Open Source Firmware @ Facebook

David Hendricks: Firmware Engineer
Andrea Barberio: Production Engineer

Agenda

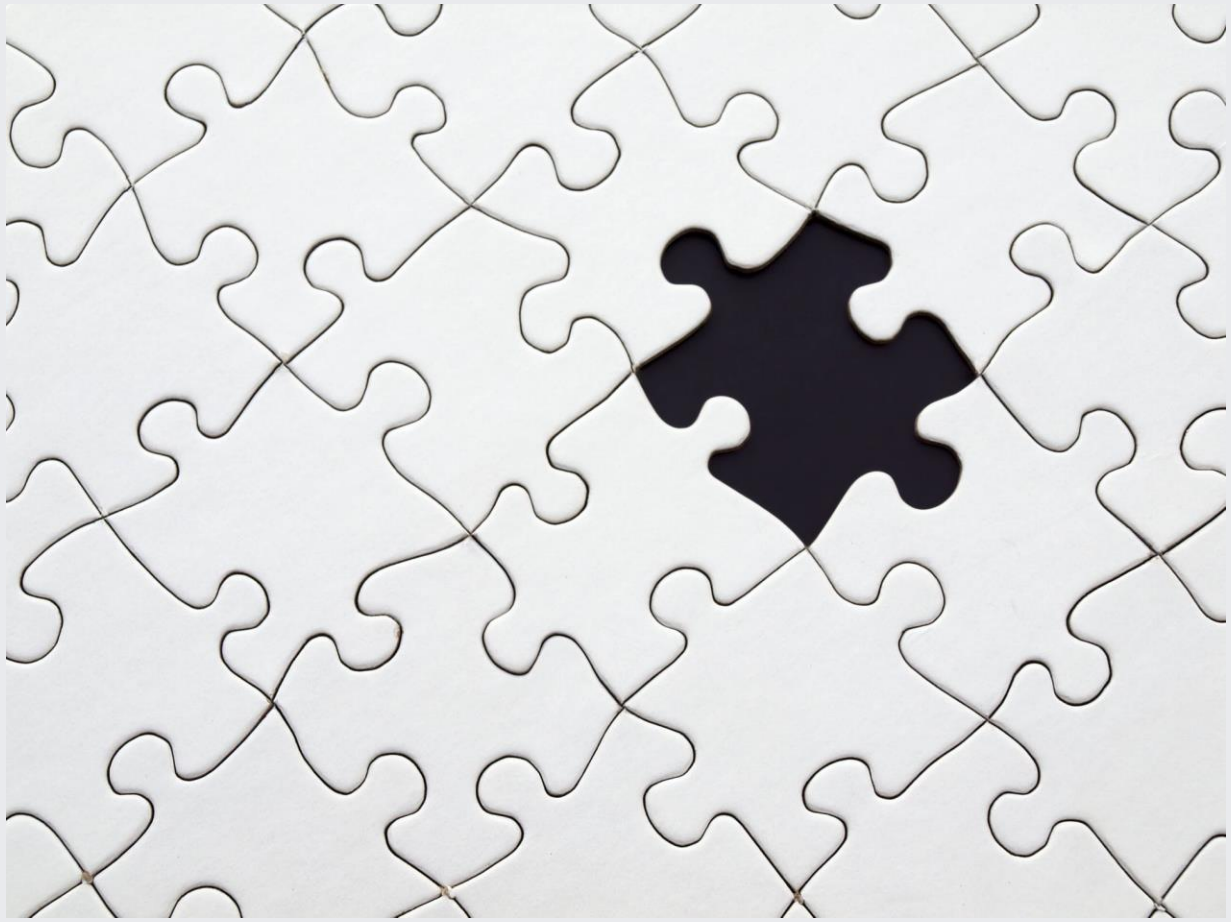
- Open source initiatives
- Problem statement
- How we're using LinuxBoot at Facebook
- systemboot and provisioning
- Collaborations and partnerships

Open Source @ Facebook

- Facebook promotes open source
  - Systems Software: Kernel, CentOS, chef, systemd, etc.
  - Hardware: Open Compute Project, Telecom Infrastructure Project
  - Lots more: https://github.com/facebook and https://github.com/facebookincubator

...but there is a missing piece

Any guesses?



Open Source Firmware @ Facebook

OpenBMC initially released in 2015 and is quickly becoming standard on OCP hardware

System firmware is the next logical step

Why open source firmware?

Open Source Firmware @ Facebook

Scoping out the problem

That's a lot of servers

(and switches, too!)

