

你的智能硬件出卖了你的信息 —浅谈办公及教育场景硬件供应链安全

演讲者：秦时、卢昊良、吴帆

议程

- IoT产品供应链的安全现状
- 常见的保护机制及实现缺陷
- 如何避免缺陷，保护供应链安全

IoT产品供应链安全现状

金立2千多部手机被植入木马

软件供应链攻击易受关注，而硬件供应链在流通环节的攻击，非常容易被忽视。



【金立手机植入“木马”被曝光，非法获利2700余万!】

#金立手机植入木马被曝光# 2650台金立手机被植入木马，非法获利2700多万元，手机中非法植入木马案层出不穷。🔗金立手机植入“木马”被曝光，非法获利2700余...



IoT产品供应链安全现状

国内外的相关规范



NIST Special Publication 800-161

Supply Chain Risk Management Practices for Federal Information Systems and Organizations

Jon Boyens
Celia Paulsen
Rama Moorthy
Nadya Bartol

This publication is available free of charge from:
<http://dx.doi.org/10.6028/NIST.SP.800-161>

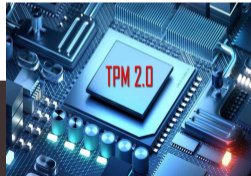
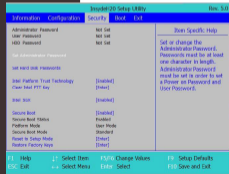
COMPUTER SECURITY

NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

IoT产品供应链安全现状

业界的做法

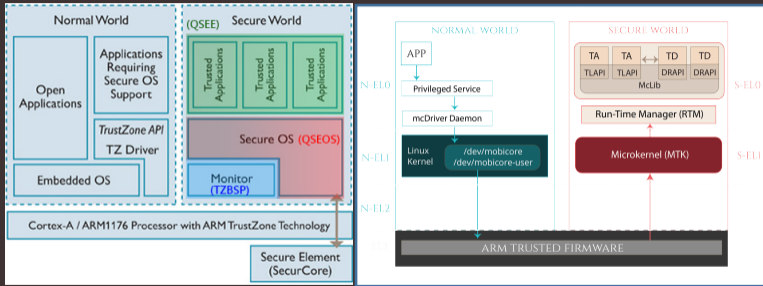
主流的桌面处理器、服务器、笔电、操作系统厂商，很早就实现了安全启动，可信计算等机制。



IoT产品供应链安全现状

移动处理器厂商

高通、MTK等IoT常用方案厂商都提供了完整的SecureBoot、TEE的实现支持，提供了可靠的保护机制。



IoT产品供应链安全现状

实际落实到IoT产品端的情况

在我们的研究中，累计发现5个头部厂商的9款IoT产品（芯片方案有完整的安全启动支持），存在设计缺陷，导致产品可以在供应链流通环节被植入恶意代码。

IoT产品供应链安全现状

只是缺陷，不是漏洞

攻击需要物理接触，
厂商不重视，不认可

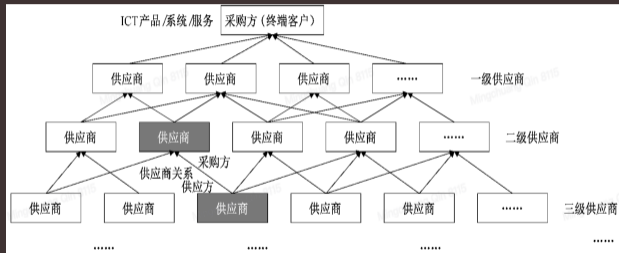


Microsoft replied to Tivadar, saying "Your report requires either physical access or social engineering, and as such, does not meet the bar for servicing down-level (issuing a security patch)."

