_frame;
  }
  if ( next_frame == a1->file_total_frame - 1 )
  {
    v18 = a1->file_capacity - (a1->file_curr_frame << 10);
    v19 = v18 + 5;
  }
  else
  {
    v19 = 1029;
    v18 = 1024;
  }
  if ( actual != v19 )
    goto LABEL_25;
  memmove((char *)(a1->file_address + (a1->file_curr_frame << 10)), (char *)&a1->ss_bulk_buf[3], v18);
  v20 = a1->file_curr_frame + 1;
  v21 = a1->file_next_frame + 1;
  a1->file_received -= 5;
  a1->file_curr_frame = v20;
  a1->file_next_frame = v21;
```

Expected

- FE 00FF 01 0002C000 00022000 0EF5
- DA 01FE 1A2B3C4D5E6F …
- DA 02FD 1A2B3C4D5E6F …
- …
- ED F00F 004E

Vulnerabilities

Bug #1

- No size check for the downloading file
  - Sending a huge file can corrupt anything beyond the buffer 0x22000
  - Current stack is sitting next to the buffer
- Exploit
  - send shellcode + 0x22000*N will jump to shellcode directly

Bug #2

- File address is set before validation
  - Some parameters are only initialized after sending the correct `file_addr`
  - `file_addr` is configured before the validation
  - Arbitrary`file_addr` can be used as the destination of memmove

Arbitrary Write

- Set file download address to 0x22000
- Set file download address to ANY ADRESS
- Download to ANY ADRESS

Arbitrary Read

- Arbitrary Read
  - Overwrite descriptor of USB device, which is at constant address
  - Fetch the descriptor by USB control message
    - Read 10 bytes each time from Device Qualifier Descriptor
  - Restore the descriptor

Code Execution

- fixed stack trace

```
#0  0x00003ada in 0x00003a54
#1  0x000038c6 in 0x000037b8
#2  0x00003908 in 0x000038d0
#3  0x000006ca in 0x000006ac (usb_download)
#4  0x00000958 in 0x0000085c
#5  0x00000072 in 0x00000048 (reset_hanler)
```

- overwrite the return address of usb_handler on stack
- shellcode
  - save and restore registers, jump back to saved LR

Fastboot Unlock

Bootrom Stage

- download usb_xloader.patch
  - bypass address and length check
  - bypass image verification
  - Jump to usb_xloader.patch

Bypass Address&Length Check

```
case 0xFE:
  if ( !v2 && a2 == 14 )
  {
    if ( (unsigned int)v4[3] - 1 <= 1 )
    {
      v18 = bswap32(*((_DWORD *)v4 + 2));
      MEMORY[0x60021AE0] = bswap32(*((_DWORD *)v4 + 1));
      MEMORY[0x60021ADC] = v18;
      if ( check_addr(v18, MEMORY[0x60021AE0]) == 0x5A5A5A5A )
      {
        if ( (MEMORY[0x60021AE0] & 0x3FF) != 0 )
          v20 = 2;
        else
          v20 = 1;
```

Bypass Image Verfication

```
cprintf("xloader main download mode to get fastboot!\n");
sub_3AD28();
v5 = v13;
image_info = &g_image_list;
while ( 1 )
{
  switch_ddr_window(image_info[2] >> 28);
  download_addr = usb_download(0);
  fastboot_or_bl2_addr = image_info[1];
  if ( download_addr != fastboot_or_bl2_addr )
  {
    cprintf("[%x]tgErr0x%x\n", v3, download_addr);
    switch_ddr_window(3u);
    sub_3B11C(162, 1);
  }
  // image verification continues ...
  v14[2] = 11;
  v9 = image_info[6];
```

Bypass Image Verfication

```
cprintf("xloader main download mode to get fastboot!\n");
sub_3AD28();
v5 = v12;
image_info = &g_image_list;
while ( 1 )
{
  switch_ddr_window(image_info[2] >> 28);
  result = usb_download(0);
  fastboot_addr = image_info[1];
  if ( result == fastboot_addr )
    return result;
  cprintf("[%x]tgErr0x%x\n", v3, result);
  switch_ddr_window(3u);
  sub_3B11C(162, 1);
  // image verification continues ...
  v13[2] = 11;
```

Xloader Stage

- download usb_uce.img to 0x60000000
- download bl2.img to 0x1E400000 without 0xED
- download fastboot.patch to 0x1A400000
  - set fb_lock_status to 1

set fb_lock_status to 1