...and we're not just working on datacenters.

Booting is hard

- Ever-increasing amount of hardware
  - Many local/removable storage media and networking devices
  - Complex setup, complex protocols
- Firmware has become an operating system
- More demands for firmware security
  - Verified/secure boot, measured/trusted boot, attestation
  - Secure network protocols, crypto
- Provisioning is hard

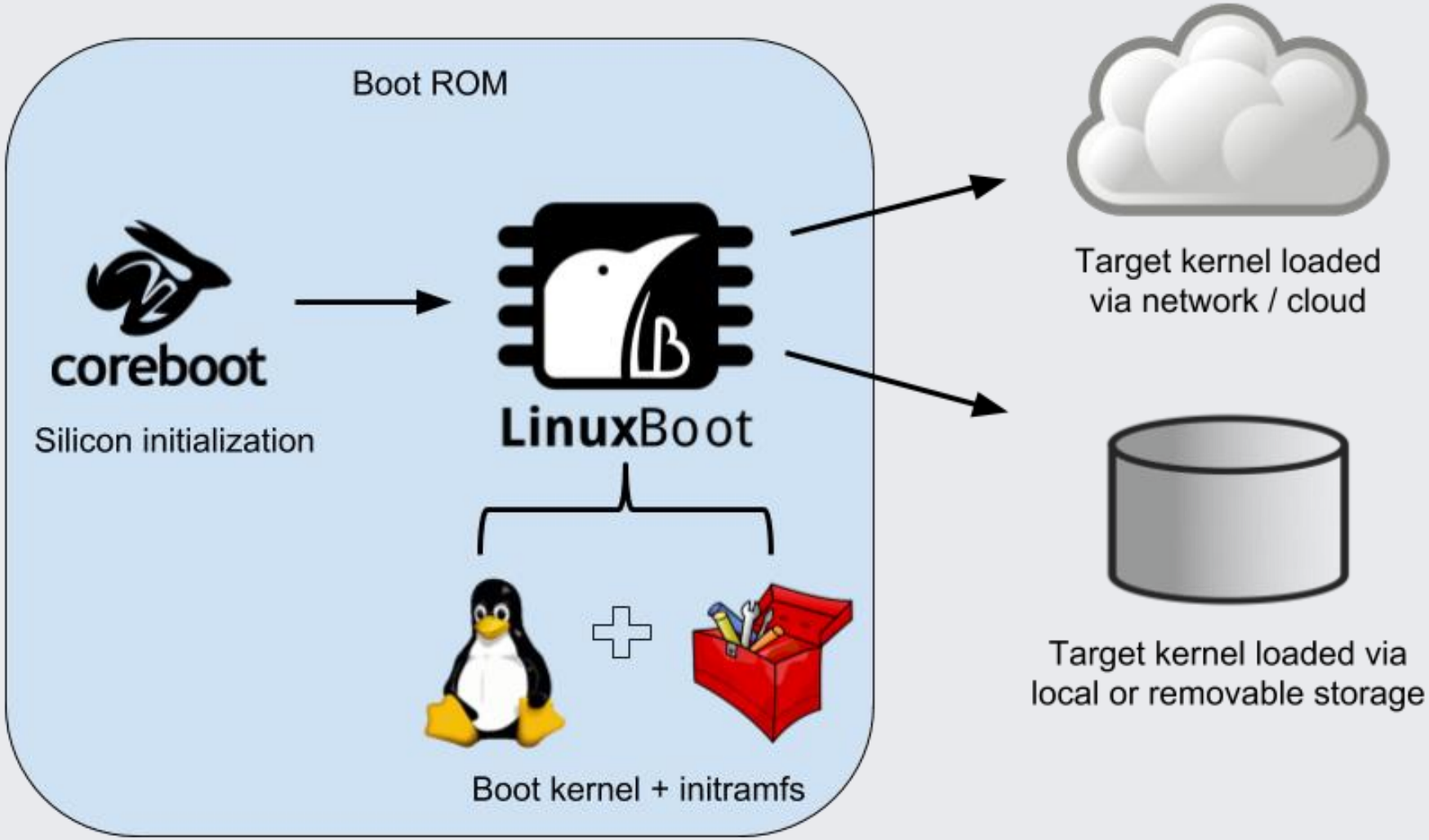
Problems with closed firmware

- Archaic, complex, often quite buggy
  - Even open firmwares are often unfamiliar and difficult to extend
- Reactive instead of Proactive debugging
- Hard to maintain, can't forward/backport features and fixes
- Vendor-specific tools
- "Dimensions" of supporting firmware at scale
  - Robustness, flexibility, debugging, build and deployment...

