KCon洞见未来 2017

TAKBOO

深度解析BOOTLOADER攻击面

WHO AM I?

takboo

ABOUT ME

- Sec·Ret 团队成员
- Android 漏洞研究
- Linux 内核漏洞研究

联系方式:takboosf@gmail.com

目录
CONTENTS

01 bootloader 背景

PC 端 BOOTLOADER

- BIOS/UEFI/UBOOT
- 检查硬件，加载操作系统
- 多阶段启动

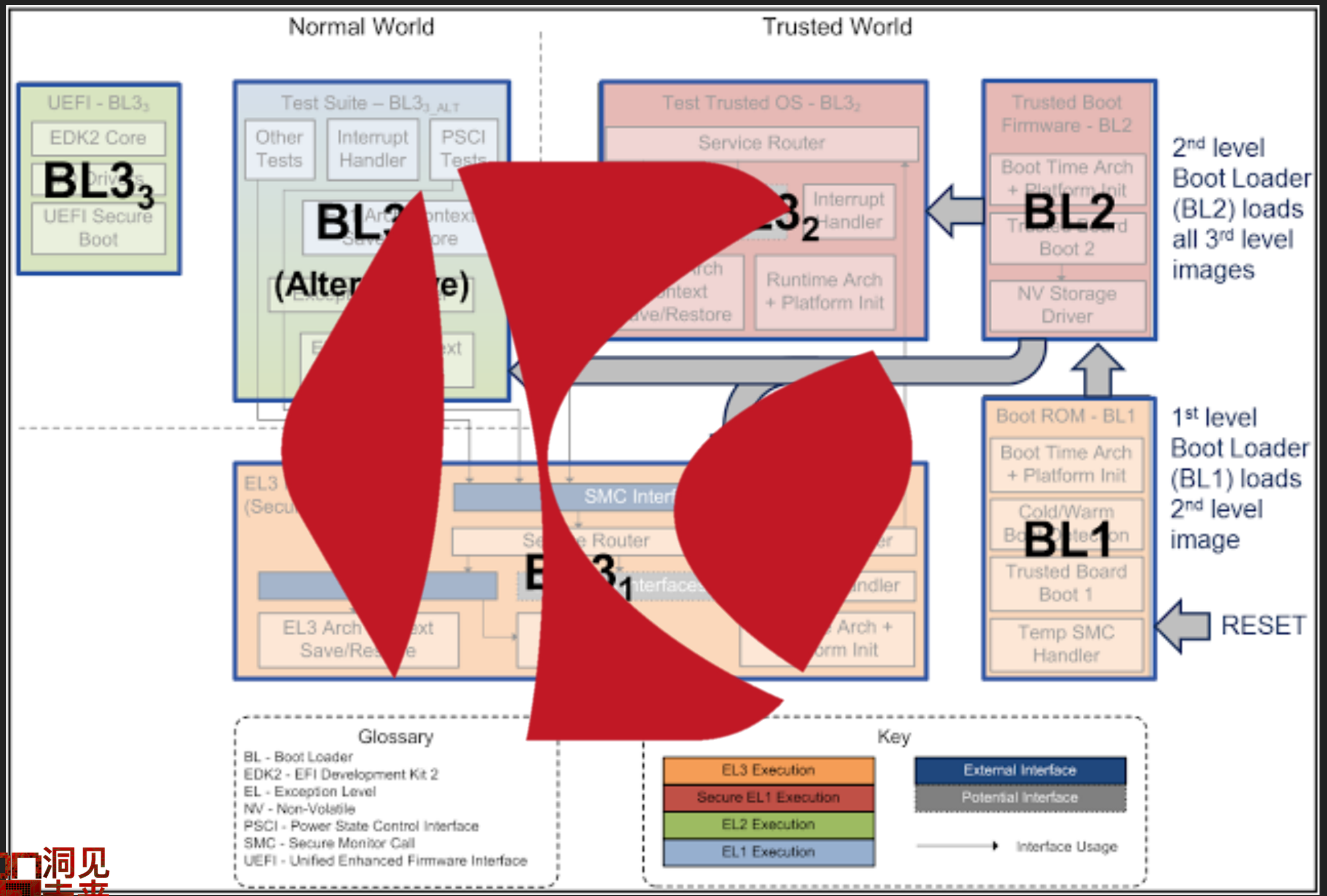
移动设备 BOOTLOADER

- 多阶段启动
- 完整性
- 来源检测
- 版本检测

ARM TRUSTED BOOT

- CoT(chain of Trust)
- Trusted Boot
- TEE(Trusted Execution Environment)
- TrustZone
- 异常级别

