Make your system firmware faster, more flexible and reliable with LinuxBoot

David Hendricks: Firmware Engineer
Andrea Barberio: Production Engineer
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](https://github.com/facebook) and [https://github.com/facebookincubator](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
**System firmware in a nutshell**

- First bit of code that runs when CPU is turned on
- Sometimes referred to as "BIOS"
Problem: Local booting is more complex

<table>
<thead>
<tr>
<th>Then</th>
<th>Now</th>
</tr>
</thead>
<tbody>
<tr>
<td>Few, simple interfaces</td>
<td>Many interfaces and protocols</td>
</tr>
</tbody>
</table>
Problem: Local booting is more complex

<table>
<thead>
<tr>
<th>Then</th>
<th>Now</th>
</tr>
</thead>
<tbody>
<tr>
<td>Few, simple interfaces</td>
<td>Many interfaces and protocols</td>
</tr>
<tr>
<td>Simple, low-speed links</td>
<td>High-speed links</td>
</tr>
</tbody>
</table>
Problem: Local booting is more complex

<table>
<thead>
<tr>
<th>Then</th>
<th>Now</th>
</tr>
</thead>
<tbody>
<tr>
<td>Few, simple interfaces</td>
<td>Many interfaces and protocols</td>
</tr>
<tr>
<td>Simple, low-speed links</td>
<td>High-speed links</td>
</tr>
<tr>
<td>Blindly execute MBR (CHS 0/0/1)</td>
<td>Decrypt &amp; mount filesystem</td>
</tr>
</tbody>
</table>
Problem: Network booting is more complex

<table>
<thead>
<tr>
<th>Then</th>
<th>Now</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small, trusted networks</td>
<td>Global, untrusted networks</td>
</tr>
</tbody>
</table>
**Problem: Network booting is more complex**

<table>
<thead>
<tr>
<th>Then</th>
<th>Now</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small, trusted networks</td>
<td>Global, untrusted networks</td>
</tr>
<tr>
<td>Few, simple interfaces and protocols</td>
<td>Many interfaces and protocols</td>
</tr>
</tbody>
</table>
Problem: Network booting is more complex

<table>
<thead>
<tr>
<th>Then</th>
<th>Now</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small, trusted networks</td>
<td>Global, untrusted networks</td>
</tr>
<tr>
<td>Few, simple interfaces and protocols</td>
<td>Many interfaces and protocols</td>
</tr>
<tr>
<td>TFTP/PXE, security an afterthought</td>
<td>TLS/HTTPS, designed for security</td>
</tr>
</tbody>
</table>
TL;DR: SysFW is complex

<table>
<thead>
<tr>
<th>Then/Now</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysFW/BIOS contains an OS</td>
</tr>
</tbody>
</table>
TL;DR: SysFW is complex

<table>
<thead>
<tr>
<th>Then/Now</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysFW/BIOS contains an OS</td>
</tr>
<tr>
<td>Opaque</td>
</tr>
<tr>
<td>Proprietary ecosystem</td>
</tr>
</tbody>
</table>
TL;DR: SysFW is complex

Then/Now

<table>
<thead>
<tr>
<th>Then/Now</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysFW/BIOS contains an OS</td>
</tr>
<tr>
<td>Opaque</td>
</tr>
<tr>
<td>Proprietary ecosystem</td>
</tr>
<tr>
<td>Vendor-specific tooling</td>
</tr>
<tr>
<td>Product-specific</td>
</tr>
</tbody>
</table>
**TL;DR: SysFW is complex**

<table>
<thead>
<tr>
<th>Then/Now</th>
<th>Now/Future</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysFW/BIOS is an OS</td>
<td>Let Linux Do It</td>
</tr>
<tr>
<td>Opaque</td>
<td></td>
</tr>
<tr>
<td>Proprietary ecosystem</td>
<td></td>
</tr>
<tr>
<td>Vendor-specific tooling</td>
<td></td>
</tr>
<tr>
<td>Product-specific</td>
<td></td>
</tr>
</tbody>
</table>
The Solution: Let Linux Do It

<table>
<thead>
<tr>
<th>Then/Now</th>
<th>Now/Future</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysFW/BIOS is an OS</td>
<td>Let Linux Do It</td>
</tr>
<tr>
<td>Opaque</td>
<td>Open, well-understood at FB</td>
</tr>
<tr>
<td>Proprietary ecosystem</td>
<td>Auditable, debuggable</td>
</tr>
<tr>
<td>Vendor-specific tooling</td>
<td></td>
</tr>
<tr>
<td>Product-specific</td>
<td></td>
</tr>
</tbody>
</table>
The Solution: Let Linux Do It

<table>
<thead>
<tr>
<th>Then/Now</th>
<th>Now/Future</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysFW/BIOS is an OS</td>
<td>Let Linux Do It</td>
</tr>
<tr>
<td>Opaque</td>
<td>Open, well-understood at FB</td>
</tr>
<tr>
<td>Proprietary ecosystem</td>
<td>Auditable, debuggable</td>
</tr>
<tr>
<td>Vendor-specific tooling</td>
<td>Open-source tools</td>
</tr>
<tr>
<td>Product-specific</td>
<td>Portable, re-usuable</td>
</tr>
</tbody>
</table>
Let Linux Do it