Sustaining Operations

- Firmware has impact across product lifecycle
  - Design, build, test, deploy, sustain, decommission
- What we want:
  - Support many generations of equipment
  - Feature parity
  - Unified, adaptable toolkit
  - Must support many different use cases
  - **Familiar / low barrier to entry**

How we're addressing the problem



Why LinuxBoot

- We use Linux... a lot
- Production-quality drivers, networking, crypto
- Versatility
  - Can be used on anything that is intended to run Linux.
- We have engineering teams who understand Linux very well
  - Leverage talent we already have
- General goodness that open source brings
  - Auditability, portability, modern development, collaboration, ...

Why LinuxBoot (cont'd)

- LinuxBoot enables us to...
  - Simplify sustaining operations
  - Maximize code reuse
  - Share tools across all products
  - Apply processes and best practices uniformly
  - Have higher eyeball-to-code ratio

Current projects

- Supporting coreboot + LinuxBoot on a few projects
- Open Cellular:
  - Rotundu, based on Intel Atom E38xx

    Elgon, based on Cavium CN81xx (ARM64)
- Open Compute Project:
  - Mono Lake, based on Broadwell-DE
  - Wedge 100S, based on Broadwell-DE
- Our LinuxBoot distribution uses u-root with systemboot
  - Our infrastructure provisioning system also uses u-root
  - Same team can support both pre- and post- boot phases

OS provisioning

OS provisioning

- Installing an OS on a single machine is simple
- Installing an OS at scale is complex
  - Lots of moving parts
  - Network booting introduces noise

- Provisioning flow:
  - Power on
  - DHCPv6 (firmware)
  - TFTP (firmware)
  - installer starts

Boot process issues

- DHCP implementations can have bugs
- TFTP implementations can have bugs
- Different firmwares can have different implementations and bugs
- At scale, a small fraction of errors can be a lot of operations

- What we need
  - reliable clients
  - better protocols
  - control the implementation: know what you run, fix it, improve it

LinuxBoot in provisioning

- LinuxBoot can simplify provisioning a lot
  - Tested DHCP or TFTP implementations
  - HTTPS instead of TFTP
  - We can run consistent firmware versions everywhere
  - We know and control the firmware that we run
- We expect to largely reduce netboot failures in provisioning with this approach
- Open means: Auditability, debuggability, security model, portability, modern development, collaboration

LinuxBoot as OS installer

- LinuxBoot is not just for firmwares
- Its components can be successfully used as a bootloader or an OS installer
  - We want to boot the infra with the same code that provisions our infra
- Facebook is experimenting systemboot as:
  - Local bootloader and installer: ProvLauncher
  - Network installer: YARD

systemboot

systemboot

A bootloader distribution based on u-root

- systemboot is a "distro" that implements a bootloader
  - Based on u-root, that we are contributors of
  - Written entirely in Go
  - Provides tools for different boot scenarios

- The goal is to create components that we can iterate fast on
  - Generic and stable ones will be contributed back to u-root

systemboot – what's inside?

- **netboot**: boot a kernel over the network using DHCPv6 or SLAAC, HTTP(s), and kexec
- **localboot**: boot a kernel from disk, using Grub/Grub2 config or direct device/kernel lookup
- **LinuxBoot VPD**: non-volatile variables storage
- **Booters interface**: a way to define something that can boot
- High level **TPM library**, and userspace utility **TPMTool**
- **uinit**: wrap all of the above in an executable to run at boot time

netboot

- Used to boot a device over the network
- Three phases
  - Acquire network configuration: DHCPv6, SLAAC (DHCPv4 coming soon)
  - Download a kernel image via HTTP or HTTPS
    - Example: DHCPv6 can give us an URL to download the kernel from
  - kexec into the kernel, using the specified command-line arguments

localboot

- Similar to netboot, but used to boot a local kernel
- Phases
  - Scan for local disks
  - Find a grub/grub2 config in a suitable location
  - Find kernel, cmdline and initramfs config
  - Kexec into kernel with the above info
- Alternatively can use boot variables for kernel/ramfs/cmdline

VPD library

- Vital Product Data
- Key-value store on the flash chip
- Based on ChromeOS's VPD format
- Used for non-volatile storage, similar to UEFI variables
  - We use it to store boot configuration (netboot and localboot config)
  - Can be extended to other uses
  - If you don't like VPD, can be easily swapped out

Boot order

- Boot order is stored in VPD variables
- Value in JSON format. Example:

