RANDORISEC

Using n-day to break kirin's secureboot for fun and profit

Oscar AZZOPARDI

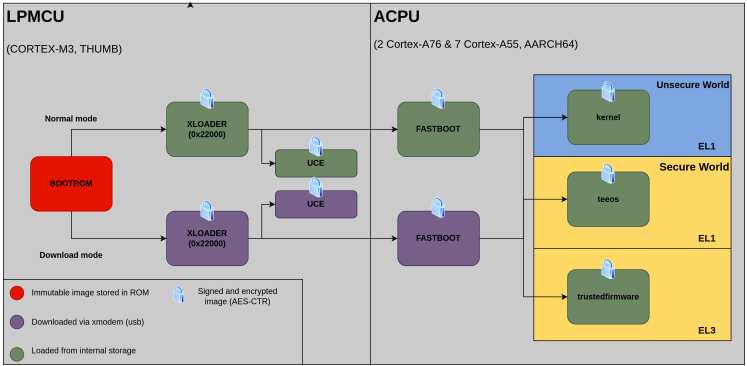
September 6, 2024

Target device

- Device: Huawei P40 Lite.
- SoC: kirin 810

RANDORISEC

Kirin's bootchain



Vendor's protocol: xmodem

HEADFRAME

CMD (0xFE) 1 bytes	Sequence 1 byte	Sequence~ 1 byte	filetype 1 byte	length 4 bytes (uint)	address 4 bytes (uint)	CRC 2 bytes (ushort)

DATAFRAME

CMD (0xDA) 1 bytes	Sequence 1 byte	Sequence~ 1 byte	Data up to 1024 bytes	CRC 2 bytes (ushort)

TAILFRAME

CMD (0xED) 1 bytes	Sequence 1 byte	Sequence~ 1 byte	CRC 2 bytes (ushort)

INQUIRY FRAME

CMD (0xCD) 1 bytes	Sequence 1 byte	Sequence~ 1 byte	CRC 2 bytes (ushort)

Response

ERR	Invalid frame field value
NAK	CRC check failed
ACK	Everything is good

RANDORISEC

Summary

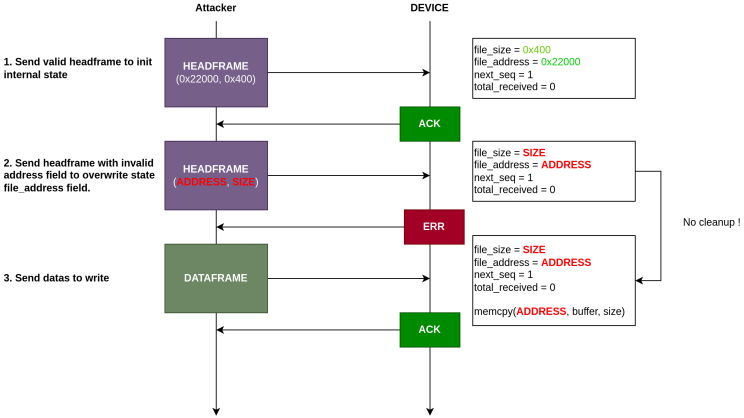
- ☐ Vulnerability root cause analysis.
- ☐ Turning arbitrary write into arbitrary execution
- ☐ Decrypting bootchain's images.
  - ☐ Setting up decryption oracle.
  - ☐ Patching ROM instructions with FPB.
- ☐ Patching images to bypass secureboot.

CVE-2021-22433: Head Re-Send State Machine Confusion

```
// Fill state struct
file_addr = _byteswap_ulong(*(_DWORD *)&usb->ss_bulk_buf[8]);
file_size = _byteswap_ulong(*(_DWORD *)&usb->ss_bulk_buf[4]);
usb->file_type = file_type;
usb->file_size = file_size;
usb->file_address = (_BYTE *)file_addr;
usb->file_complete = file_addr;
// Check download address
if ( file_addr == 0x22000 )
{
  if ( (file_size & 0x3FF) != 0 )
    size = 2;
  else
    size = 1;
  usb->total_received = 0;
  usb->latest_seen_seq = 0;
  usb->total_frame_count = size + (file_size >> 10);
  usb->next_seq = 1;
}
else
{
  printf("[USBE]file load addr error:%x\n", file_addr);
  curr_frame_seq = 7;
}
```

Credits to TASZK Security Labs for discovery and exploitation

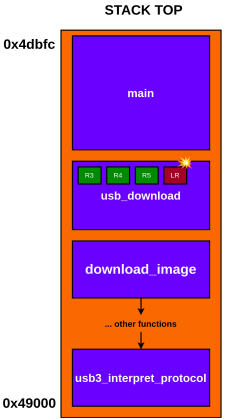
CVE-2021-22433: Triggering vulnerability



Summary

- ☒ Vulnerability root cause analysis.
- ☐ Turning arbitrary write into arbitrary execution
- ☐ Decrypting bootchain's images.
  - ☐ Setting up decryption oracle.
  - ☐ Patching ROM instructions with FPB.
- ☐ Patching images to bypass secureboot.

