Ownership Transfer
Where security meets the circular economy
Jeff Andersen (Google)
Agenda

• Exploring the concept of "ownership"
• Exploring the circular economy
• Methods of ownership transfer
Ownership: Identity Endorsement

UDS = unique device secret
FMC = first mutable code
Ownership: Identity Endorsement

UDS = unique device secret
FMC = first mutable code
Ownership: Identity Endorsement

UDS = unique device secret
FMC = first mutable code
Ownership: Identity Endorsement

- Does not impact on-device state management
- Does not impact the device's behavior
  - Only affects the trustworthiness of the device's attestations
Ownership: Configuration Management

- ROM
- FMC verification key

Verifies and launches

- Signed FMC
- Runtime fw verification key

Verifies and launches

- Signed runtime FW
Ownership: Configuration Management

• Impacts the device's behavior
• Impacts on-device state management
• Is the subject of ownership transfer
Circular Economy

Device manufacturer

OEM

Datacenter operator

Second-hand clearinghouse

Second-hand clearinghouse

Second-hand clearinghouse

Second-hand clearinghouse

Second-hand purchaser

Second-hand purchaser

Second-hand purchaser

Second-hand purchaser

Circular Economy and Statefulness

- Device manufacturer
- Datacenter operator
- Second-hand purchaser
Circular Economy and Statefulness

Identity endorsement: (mostly) stateless
Circular Economy and Statefulness

Device configuration: stateful

- Manufacturer-signed FMC
- Device manufacturer
- DC operator-signed FMC
- Datacenter operator
- Purchaser-signed FMC
- Second-hand purchaser
Ownership transfer definition

- **Persistent device configuration, updateable by mutually-distrusting entities**
  - At time $T=0$, device is controlled by the manufacturer
  - At time $T=1$, device is controlled by the DC operator
  - At time $T=2$, device is controlled by the second-hand purchaser
  - ...

- What ownership transfer is not:
  - Device identity endorsement
  - Unauthenticated commands (ex: changing storage disk encryption keys)
Vendor control over devices

- Ownership transfer is not about giving *total* control to owners
- Vendors may wish to exert ongoing control over device behavior
  - Example: CPUs that only boot vendor-signed microcode
- The spec has an explicit carve-out for vendor control
  - Vendors might not "own" the device, but they control many of its aspects
Ownership transfer goals

Support two competing interests:

• Secure device configuration management
  - "When the device is mine, it does what I say."

• Healthy circular economy
  - "I can easily sell the device to whomever I want."
Ownership transfer on a spectrum

- Fewer security assurances given to owners
- Fewer circular economy roadblocks
- Stronger security assurances given to owners
- Greater circular-economy roadblocks

No ownership transfer
Physical-presence ownership transfer
Authenticated ownership transfer
Not all devices need ownership transfer

- Some devices only need vendor-managed configuration
  - Ex: devices that only ever run vendor firmware

- Caliptra is an example
  - An iRoT that only boots vendor-signed firmware
Physical-presence ownership transfer

- Assert some pin to prove to the device that a prospective owner is physically present
  - Device then accepts a new configuration

- Resell is conceptually simple: buyer receives part, asserts physical presence, pushes their config
  - Devil is in the details: how to assert physical presence across a multitude of device models and form-factors
Authenticated ownership transfer

• Device configuration updated via signed tickets
  - "Configuration ownership" is conferred by way of controlling a private key
  - Physical control is not enough to assert ownership

• **Resell is much more complex**: seller and buyer must coordinate to ensure the buyer can ingest the device
  - Seller and buyer do not always know each other's identity
Terminology: DevUpdtK

Device

DevUpdtK\textsubscript{pub}

Authenticates configuration updates

Authenticates ownership transfer tickets

Owner

DevUpdtK\textsubscript{priv}
Ownership transfer via handshake

1. Get nonce
2. Sign new DevUpdtK\textsubscript{pub} with current DevUpdtK\textsubscript{priv}
3. Provision new signed DevUpdtK\textsubscript{pub}
When we don't know who the next owner is

Two options:

- Transition the device into an "unowned" state before shipping
  - Provision an empty/null DevUpdtK_{pub} to the device

- Use a "service key"
  - A temporary key used to ferry ownership from one entity to another
Device