IoT产品供应链安全现状

虽是缺陷，后果严重

复杂的供应链网络，导致产品在到达客户之前的流通环节，存在大量被供应链植入的时间窗口。



IoT产品供应链安全现状

虽是缺陷，后果严重

智能盒子
智能电视
会议终端



窃取商业机密

IoT产品供应链安全现状

虽是缺陷，后果严重

没有安全保护的教育硬件，可能被破解改变产品原有设计用途，变身为游戏机、浏览不良信息的媒介等。



IoT产品供应链安全现状

虽是缺陷，后果严重

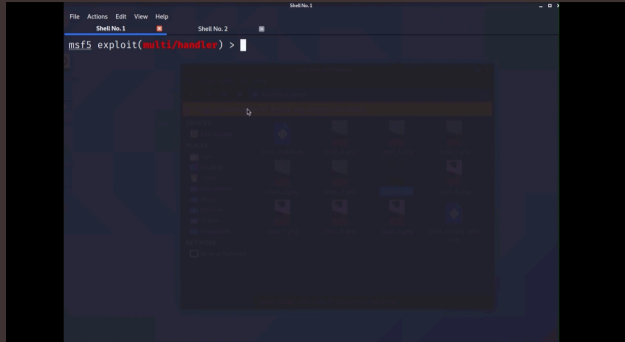
智能音箱
智能教育屏
智能学习灯



监控家庭敏感地带

IoT产品供应链安全现状

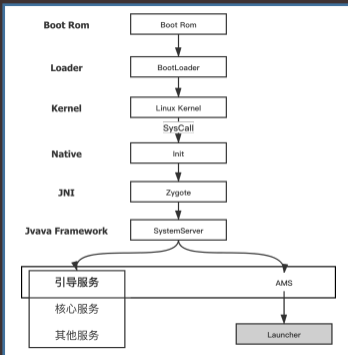
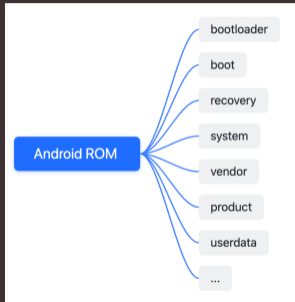
某教育产品供应链植入风险演示



常见的保护机制及实现缺陷

常见的保护机制及实现缺陷

Android ROM



常见的保护机制及实现缺陷

Secure Boot

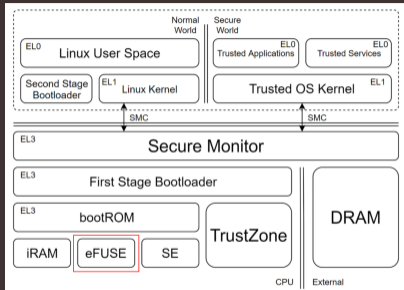


核心思想：当前阶段的启动代码加载下一级代码之前，对所加载的代码基于PKI进行完整性校验。

常见的保护机制及实现缺陷

信任根

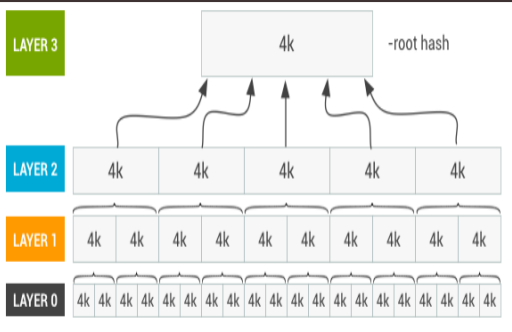
所有支持 Secure Boot的CPU都会有一块很小的OTP储存，也称为 FUSE 或者eFUSE，它的工作原理跟现实中的保险丝类似：在芯片出厂之前会被写入信息，一旦被写入便无法被更改