```
v1 = get_rd_lock();
if ( !v1 )
  v1 = get_securedebug_efuse_value() == 0;
v0[65] = v1;
v0[66] = 0;
v2 = huawei_chkroot();
v0[68] = 0;
v0[67] = v2;
v0[69] = huawei_frp();
current_cpu = (unsigned int)get_current_cpu();
boot_time = get_boot_time();
log_buffer(
  "[cpu%d][%d ms]huawei_fblock: hwdog_certify->lock_state.fb_lock_stat = %d\n"
  current_cpu,
  boot_time,
  (unsigned int)v0[65]);
```

FASTBOOT&RESCUE MODE

Please Connect Usb Cable to
Your Computer and Open Hisuite

If You Want to Force Restart Your Device
Disconnect The Usb Cable and Press
and Hold The Volume Down and Power
Buttons for More Than 10 Seconds

Device reboot reason:
COLDBOOT 16
no
NA
check_boot_mode

KEEP CALM

AND

PHONE Unlocked

EL3 Root Shell

Xloader Stage

- download bl2.patch
  - Drop a SMC wrapper for bl2 RW
- download fastboot.patch
  - disable load_kernel & load_bl31
  - enable dts_init
  - Drop a wrapper for fastboot RWE

Disable Loading Kernel & BL31

```
v6 = kernel_ptn_name();
v7 = load_and_decompress_image(
       (__int64)v6,
       *(unsigned int *)(bootimg_addr + 0xC),
       *(_DWORD *)(bootimg_addr + 8),
       &v18);
current_cpu = (unsigned int)get_current_cpu();
v8 = get_boot_time();
log_buffer("[cpu%d][%d ms]load_kernel: Decompressed kernel size is 0x%x\n", current_cpu, v8, v18);
```

```
v1();
v2 = sub_3A52718C();
v3 = 0;
if ( v2 )
{
  cprintf("load_bl31: Fail to load bl31 iamge!\n", 0i64);
  current_cpu = (unsigned int)get_current_cpu();
  boot_time = get_boot_time();
```

Enable dts_init

```
get_mode_state = get_ops((__int64)"getmode");
if ( get_mode_state && !(*(unsigned int (**)(void))get_mode_state)() )
{
  cprintf("dtimage: %s usb download mod,not load dt.\n", "load_dts_by_boot_mode");
  current_cpu = (unsigned int)get_current_cpu();
  boot_time = get_boot_time();
  log_buffer(
    "[cpu%d][%d ms]dtimage: %s usb download mod,not load dt.\n",
```

Fastboot Stage

- load kernel.patch
  - Drop a wrapper for SU ability
  - disable mod_verify_sig
- load trustfirmware.patch
  - Drop a SMC wrapper for EL3 RWE
- fastboot oem boot

Android Stage

- /data/local/tmp/su to EL0 root
- insmod exploit.ko to EL1 root
- smc wrapper to EL3 root

Enable JTAG

Secdbg Switch (Xloader)

```
  ret = SEB_ReadOTPWord(0x7F, &dbg_ctrl);
  if ( ret )
  {
    log("dbg ctl read err.\n");
    v1 = "parse dbg ctrl err(0x%x).\n";
LABEL_29:
    log(v1, ret);
    goto LABEL_37;
  }
  if ( ((dbg_ctrl >> 9) & 3) != 2 )
  {
    log("dbg not dcu.\n");
LABEL_37:
    MEMORY[0xB8100F64] |= 0x88000000;
    return ret;
  }
```

```
SECENG_S_HOST_DCU_EN (0xB8100F64)
SECENG_S_HOST_DCU_EN_CPU_dbgen_en   (00)
SECENG_S_HOST_DCU_EN_CPU_nigen_en   (01)
SECENG_S_HOST_DCU_EN_CPU_spiden_en  (02)
SECENG_S_HOST_DCU_EN_CPU_spniden_en (03)
...
SECENG_S_HOST_DCU_EN_CPU_lock_bit   (31)
```

DBGAUTHSTATUS_EL1

- NSID, bits [1:0], Non-secure invasive debug
- NSNID, bits [3:2], Non-secure non-invasive debug
- SID, bits [5:4], Secure invasive debug
- SNID, bits [7:6], Secure non-invasive debug
- 0b10: disabled, 0b11: enabled

Secdbg Switch Test

