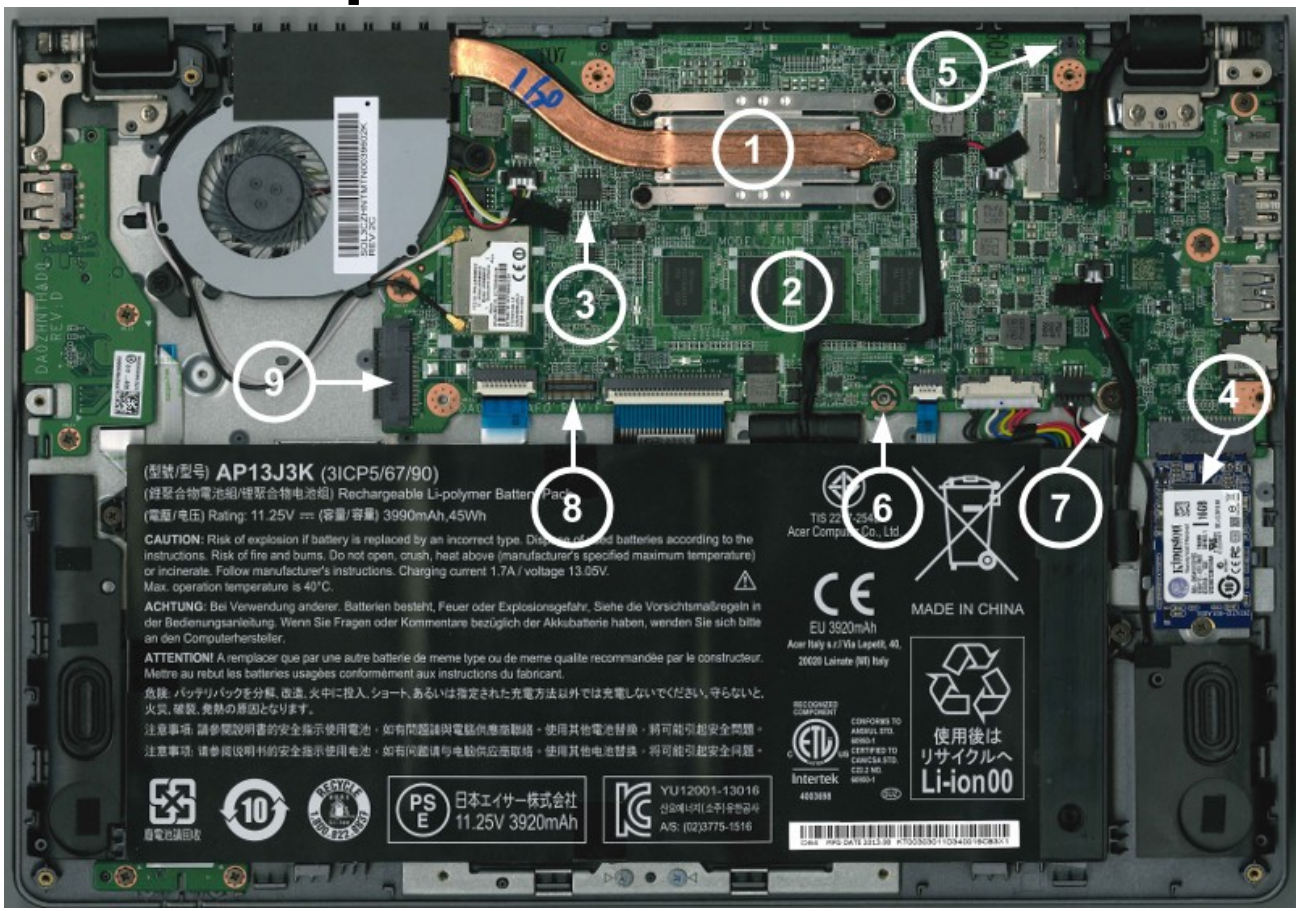
Component Firmware



http://outflux.net/slides/2014/lss/firmware.pdf

Linux Security Summit, Chicago 2014

Kees Cook <keescook@chromium.org>
(pronounced “Case”)

chrome

Overview

- Wait, which firmware?
- Threats
- Update methods
- request_firmware() hooking
- Regression testing
- Future work

Wait, which firmware?