Gaining arbitrary code execution

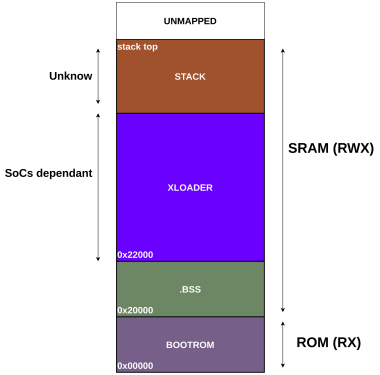


- Higly determinist !
  - Offset from stack top is easily predictable.
  - Call happen early in bootrom.
  - We can smash first 52 bytes of the stack to overwrite saved PC.
- Same technique can be used on other SoCs.

RANDORISEC

Gaining arbitrary code execution: where is stack top ?

Hopefully we can reduce range of values to bruteforce



Model	SoC	Xloader size	Stack top
POT	kirin 710	0x2d000	0x49bfc
JNY	kirin 810	0x28000	?
EML	kirin 970	0x22000	0x4dbfc
YAL	kirin 980	0x27000	0x4dbfc

RANDORISEC

Gaining arbitrary code execution: finding usb_dowload with shellcode

```
4F F4 08 30                 MOV.W   R0, #0x22000     ; image_addr
FF F7 35 FE                 BL      usb_download
usb_download                                        ; CODE XREF: start+142↓p
38 B5                       PUSH    {R3-R5,LR}      ; Save registers on stack
05 46                       MOV     R5, R0
04 F0 C2 FD                 BL      flip_bootrom_flag
04 F0 94 FC                 BL      sub_4D90
04 F0 BE FD                 BL      flip_bootrom_flag
00 20                       MOVS    R0, #0          ; result
04 F0 AD FD                 BL      sub_4FCC
04 F0 B9 FD                 BL      flip_bootrom_flag
03 F0 7F F9                 BL      download_image
```

- **usb_download** open/close serial connection.
- Search a pattern in bootrom to find the function.

Gaining arbitrary code execution

Memory at 0x4ebfc is not maped (cause device to crash), which mean that stack top is at 0x4dbfc

```
[+] Trying to write at address 0x4cbfc
[+] Trying to write at address 0x4dbfc
[+] Trying to write at address 0x4ebfc
[!] Failed to send dataframe 1
[!] Failed to write at address 0x4ebfc
Press a key to continue ...
```

Our shellcode found **usb_download** at 0x045C. By using inquiry frame, we sucessfully dumped bootrom in 4h.

RANDORISEC

Summary

- ☒ Vulnerability root cause analysis.
- ☒ Turning arbitrary write into arbitrary execution
- ☐ Decrypting bootchain's images.
  - ☐ Setting up decryption oracle.
  - ☐ Patching ROM instructions with FPB.
- ☐ Patching images to bypass secureboot

RANDORISEC

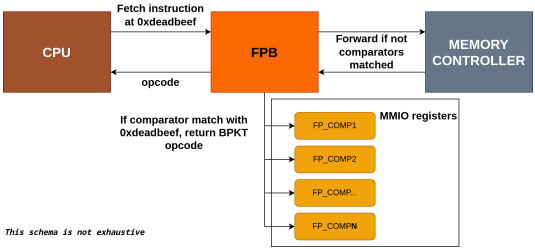
Decrypting images

We need to call these 2 functions :

- **AES_Init** located at 0x00006968. Setup cryptocell in AES-CTR mode and fill nonce + counter.
- **AES_Update** located at 0x000069E4. Decrypt buffer of arbitrary size using AES-CTR.

But we can only read 4 bytes per 4 bytes which reduced decryption …

Increasing decryption speed: Hooking with FPB



- Patch memory access in ROM and SRAM.
  - Insert breakpoints in bootrom.
- Hijack execution flow and change registers content.

Increasing decrpytion speed: Hooking with FPB

```
inquiry_val = get_inquiry_val();
xmodem_send_inquiry(usb, inquiry_val);
return 0;
```

```
int __fastcall xmodem_send_inquiry(struct usb3_pcd *usb, int inquiry)
{
  usb->ss_bulk_buf[0] = inquiry;
  usb->ss_bulk_buf[2] = BYTE2(inquiry);
  usb->ss_bulk_buf[3] = HIBYTE(inquiry);
  usb->in_ep.req.length = 4;
  usb->in_ep.req.bufdma = usb->ss_bulk_buf;
  usb->ss_bulk_buf[1] = BYTE1(inquiry);
  usb->in_ep.req.complete = (void (__fastcall *)(void *))ep_req_complete;
  return usb3_bulk_out_continue_transfer((int)usb, (int)&usb->in_ep);
}
```

- Hook **xmodem_send_inquiry** using FPB and breakpoints
- Inside debug handler we have to :
  - Change size of usb data to be sent to host
  - Change content of usb data to be sent to host

RANDORISEC

Summary

- ☒ Vulnerability root cause analysis.
- ☒ Turning arbitrary write into arbitrary execution
- ☒ Decrypting bootchain's images.
  - ☒ Setting up decryption oracle.
  - ☒ Patching ROM instructions with FPB.
- ☐ Patching images to bypass secureboot.