ARM TRUSTED BOOT



ANDROID BOOTLOADER

- aboot/hboot/sboot
- 启动安卓系统
- 厂商实现差异

ANDROID VERIFIED BOOT

- 延续 CoT
- 两套实现
- 安全状态转换
- Bootloader 解锁

02 主流厂商对比

QUALCOMM BOOTLOADER

- Aboot
- 市场占有率高
- EL1
- LK(Little Kernel)
- 符合 Trusted Boot 和 Verified Boot

MEDIATEK BOOTLOADER

- 类似于aboot
- EL1
- 不开源
- 初始化重要硬件
- 符合 Trusted Boot 和 Verified Boot

HUAWEI BOOTLOADER

- 整合后续启动阶段
- EL3
- 不开源
- 符合 Trusted Boot 和 Verified Boot

SAMSUNG BOOTLOADER

- sboot
- EL1
- 不开源
- Odin 模式

主流厂商 Bootloader 对比

Vendor	EL	Fastboot
Qualcomm	EL1	TRUE
MediaTek	EL1	TRUE
Huawei	EL3	TRUE
Samsung	EL1	TRUE

03 Qualcomm aboot

LK

- 开源(git://codeaurora.org/kernel/lk.git)
- BL33
- 支持多种启动模式
- 支持 unlocking

Lk 源码分析 http://www.freebuf.com/news/135084.html

LK

- 进行各种早期的初始化工作(cpu, emmc thread etc)。
- 判断进入 recovery 或 fastboot 的条件是否被触发。
- 从 emmc 中获取 boot.img 并加载到指定内存区域。
- 从内存加载 kernel 到 KERNEL_ADDRESS。
- 从内存加载 ramdisk 到 RAMDISK_ADDRESS。
- 加载设备树到 TAGS_ADDRESS。
- 关闭 cache, interrupts, 跳转到 kernel。

FASTBOOT

- 指令注册
- 启动监听
- 指令解析与执行

Qualcomm aboot

- 指令数组

command	handler
flash:	cmd_flash
erase:	cmd_erase
boot	cmd_boot
continue	cmd_continue
reboot	cmd_reboot
reboot-bootloader	cmd_reboot_bootloader
oem unlock	cmd_oem_unlock
oem unlock-go	cmd_oem_unlock_go
oem lock	cmd_oem_lock
oem verified	cmd_oem_verified
oem device-info	cmd_oem_devinfo
preflash	cmd_preflash
oem enable-charger-screen	cmd_oem_enable_charger_screen
oem disable-charger-screen	cmd_oem_disable_charger_screen
oem select-display-panel	cmd_oem_select_display_panel
oem run-tests	cmd_oem_runtests
getvar:	cmd_getvar
download:	cmd_download

- 指令链表

```
struct fastboot_cmd {
  struct fastboot_cmd *next;
  const char *prefix;
  unsigned prefix_len;
  void (*handle)(const char *arg, void *data, unsigned sz);
};

static struct fastboot_cmd *cmdlist;
```

- 指令注册

```
void fastboot_register(const char *prefix,
                       void (*handle)(const char *arg, void *data, unsigned sz))
{
  struct fastboot_cmd *cmd;
  cmd = malloc(sizeof(*cmd));
  if (cmd) {
    cmd->prefix = prefix;
    cmd->prefix_len = strlen(prefix);
    cmd->handle = handle;
    cmd->next = cmdlist;
    cmdlist = cmd;
  }
}
```