- Not the BIOS, UEFI, or other boot firmware
  - Though consider ME, SMM, TrustZone
- “Component Firmware” in system:
  - Network
  - Storage
  - Input
  - CPU
  - Other weird stuff

Threat: memory access

- Attached via DMA?
  - Example: network card
- Access to physical memory!
- Best protection is to configure IOMMU (when available) to limit the scope of attack

http://esec-lab.sogeti.com/dotclear/public/publications/11-recon-nicreverse_slides.pdf

chrome

Threat: interception and spoofing

- Attached to existing driver
  - Example: Bluetooth keyboard, drives
- On a similar bus? LPC devices can imitate legacy keyboards
  - Example: SuperIO and TPM are on LPC

http://spritesmods.com/?art=hddhack

https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=59565

chrome

Update methods

- None: actual ROMs
- Persistent?
- Over kernel or userspace exposed bus?
- Checksum?
- Signature?
- Documented?
- Once per boot?

Read-only

- Code is part of chip mask
- Pros
  - Cannot be replaced with malicious code
- Cons
  - Cannot be replaced with bug-fixed code without physically replacing the chip

Direct programmed

- Programmable chip but update mechanism not wired to anything
- Pros
  - Cannot be replaced with malicious code
- Cons
  - Cannot be replaced with bug-fixed code without physically attaching chip programmer

Vendor-defined Crypto

- A chip has a vendor-supplied key loaded for crypto verification of the incoming firmware
- Pros
  - Secure firmware updates
- Cons
  - System owner has no access to change firmware
  - Uncommon for component vendors to implement

chrome

System-defined Crypto

- A chip can have a key loaded for crypto verification of the incoming firmware
- Pros
  - Secure firmware updates
  - Key chain defined by system owner
- Cons
  - Very unlikely for component vendors to implement

chrome

Per-power-on Toggle

- A chip will come up allowing updates, but once a flag is set, will reject further updates
- Pros
  - Secure firmware updates
- Cons
  - Only as secure as boot firmware
  - Updates must be applied during initial system boot
  - Boot firmware may need to re-flip bit across power events (e.g. suspend)

chrome

Kernel-only bus

- The bus for communicating updates is only exposed to the kernel (e.g. reserved IO memory on PCI bus)
- Pros
  - Updates possible
- Cons
  - Only as secure as the kernel
  - Need kernel API to verify firmware origin

Userspace bus

- The bus for communicating updates is exposed to userspace (e.g. SCSI generic, i2c, etc)
- Pros
  - Updates possible
- Cons
  - Crosses the line between userspace and ring0
  - May not be limited to root user (e.g. 3G modem “AT” commands)

request_firmware()

- Kernel drivers use this to get firmware loaded from userspace somewhere
- Designed for non-persistent firmware, covers a wide range of existing devices
- Go look in /lib/firmware

kernel_fw_from_file() LSM hook

```
/*
 * @kernel_fw_from_file:
 *  Load firmware from userspace (not called for built-in firmware).
 *  @file contains the file structure pointing to the file containing
 *  the firmware to load. This argument will be NULL if the firmware
 *  was loaded via the uevent-triggered blob-based interface exposed
 *  by CONFIG_FW_LOADER_USER_HELPER.
 *  @buf pointer to buffer containing firmware contents.
 *  @size length of the firmware contents.
 *  Return 0 if permission is granted.
 */
```


request_firmware() validation

- Check origin of fd (e.g. coming from expected read-only filesystem?)
- Check contents (e.g. against known signature)

https://git.kernel.org/cgit/linux/kernel/git/kees/linux.git/log/?h=lsm/modpin

chrome

Regression testing

- CONFIG_TEST_FIRMWARE creates test_firmware.ko that can be used to verify request_firmware calls
- Echo desired firmware name into /sys/devices/virtual/misc/test_firmware/trigger_request
- Check /dev/test_firmware for resulting contents (or dmesg for errors)

Demo!

Future work

- What's the best way to intercept userspace APIs that allow firmware updates?
- How to convince vendors to implement once-per-boot updates?
- Can we modify existing component firmware to implement once-per-boot updates?

Questions?

http://outflux.net/slides/2014/lss/firmware.pdf

keescook@chromium.org

keescook@google.com

kees@outflux.net