- Put a kernel+initramfs in boot ROM
- Do minimal silicon init and jump to Linux as soon as possible
- Use Linux to boot Linux
Let Linux Do it

- Put a kernel+initramfs in boot ROM
- Do minimal silicon init and jump to Linux as soon as possible
- Use Linux to boot Linux
- Production-quality drivers, networking
- Add features + tools as needed
- Debug, build, deploy on our schedule
Let Linux Do it

- Put a kernel+initramfs in boot ROM
- Do minimal silicon init and jump to Linux as soon as possible
- Use Linux to boot Linux
- Production-quality drivers, networking
- Add features + tools as needed
- Debug, build, deploy on our schedule
- Flexible security architecture
- Boot in seconds, not minutes
- Bring modern, open-source development to the firmware
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.
OS provisioning
OS Provisioning

- Andrea Barberio – Host Provisioning Engineering @ Facebook

- 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 a physical machine

• From the machine's perspective:
  • Power on
  • DHCPv6 (firmware)
  • TFTP (firmware)
  • installer starts
Boot process issues

- It works most of the times
- However:
  - DHCP and TFTP implementations can have bugs
  - Different firmwares can have different bugs
  - Fixing one firmware doesn't fix the others
- At scale, a small fraction of errors can translate to a lot of operations
LinuxBoot in provisioning

- LinuxBoot can simplify provisioning a lot
  - Tested DHCP/TFTP implementations
  - Better protocols: HTTPS instead of TFTP
  - Consistent firmwares everywhere
  - We know and control what we run
Testing and upgrading firmware now depends on vendors
- Different vendors have different standards and response time
- Vendors may be unable to reproduce the issue on their infra

On our side:
- Debugging closed source firmware can be hard
- Once the update is ready, we run our validation

Rinse and repeat

The time between bug identification and roll-out to prod can be very long
We want to speed this process up, and enable in-house debugging
LinuxBoot allows us to do this
Not just firmware: 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
LinuxBoot architecture @ FB
Architecture

coreboot, LinuxBoot, u-root, systemboot?

Multiple open-source components:

- **coreboot**: low-level hardware initialization
- **Linux**: device drivers, network stack, multiuser/multitask environment, etc
- **u-root**: user-space environment with command-line utilities
- **systemboot**: additional tools, and bootloader "personality"
u-root
User-space initramfs written in Go

Think of it like busybox, but written in Go

- Multi-architecture
- Single binary, all the tools built-in, symlink determines what to run
- Alternatively, source mode: modify and recompile on the fly
- Fast build time: <10s on a modern laptop
- Created at Google; contributors from Facebook, 9elements, and several others
systemboot
A bootloader distribution based on u-root

- systemboot is a "distro" that implements a bootloader
  - Based on u-root, also written in Go

We want components that provide flexibility in various boot scenarios, and that we can iterate fast on
systemboot workflow

- Look for boot entries in VPD vars: Boot0000, Boot0001, ...
- Find a *Booter* for the boot entry, and try it
- If it fails, try the next boot entry, until one succeeds
- If all fails, start over
Boot entries

- Boot entries and their order are stored in VPD variables
- Value in JSON format. Example:

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

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

Use the **u-root** ramfs builder and a valid kernel:

```bash
  go get -u github.com/u-root/u-root
  go get -u github.com/systemboot/systemboot/{uinit,localboot,netboot}
  "${GOPATH}/bin/u-root -build=bb core \
   github.com/systemboot/systemboot/{uinit,localboot,netboot}
```

• Try it!

```bash
  qemu-system-x86_64 -nographic -kernel /path/to/your/kernel \
   -initramfs /tmp/initramfs.linux_arm64.cpio
```
**Booster interface**

Can be used to

- Implement new boot methods
  - e.g. “brute-force” bootloader

- Define new boot policies
  - e.g. fail if signature is bad; or continue and leave it to remote attestation

- Implementation:
  - Define JSON structure and custom `Boot()` method
Example: netbooter

https://github.com/systemboot/systemboot/blob/master/pkg/booter/netbooter.go

type NetBooter struct {
    Type string `json:"type"
    Method string `json:"method"
    MAC string `json:"mac"
    OverrideURL string `json:"override_url,omitempty"
}
func (nb *NetBooter) Boot() error {
    bootcmd := []string{"netboot", "-d", "-userclass", "linuxboot"}
    cmd := exec.Command(bootcmd[0], bootcmd[1:][...])
    if err := cmd.Run(); err != nil {
        return fmt.Errorf("Error executing %v: %v", cmd, err)
    }
    return nil
}
Systemboot demo
Thanks!

Questions?

David Hendricks <dhendrix@fb.com>
Andrea Barberio <barberio@fb.com>

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