```
// exploit.ko
void dbg_status_test(void)
{
  uint64_t status;

  //enable SECENG
  smc_call(0xc500aa01, 0xaa55a5a5, 0, 0x55bbccf0);
  asm volatile("mrs %0, DBGAUTHSTATUS_EL1" : "=r" (status));
  printk(KERN_ALERT "DBGAUTHSTATUS_EL1(before): 0x%X\n", status);

  // SECENG_S_HOST_DCU_EN maps differently
  // between Cortex-A and Cortex-M
  el3_w4(0xF8100F64, 0xFF);
  asm volatile("mrs %0, DBGAUTHSTATUS_EL1" : "=r" (status));
  printk(KERN_ALERT "DBGAUTHSTATUS_EL1(after) : 0x%X\n", status);
  smc_call(0xc500aa01, 0x55aa5a5a, 0, 0x55bbccf0);
}
```

EL3 DEMO

JTAG Multiplexing (Xloader)

```
if ( v2[15] == '1' )
{
  mux = 0x80000001;
}
else if ( v2[31] == '1' )
{
  mux = 0x80000002;
}
else
{
  if ( v2[39] != '1' )
  {
    MEMORY[0x4021B854] = 0;
    MEMORY[0x4021B858] = 0;
    return sub_3A80C(0);
  }
  mux = 0x80000032;
}
MEMORY[0x4021B854] = MEMORY[0x4021B854] & 0x1FFFF00 | mux;
MEMORY[0x4021B858] &= 0xFFFFFF00;
sub_3A7A4(MEMORY[0x4021B854], 1);
sub_3A7E4(1);
return log("jtagmux_en\n");
```

```
int __fastcall sub_3A7A4(int a1, int a2)
{
  int result; // r0
  int v3; // r3

  MEMORY[0x4021B858] &= 0xFFFFFF00;
  result = a1 | MEMORY[0x4021B854] & 0x1FFFF00;
  MEMORY[0x4021B854] = result;
  v3 = MEMORY[0x4021B31C];
  if ( a2 == 1 )
  {
    v3 = MEMORY[0x4021B31C] | 0x400040;
  }
  else if ( !a2 )
  {
    v3 = MEMORY[0x4021B31C] | 0x300030;
  }
  MEMORY[0x4021B31C] = v3;
  return result;
}
```

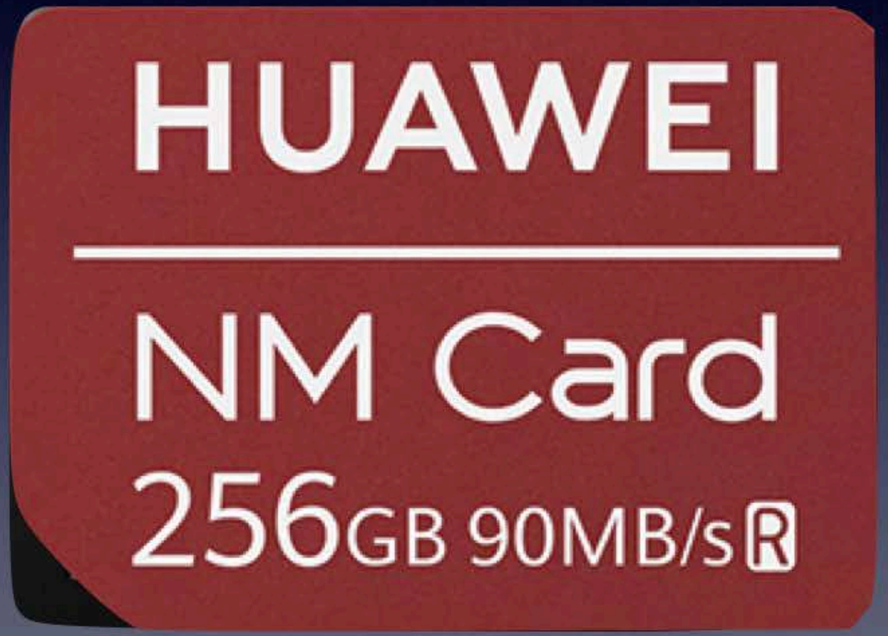
JTAG Multiplexing (Device Tree)