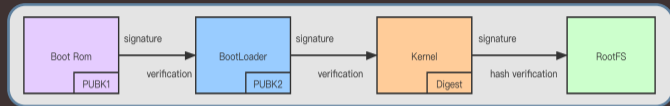
常见的保护机制及实现缺陷

DM-Verity

- 对于小分区，会使用信任根进行直接或间接签名的。
- 对于较大的分区，比如 system 分区，与预置的 root hash 进行比对验证。



常见的保护机制及实现缺陷



信任根 → Boot Verify → DM Verity

牢不可破？

常见的保护机制及实现缺陷

绕过 SECURE BOOT



90%以上均未烧写eFuse



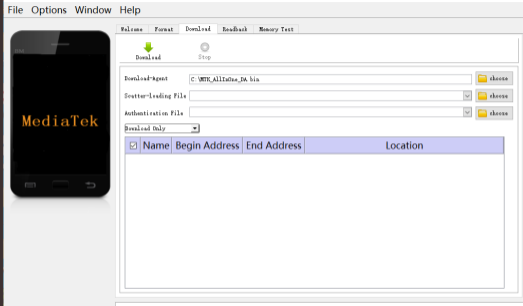
可以直接绕过

虽然安全启动没开，但是从BootLoader向下的保护机制可能是开启的，需要拿到固件，分析保护逻辑。

常见的保护机制及实现缺陷

绕过 SECURE BOOT

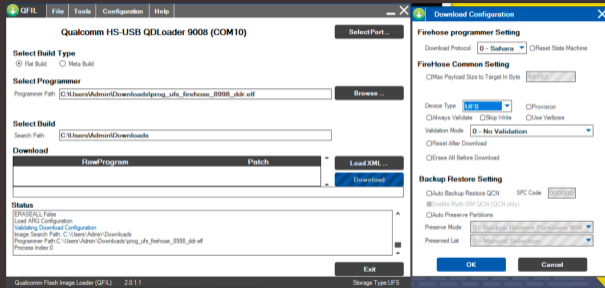
没有开启安全启动，
利用芯片厂商工具读写整个磁盘固件（MTK）



常见的保护机制及实现缺陷

绕过 SECURE BOOT

没有开启安全启动，
利用芯片厂商工具
读写整个磁盘固件
(QUALCOMM)



常见的保护机制及实现缺陷

通过固件分析绕过Boot Verify



```

00000000 ; -----
00000000
00000000 SECURE_CFG_V4  struct ; (sizeof=0x3C, align=0x4, copyof_658)
00000000                                     ; XREF: .bss:g_seccfg_major/r
00000000                                     ; .bss:g_seccfg_bak/r
00000000 seccfg_body      _SECURE_CFG_V4 ?
00000000                                     ; XREF: sec_check_major_magic_seccfg+A/r
00000000                                     ; sec_check_bak_magic_seccfg+A/r ...
0000001C seccfg_cipher  DCB 32 dup(?)
0000003C SECURE_CFG_V4  ends
0000003C
00000000 ; -----
  
```

- frp

seccfg结构体

```

00000000
00000000 _SECURE_CFG_V4  struct ; (sizeof=0x1C, align=0x4, copyof_657)
00000000                                     ; XREF: SECURE_CFG_V4/r
00000000 magic_number     DCD ?
00000000                                     ; XREF: sec_check_major_magic_seccfg+A/r
00000000                                     ; sec_check_bak_magic_seccfg+A/r
00000004 seccfg_ver      DCD ?
00000008 seccfg_size DCD ?
0000000C lock_state    DCD ?
0000000C                                     ; get_lock_state+18/s
00000010 lock_critical_state DCD ?
00000010                                     ; XREF: sec_init_seccfg_major+4C/w
00000014 sboot_runtime DCD ?
00000014                                     ; XREF: sec_init_seccfg_major+5A/w
00000018 end_pattern  DCD ?
00000018                                     ; XREF: sec_init_seccfg_major+52/w
0000001C _SECURE_CFG_V4  ends
  
```

- seccfg

seccfg body 结构体

常见的保护机制及实现缺陷

通过固件分析绕过Boot Verify



```

static int fastboot_get_unlock_perm(int *unlock_allowed)
{
    ...
    ret = partition_exists("frp");
    if (ret == PART_NOT_EXIST) {
        dprintf(CRITICAL, "frp partition does not exist\n");
        *unlock_allowed = 1;
        return OK;
    }

    size = partition_get_size_by_name("frp");
    unlock_allowed_flag_offset = size - 4;

    ret = partition_read("frp", unlock_allowed_flag_offset, (u8 *)unlock_allowed, 4);
    if (ret < 0) {
        *unlock_allowed = 0;
        return ret;
    } //只要frp分区非空且最后4个字节不为0, 即满足unlock

    return OK;
    ...
}
  