Qualcomm aboot

- fastboot 初始化
- fastboot 线程启动
- 等待USB

```
int fastboot_init(void *base, unsigned size)
{
  //...

  fastboot_register("getvar:", cmd_getvar);
  fastboot_register("download:", cmd_download);
  fastboot_publish("version", "0.5");

  thr = thread_create("fastboot", fastboot_handler, 0, DEFAULT_PRIORITY, 4096);
  if (!thr)
  {
    goto fail_alloc_in;
  }
  thread_resume(thr);

  usb_if.udc_start();

  return 0;
  //..
}
```

```
static int fastboot_handler(void *arg)
{
  for (;;) {
    event_wait(&usb_online);
    fastboot_command_loop();
  }
  return 0;
}
```

Qualcomm aboot

- 读取 USB

```
r = usb_if.usb_read(buffer, MAX_RSP_SIZE);
if (r < 0) break;
buffer[r] = 0;
dprintf(INFO,"fastboot: %s\n", buffer);
```

- 解析命令

```
for (cmd = cmdlist; cmd; cmd = cmd->next) {
  if (memcmp(buffer, cmd->prefix, cmd->prefix_len))
    continue;
  cmd->handle((const char*) buffer + cmd->prefix_len,
        (void*) download_base, download_size);
  if (fastboot_state == STATE_COMMAND)
    fastboot_fail("unknown reason");
  goto again;
}
```