```
jtagtosd {
    compatible = "hisilicon,jtagtosd";
    sdio-ldo = <0xfe>;
    reg = <0x0 0x4021b854 0x0 0x1000>;
    value = <0x80000001>;
    sim2jtag_value = <0x40000001>;
    mask = <0xfe0000ff>;
    sdcard_io_sel_reg = <0x0 0x4021b31c 0x0 0x1000>;
    sdcard_io_sel_value = <0x400040>;
    simcard_io_sel_value = <0x300030>;
    sdcard_io_sel_mask = <0x0>;
    testmode_pd_gpio = <0x1>;
    phandle = <0x556>;
};
0x4021b854: SCTRL_SCJTAGSD_SW_SEL_ADDR
0x80000002: DJTAG
0x80000032: Serial Port
```

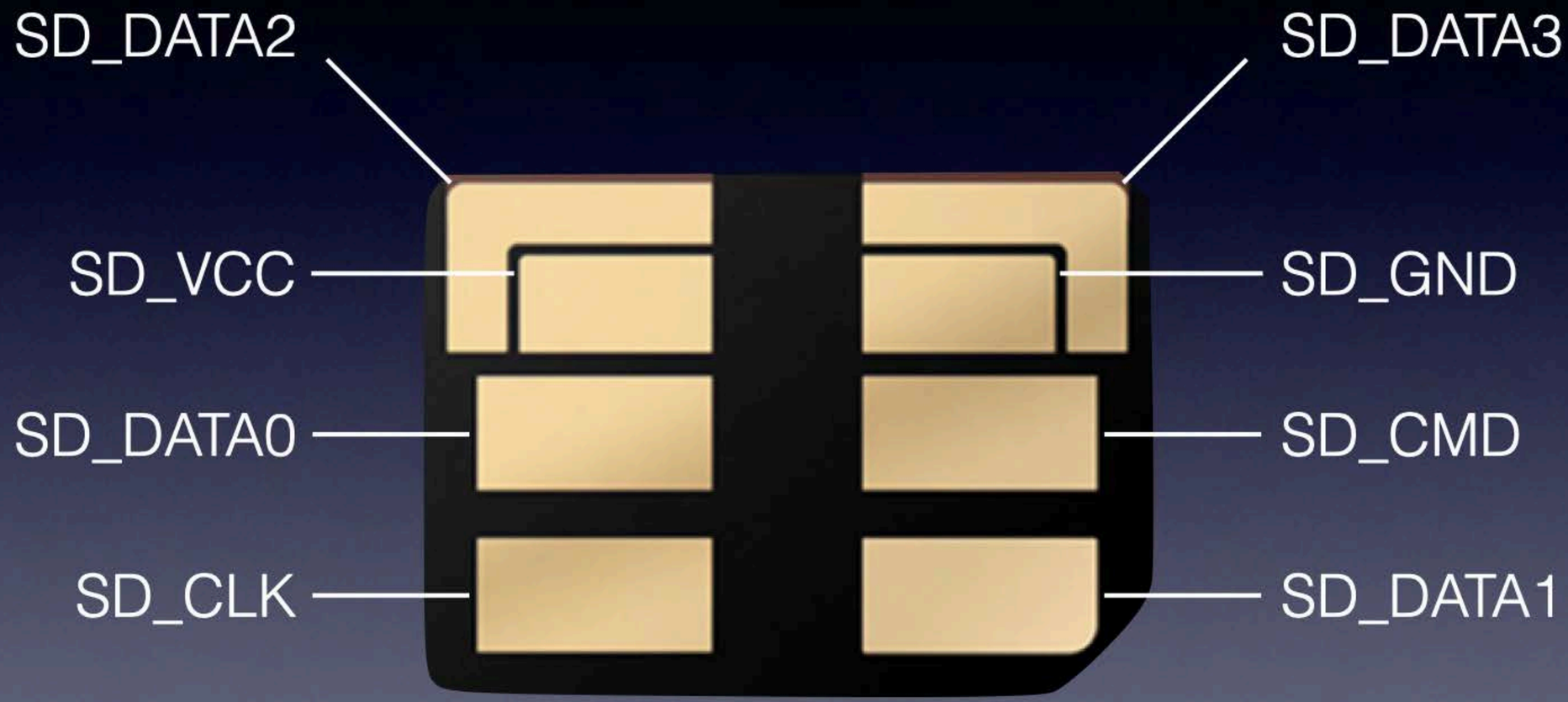
Enable JTAG (Software Approach)

- enable debug authentication signals
  - 0xFFFFFFFF => 0xB81F0064
- select JTAG function