```

```

int seccfg_set_lock_state()
{
    ...
    if ( sec_get_seccfg() )
    {
        if ( get_log_level() )
        {
            v5 = dprintf("[%s] seccfg not found!\n", "SEC_UNLOCK");
            return 1;
        }
    }
    else
    {
        v3 = sec_backup();
        *(_DWORD *) (g_seccfg_major[0] + 12) = 3;
        v4 = sec_update(v3);
        sec_erase_bak(v4);
        result = 0;
    }
    ...
}
  
```

常见的保护机制及实现缺陷

通过固件分析绕过DM-Verity



```

ment@7 ..... __overlay__ .....
      fragment@8 ..... __overlay__
      ..... fragment@9 ..... __overla
y__ ..... fragment@10 .....
      __overlay__ system .android,system
4 ./dev/block/platform/soc/11120000.mmc/by-name/system
      =ext4 .ro .wait,verify
      vendor .android,vendor 4 ./dev
/block/platform/soc/11120000.mmc/by-name/vendor =
ext4 .ro .wait,verify
      fragment@11 ..... __overlay__
default hdmi_hpd . .
  
```

常见的保护机制及实现缺陷

Gain Full Access



常见的保护机制及实现缺陷

某会议盒子供应链植入风险
演示



如何避免缺陷，保护供应链安全

如何避免缺陷，保护产品安全

部分厂商的现状

- 安全意识，认为接触式攻击不属于漏洞
- 芯片成本，支持安全启动的芯片增加成本
- 研发成本，不想在保护机制上投入人力物力
- 维修成本，开启安全启动增加维修难度

提高安全意识，将安全特性支持考虑到产品基础成本里。

如何避免缺陷，保护产品安全

部分厂商的保护方案

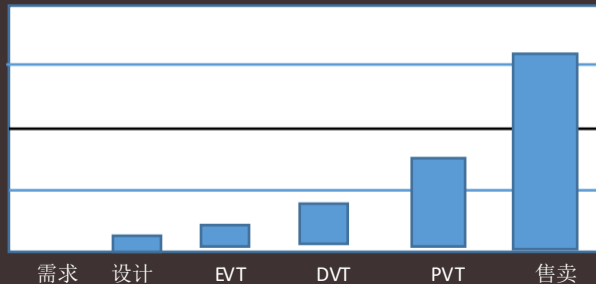
- 采用巧妙的后门，隐藏调试开关。
- 对控制调试的程序进行混淆加密，增加分析成本。
- 在底层Framework上，定制程序的安装逻辑，对抗植入。

完美的保护机制也需要在根源上做到安全。

如何避免缺陷，保护产品安全

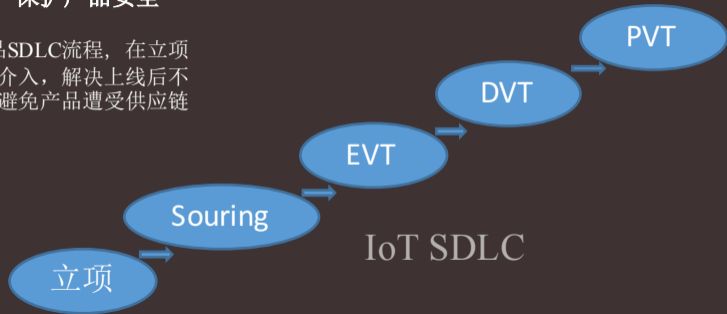
硬件产品不同阶段的缺陷修复需要付出的代价

安全应该尽早的介入到硬件产品的设计研发流程中。





如何避免缺陷，保护产品安全

通过建立IoT产品SDLC流程，在立项阶段就进行安全介入，解决上线后不可召回的漏洞，避免产品遭受供应链植入攻击。



感谢观看！

 知道创宇 |  KCon