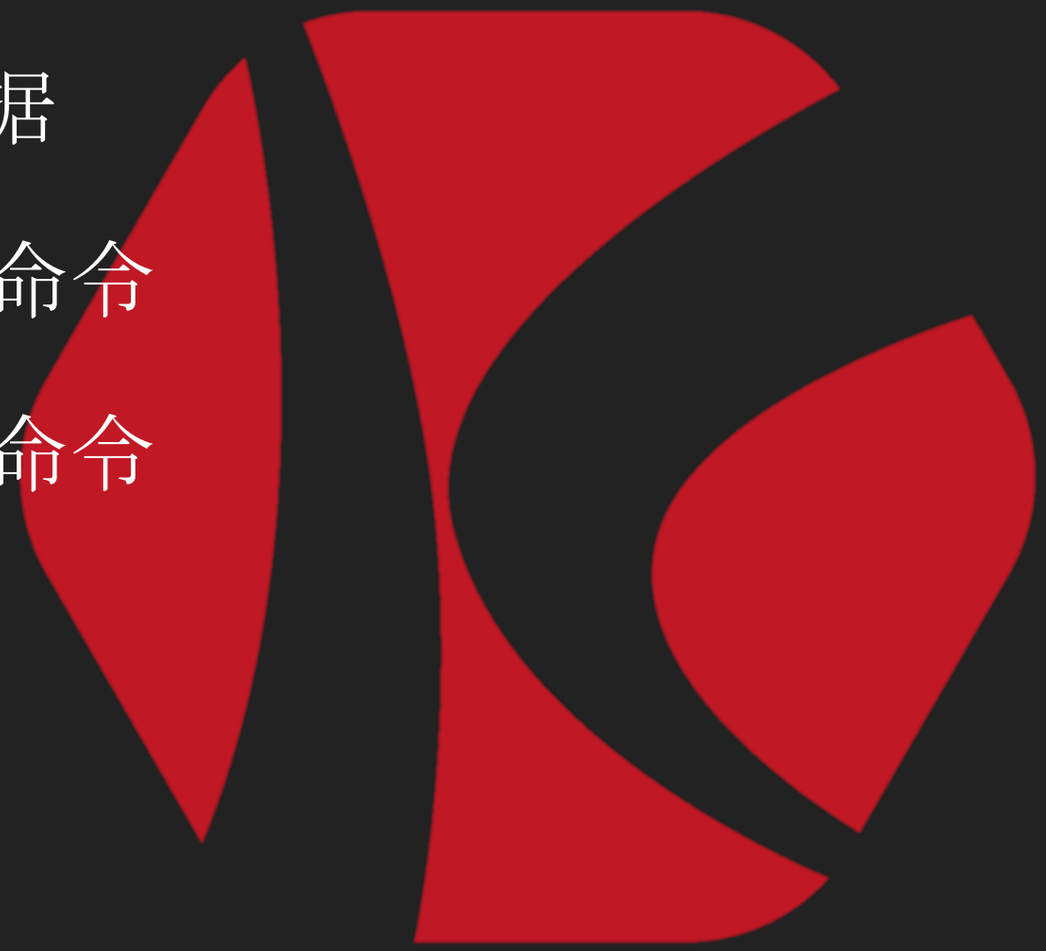
04 Bootloader 漏洞挖掘

BOOTLOADER 难点

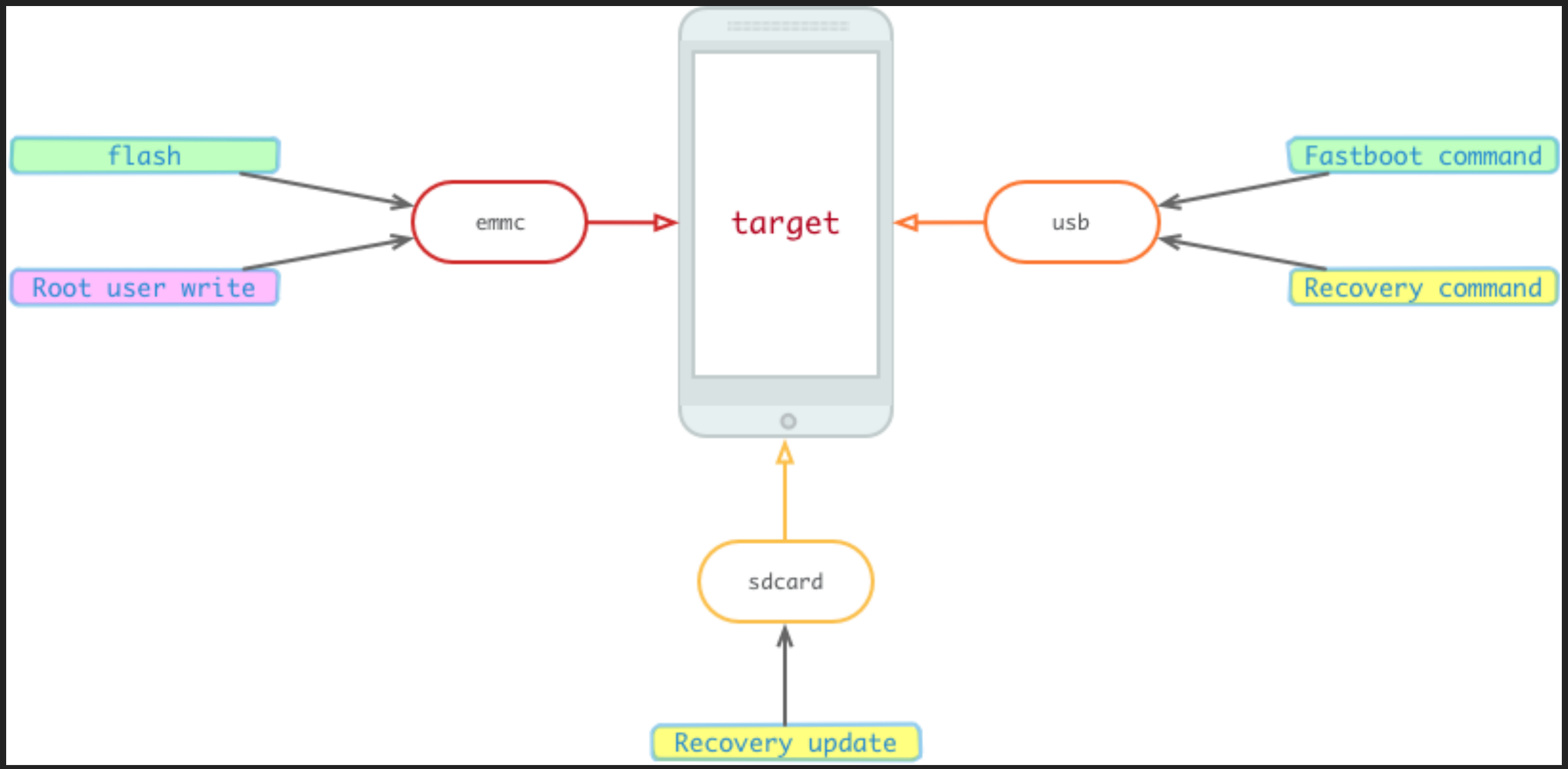
- 闭源
- 无调试符号
- 格式不统一
- 函数库/函数签名
- 硬件耦合

BOOTLOADER 攻击面

- 存储数据
- Sdcard 数据
- Recovery 命令
- Fastboot 命令



BOOTLOADER 攻击面

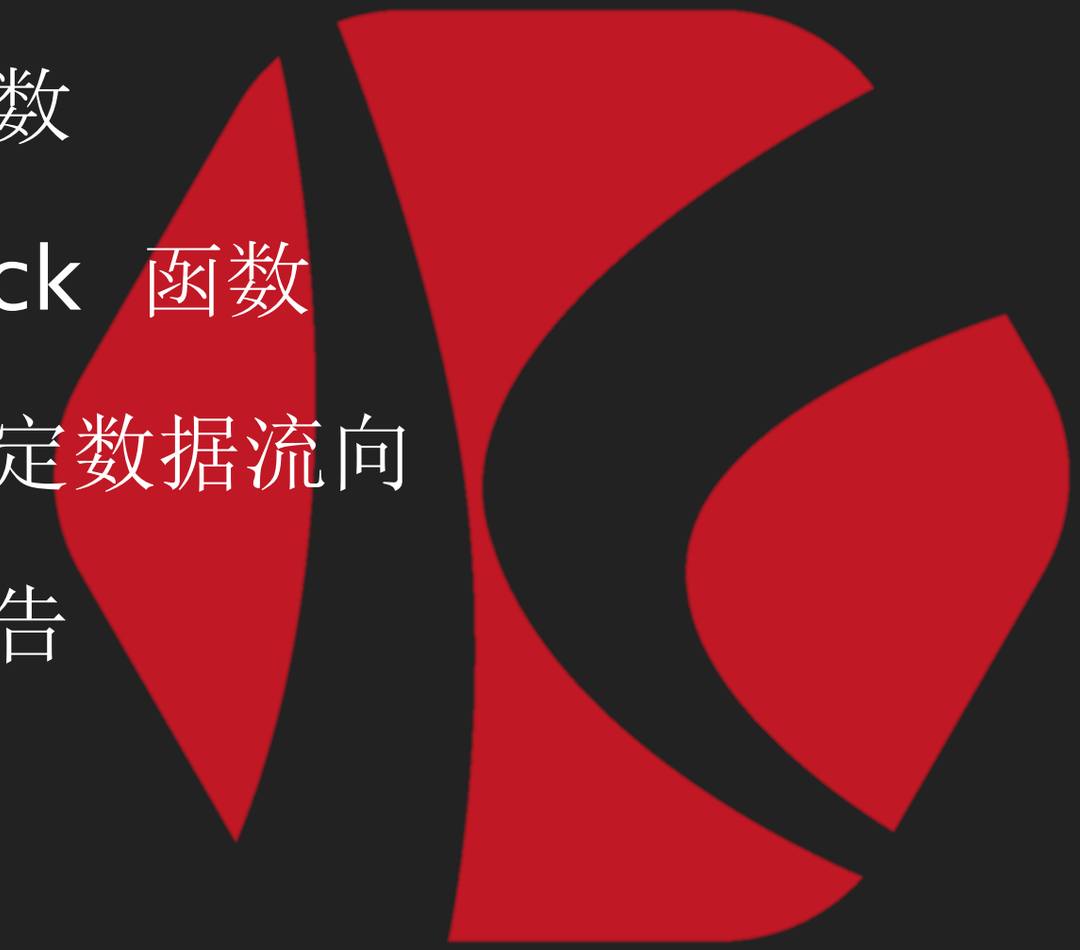


BOOTLOADER 漏洞类型

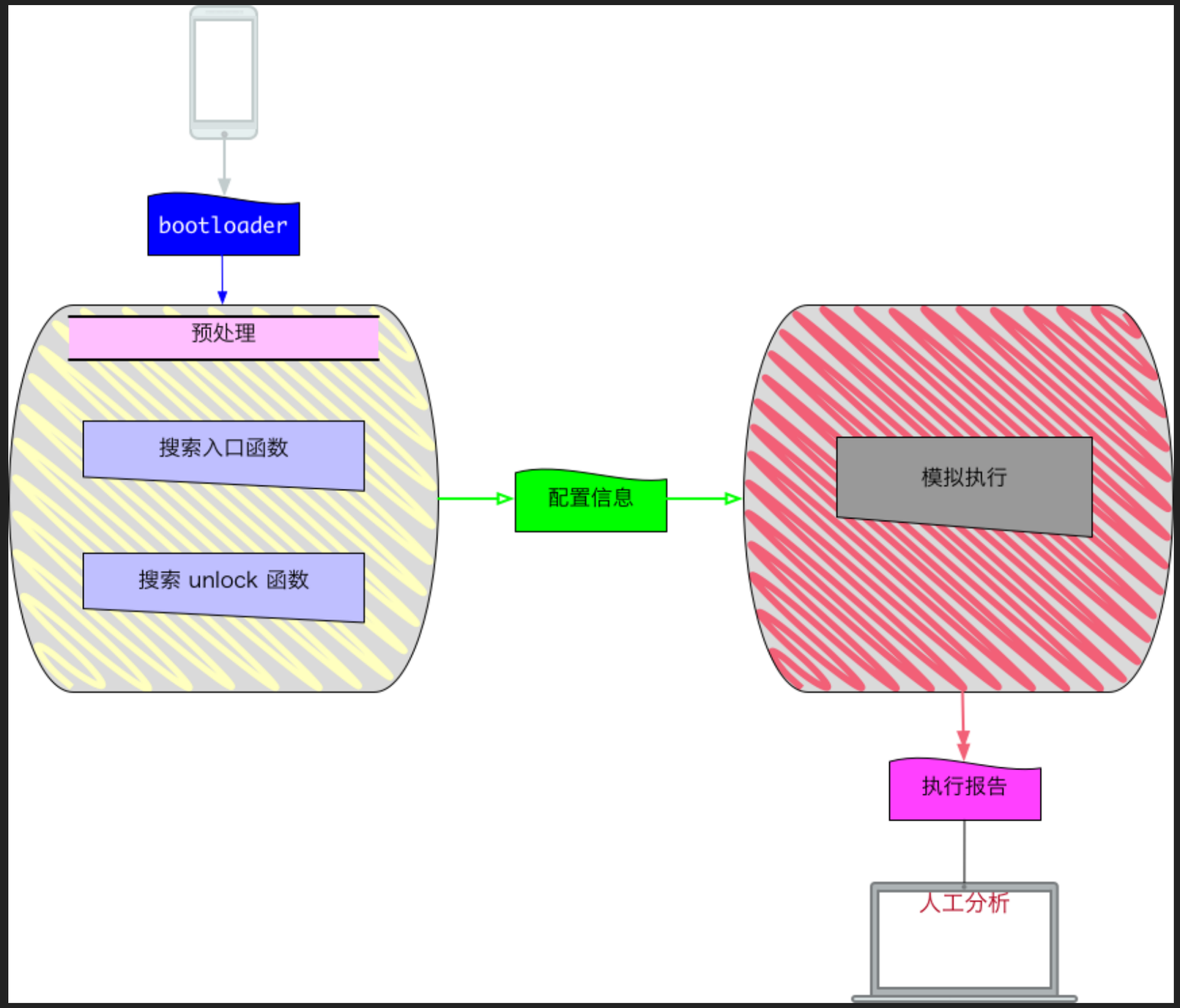
- 内存破坏漏洞
- 存储设备写入漏洞
- Bootloader 解锁漏洞

BOOTLOADER 漏洞挖掘框架

- 提取 bootloader