  - 0x80000001 => 0x4021B854
- Multiplexing SDCARD
  - 0x400040 => 0x4021B31C

NM Card



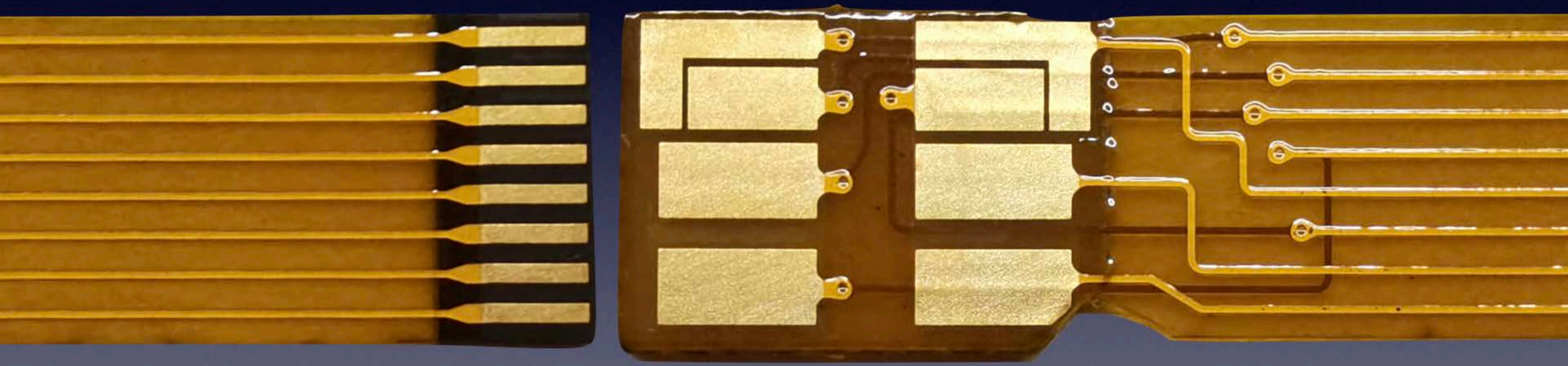
NM Card



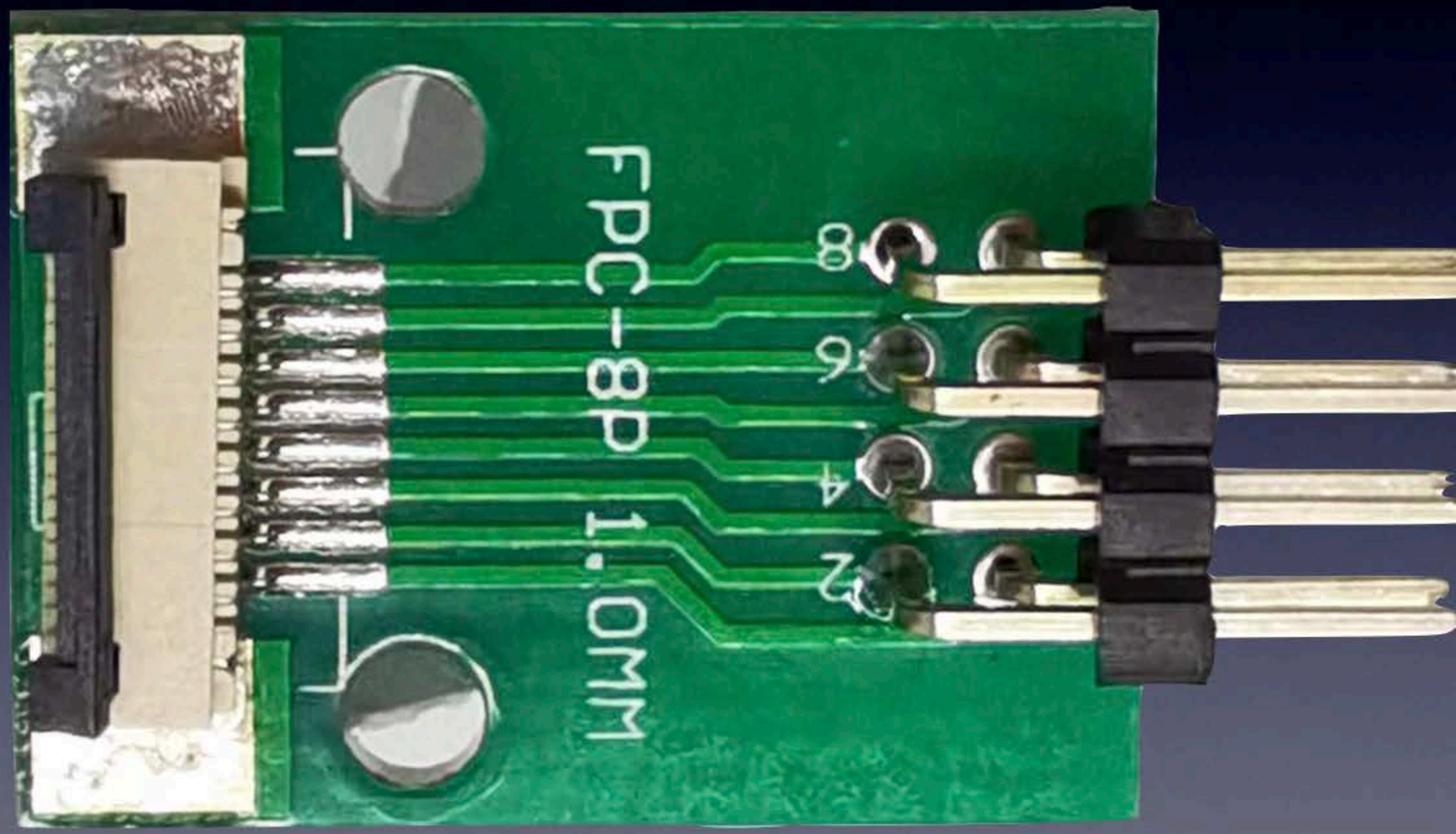
NM to JTAG

NM	JTAG
SD_DATA0	JTAG_TCK
SD_DATA1	JTAG_TDI
SD_DATA2	JTAG_TDO
SD_DATA3	JTAG_TRST
SD_CMD	JTAG_TMS
SD_GND	JTAG_GND

Connect JLink to NM



Connect JLink to NM



Enable JTAG

- Connect to Mate30 via usb (with bootmode to GND)
  - Boot into download mode
- Connect NM pins to Jlink Device
  - Setup registers via bootrom exploit
  - Start JLinkGDBServer

JTAG DEMO

The Fix

checkm8

- A5 - A11: still vulnerable
- A12 - A13: only memory leak issue was fixed (no public exploits)