  - **Boot0000**={
    "type":"netboot",
    "method":"dhcpv6",
    "mac":"00:fa:ce:b0:0c:00"
    }

  - **Boot0001**={
    "type": "localboot",
    "method": "grub"
    }
  - **Boot0002**={
    "type": "localboot",
    "kernel": "/path/to/kernel",
    "device_guid": "….",
    }

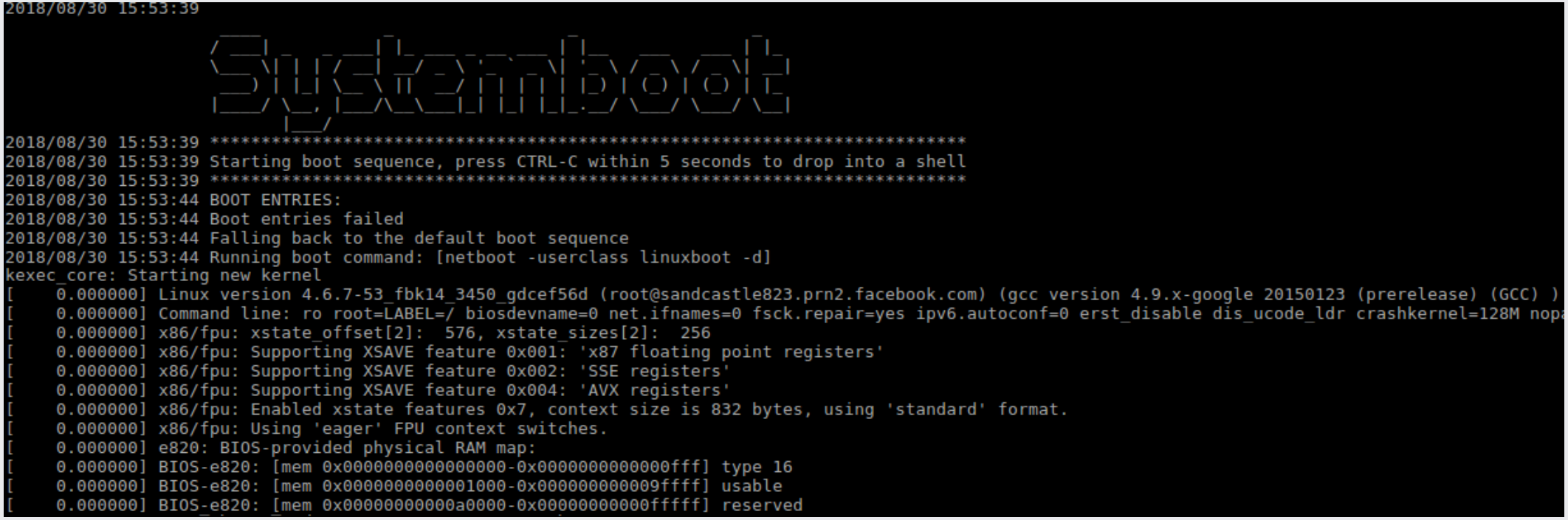
Booters interface

- A generic interface to create new booters
  - netboot and localboot are based on it
  - New booters can implement it
  - You can implement higher level policies, e.g. recovery from failed boot
- Very simple
  - define TypeName() and Boot() methods
  - Define JSON format by extending the generic booter JSON
  - Register the booter, and systemboot will pick it up

TPM library and TPMTool

- High-level TPM library
  - Goal: simplify the use of the TPM
  - Based on Google's go-tpm
  - Parts of it have been merged in go-tpm
  - Can show info, take and clear TPM ownership, seal/unseal, dump PCRs, pre-calculate hashes, dump TPM event log, and more
- TPMTool
  - High-level userspace utility for TPM
  - Written by Philipp Deppenwiese / 9elements CyberSecurity
  - See tpmtool.org

Systemboot: how does it look like?



(demo time)

Future work

- Implement different security models:
  - Boot configurations (almost completed)
  - Boot EFI binaries (partially implemented)
  - Measured boot
  - Verified boot

Bringing it all together

Open Source Firmware@FB

- Improving while simplifying our boot flow
- Enabling collaboration inside and outside of Facebook
  - Industry initiatives such as OCP and TIP
- Opening up firmware to be more inclusive
  - Turning our Linux engineers into firmware engineers

Thanks!

Questions?

Additional resources:

- tpmtool.org
  u-root.tk
- systemboot.org
- linuxboot.org
- opencompute.org
- telecominfraproject.com