- 搜索入口函数
- 搜索 unlock 函数
- 模拟执行确定数据流向
- 生成执行报告



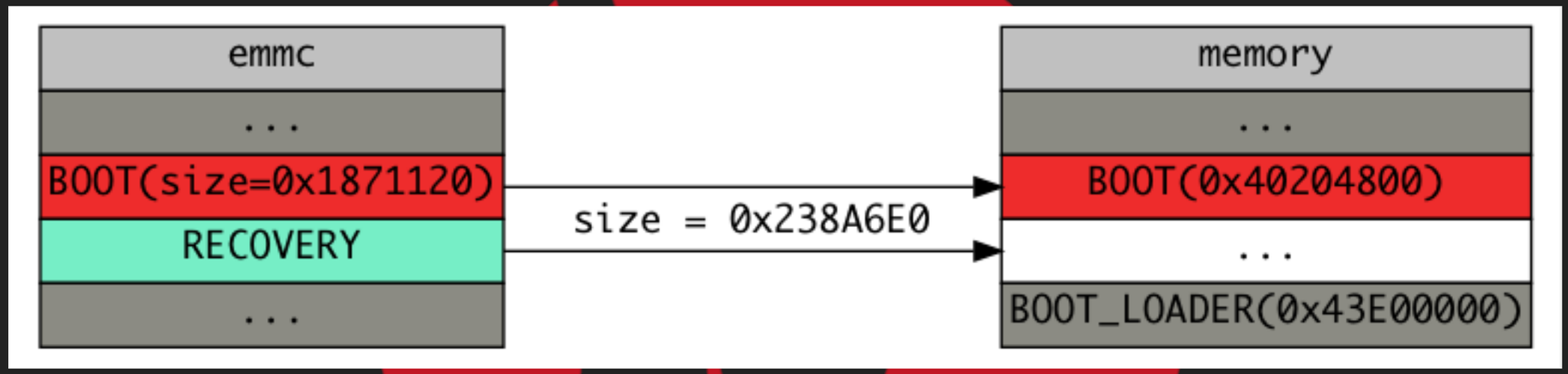
BOOTLOADER 漏洞挖掘框架



05 Bootloader 漏洞分析

CADMIUM

- Boot.img 加载过程



CADMIUM

- 内存破坏漏洞
- 绕过Verified Boot
- 漏洞成因

CADMIUM

```
#define ROUND_TO_PAGE(x, y) (((x) + (y)) & (~(y)))
#define SIGNATURE_SIZE 0x120
char*buf=(char *)0x40204800;

void ufs_read(void *buf, int block, int count);
int signature_check(char *name, void *buf,int length);

int load_kernel(void) {
  struct partition_entry *part;
  struct boot_img_hdr hdr;
  int page_mask;
  int kernel_actual, ramdisk_actual, dt_actual;
  int boot_img_size;

  part = partition_get_by_name("BOOT");

  ufs_read(&hdr, part->start, sizeof(hdr));
  if (memcmp(hdr.magic, "ANDROID!", 8)){
    return -1;
  }

  page_mask = hdr.page_size - 1;
  kernel_actual = ROUND_TO_PAGE(hdr.kernel_size, page_mask);
  ramdisk_actual = ROUND_TO_PAGE(hdr.ramdisk_size, page_mask);
  dt_actual = ROUND_TO_PAGE(hdr.dt_size, page_mask);

  boot_img_size=hdr.page_size+ /*header */
    kernel_actual+ /*kernel */
    ramdisk_actual + /* ramdisk */
    dt_actual; /*device tree */

  ufs_read(buf, part->start, boot_img_size + SIGNATURE_SIZE);
  if (signature_check("BOOT", buf, boot_img_size + SIGNATURE_SIZE))
    return -1;
}
```