Owner 1
DevUpdtK_pub

(1) Get nonce

Device
Null DevUpdtK_pub

(2) Sign null DevUpdtK_pub with Owner 1 DevUpdtK_priv

Device
Null DevUpdtK_pub

(3) Provision null DevUpdtK_pub

Device
Null DevUpdtK_pub

(4) Ship device to Owner 2

Device
Owner 2 DevUpdtK_pub

(5) Provision Owner 2 DevUpdtK_pub

Legend
Nonce
Public key
Signature
(1) Get nonce
(2) Generate service key
(3) Sign ServiceK\(_{pub}\) with Owner 1 DevUpdtK\(_{priv}\)
(4) Provision signed ServiceK\(_{pub}\)
(5) Ship device to Owner 2
(6) Transmit ServiceK\(_{priv}\)
(7) Get nonce
(8) Sign Owner 2 DevUpdtK\(_{pub}\) with ServiceK\(_{priv}\)
(9) Provision signed Owner 2 DevUpdtK\(_{pub}\)
Handling RMA and Decom

• These flows all assume that ownership transfer can occur before the device leaves possession of the current owner
  - This assumption does not hold in two common cases

• **RMA:** Assembly is broken; techs ship it back to the vendor
  - The device is functional and may return to the owner, but it was unreachable when the assembly left the owner

• **Decom:** Rack is powered down, and later stripped for parts
  - Decision to decom is not made until after the rack is powered down
Approach: deferred ownership transfer

• Make ownership-transfer nonces long-lived
  - Call them "ownership epochs" instead
  - May be deterministic based on an internal monotonic counter

• Owners pre-fetch device ownership epochs
  - Store them in a database

• Owners can generate and sign an ownership transfer ticket after the device goes out the door
  - Signed tickets are transferred to the subsequent owner out-of-band
1. Get epoch, store in DB

2. Ship device to Owner 2

3. Sign null \( \text{DevUpdtK}_{\text{pub}} \) with Owner 1 \( \text{DevUpdtK}_{\text{priv}} \)

4. Transmit signed unlock ticket

5. Provision signed unlock ticket

6. Provision Owner 2 \( \text{DevUpdtK}_{\text{pub}} \)
Device

(1) Get epoch, store in DB

Owner 1

(3) Sign null DevUpdtK\textsubscript{pub} with Owner 1 DevUpdtK\textsubscript{priv}

Ownership epoch
Null DevUpdtK\textsubscript{pub}

(4) Transmit signed unlock ticket

Owner 2

Epoch DB
Clients pull down any tickets needed to take ownership of devices they have acquired.
Challenges with ownership transfer

- Device-side implementation questions
  - Storage substrate: fuses vs flash
  - Where and how to route a physical-presence signal

- Coordination problems - central databases are tricky
  - Database reliability
  - Owner reliability
  - Ownership transfer confidentiality
Ownership transfer on a spectrum

- No ownership transfer
  - Fewer security assurances given to owners
  - Fewer circular economy roadblocks

- Physical-presence ownership transfer
  - Stronger security assurances given to owners
  - Greater circular economy roadblocks

- Authenticated ownership transfer
  - Stronger security assurances given to owners
  - Greater circular economy roadblocks
Ownership transfer on a spectrum

- Fewer security assurances given to owners
- Fewer circular economy roadblocks
- Stronger security assurances given to owners
- Greater circular economy roadblocks

- No ownership transfer
- Physical-presence ownership transfer
- Authenticated ownership transfer
- Volatile ownership authorization
Volatile ownership authorization

• $DevUpdtK_{\text{pub}}$ is provisioned into the device on every cold boot
  - Latched into device RAM
  - Authorizes runtime device configuration updates

• Device attests to its current $DevUpdtK_{\text{pub}}$
  - Owners can refuse to admit a misconfigured device into a serving state

• Device is "cryptographically stateless" w.r.t. config management
  - Owners can yank power and sell the device on, with no fuss
DICE in Caliptra

- Caliptra firmware
  - Manufacturer CA
    - Manufacturer CA
  - IDevID
    - IDevID
  - FMC
    - FMC CDI
      - Runtime fw
        - Runtime fw CDI
          - Runtime Alias
            - Runtime Alias
  - Signs
  - Signs
  - Signs

Vendor signature
DICE in Caliptra

- Co-signs Caliptra firmware
- Latched into Caliptra RAM on cold-boot
- Authorizes runtime Caliptra updates
Current status

- High-level ownership transfer document under revision and review
  - Physical-presence ownership transfer is preferred
  - Authenticated ownership transfer is deprioritized
  - Volatile ownership authorization will be under discussion
Ownership Transfer
Where security meets the circular economy
Jeff Andersen (Google)

Thanks!
Q&A