Secureboot bypass: Bootrom stage

```
46      usb_err = usb_download(0x22000u);
47      if ( usb_err )
48        break;
49  verify_xloader_signature:
50      if ( al || (init_err = init_cc()) == 0 )
51      {
52        counter = 0x50C7A5C3;
53        for ( i = 0; i != 2; ++i )
54        {
55          verify_err = cc_img_verification(0x22000, i);
56          if ( verify_err )
57          {
58            if ( verify_err != 0x50C7B1E4 )
59              update_inquiry_by_err(verify_err);
60            al = 1;
61            goto LABEL_18;
62          }
63          j = 3;
64          do
65          {
66            ++counter;
67            udelay_100us(1);
68            --j;
69          }
70          while ( j );
71        }
72        if ( counter != 0x50C7A5C9 )
73        {
74          update_inquiry(0x50);
75          al = 1;
76          goto LABEL_18;
77        }
78        goto jmp_xloader;
79      }
```

- Patch call to cc_image_verification.
- Bootrom will naturally go to label **jmp_xloader**.

RANDORISEC

Secureboot bypass: Xloader stage

```
printf("xloader main download mode to read uce!\n");
v4 = (v2 + 52 + 32 * phdr_idx);           // parse ELF header
expected_address = v4->p_paddr;
if ( expected_address != download_image(0) )
{
  printf("UCEStgErr0x%x\n", v4->p_paddr);
  system_reset(162, 1);
}
if ( UceDlVerify(phdr_idx, lcs) != -1 )
  goto done;
printf("UceDlVerify fail!\n");
```

load_uce function

```
err = sub_33050(&v10, FASTBOOT, 0x5FE000);
if ( err != -1 )
{
  result = err + 15;
  if ( !result )
    return result;
  if ( memcpy_s(0xBFC20000, 4096, 0x1C000000, 0x1000u) )
  {
    printf("vrl cpy E\n");
  }
  else if ( sub_23A02(0x1C000000, 0x5FE000, 469766144, 0x5FE000) )
  {
    printf("img cpy E\n");
  }
  else
  {
    v10.vrl_start = 0xBFC20000;
    v10.img_start = 0x1C000000;
    result = sub_3309C(&v10) + 1;
    if ( result )
      return result;
  }
```

load_fastboot function

RANDORISEC

Secureboot bypass: Fastboot stage

```
hwdog_certify->lock_state.fb_lock_stat = fb_lock_stat;
hwdog_certify->lock_state.user_lock_stat = 0;
```

```
fb_lock_stat = get_hwdog_certify()->lock_state.fb_lock_stat;
if ( fb_lock_stat == 1 )
{
  fb_lock_stat_str = "UNLOCKED";
}
else if ( fb_lock_stat )
{
  fb_lock_stat_str = "UNKNOWN";
}
else
{
  fb_lock_stat_str = "LOCKED";
}
```

Credits to Guanxing Wen & Slipper (checkm30)

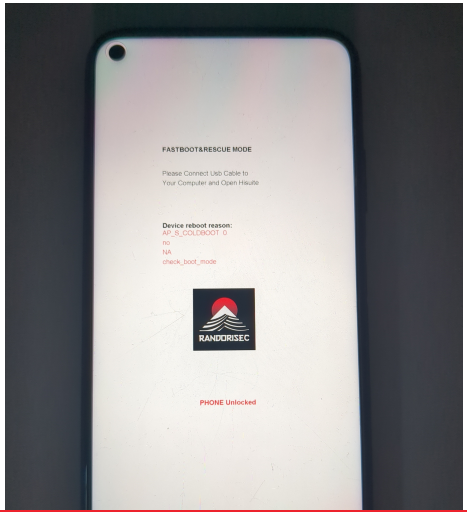
Summary

- ☒ Vulnerability root cause analysis.
- ☒ Turning arbitrary write into arbitrary execution
- ☒ Decrypting bootchain's images.
  - ☒ Setting up decryption oracle.
  - ☒ Patching ROM instructions with FPB.
- ☒ Patching images to bypass secureboot.

Now let's have fun !!!!

RANDORISEC

Questions ?



RANDORISEC