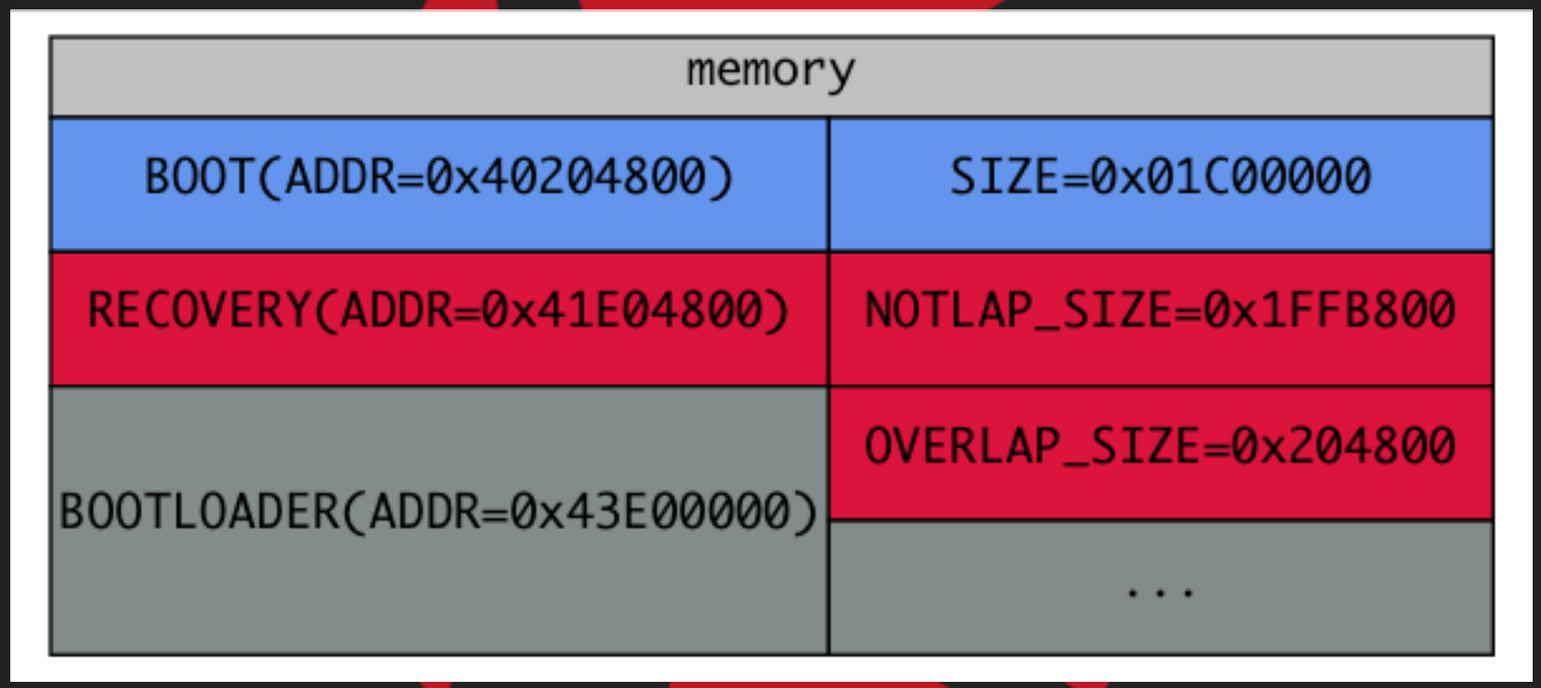
CADMIUM 利用

- Emmc 结构

sda partitions
BOTA0
BOTA1
EFS
m9kefs1
m9kefs2
m9kefs3
PARAM
BOOT
RECOVERY
OTA
RADIO
TOMBSTONES
DNT
PERSISTENT
STEADY
PERSDATA
SBFS
SYSTEM
CACHE
CARRIER
USERDATA

CADMIUM 利用

- Bootloader 覆盖



CADMIUM 利用

- 修复数据

```
#define DSB .byte 0x4f, 0xf0, 0x7f, 0xf5
#define ISB .byte 0x6f, 0xf0, 0x7f, 0xf5

.section ".text.boot"
  ;; init stack
  STP             X29, X30, [SP,#0xFFFFFE80]!
  MOV             X29, SP
  STP             X19, X20, [SP,#0x10]
  STR             X23, [SP,#0x30]
  ADD             X20, X29, #0x40
  MOV             X23, X1
  MOV             X23, X1
  STP             X21, X22, [SP,#0x20]
  ;; restore dt_size actual_size
  MOV             X3,#0
  MOV             X25,#0
  LDR             X3, =0x1871000
  LDR             X25, =0x41A75800
  LDR             W0, =0xAE000
  STR             W0,[X21,#0x28]
  ;; restore stack and return 0
  MOV             W0, #0
  LDR             X23, [SP,#0x30]
  LDP             X19, X20, [SP,#0x10]
  LDP             X21, X22, [SP,#0x20]
  LDP             X29, X30, [SP],#0x180
  RET
```

Thanks !