```
__int64 __fastcall usb_core_handle_usb_control_receive(__int64 ep0_rx_buffer,
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  *a4 = 0;
  if ( (a2 & 1) != 0 )                          // setup_phase
  {
    setup_request = *(usb_device_request *)ep0_rx_buffer;
    v6 = usb_controller_abort_endpoint(0x80i64);
    v7 = setup_request.bmRequestType & 0x60;
    if ( v7 == 0x40 )
    {
      ep0_rx_buffer = sub_ADE0();
      if ( (_DWORD)ep0_rx_buffer == -1 )
      {
LABEL 79:
```

checkm8

- A5 - A11: still vulnerable
- A12 - A13: only memory leak issue was fixed (no public exploits)
- A14 - : fully fixed

```
__int64 __fastcall usb_core_handle_usb_control_receive(__int64 ep0_rx_buffer,
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  if ( !a4 )
    goto LABEL_121;
  v5 = ep0_rx_buffer;
  *a4 = 0;
  if...                                         // data phase
  ep0_data_phase_rcvd = 0;                      // setup phase
  ep0_data_phase_length = 0xFFFFFFFE00000000ui64;
  ep0_data_phase_buffer = 0i64;
  setup_request = 0i64;
  setup_request = *(_QWORD *)ep0_rx_buffer;
  unk_1FC025079 = 1;
  ep0_rx_buffer = usb_controller_abort_endpoint(0x80i64);
  v6 = setup_request & 0x60;
  if ( v6 == 0x40 )
  {
```

checkm30

- Mate 40 and later are immune to checkm30
- Older devices?
  - Surprisingly, they were fixed!!!
  - OTAs that contain **bootrom patches** were pushing to them
    - I have no idea how this magic trick was done :)

Thank you